НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебсервіс для автоматизованого аналізу даних і генерації звітів

Виконав студент IV курсу, групи ІП-11
(шифр групи)

Ляля Іван Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліхоузова Т. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Солдатова М. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Лялі Івану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебсервіс для автоматизованого аналізу даних і генерації звітів

керівник проєкту Ліхоузова Тетяна Анатоліївна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Передпроєктне обстеження предметної області: постановка завдання дипломного проєктування, аналіз предметної області, аналіз існуючих рішень, аналіз відомих програмних продуктів, аналіз відомих алгоритмічних та технічних рішень, аналіз та моделювання бізнес-процесів.
2) Розроблення вимог до програмного забезпечення: варіанти використання програмного забезпечення, розроблення функціональних вимог, розроблення не функціональних вимог, аналіз системних вимог, аналіз економічних показників програмного забезпечення, постановка завдання на розробку програмного забезпечення.
3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, архітектурні рішення та обґрунтування вибору засобів розробки, конструювання програмного забезпечення, моделювання та валідація вхідних даних, розробка алгоритмів обробки даних, опис структури

сховища даних, модульність та структура вебзастосунку, розробка механізму обробки помилок, інструменти розробки.
4) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис процесів тестування, опис контрольного прикладу.
5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу
1) Схема структурна варіантів використань
2) Архітектура програмного забезпечення
3) Моделі бізнес-процесів

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	01.04.2025	
3	Постановка та формалізація задачі	19.04.2025	
4	Розробка інформаційного забезпечення	23.04.2025	
5	Алгоритмізація задачі	27.04.2025	
6	Обґрунтування вибору використаних технічних засобів	30.04.2025	
7	Розробка програмного забезпечення	05.05.2025	
8	Налагодження програми	10.05.2025	
9	Виконання графічних документів	20.05.2025	
10	Оформлення пояснювальної записки	29.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент _____ Іван ЛЯЛЯ
(підпис) (ініціали, прізвище)

Керівник _____ Тетяна ЛІХОУЗОВА
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 37 таблиць, 28 рисунків та 36 джерел – загалом 81 сторінка.

Дипломний проєкт присвячений розробці вебсервісу для автоматизованого аналізу даних та генерації звітів.

Мета проєкту – прискорення та спрощення процесу підготовки даних до подальшого аналізу, зокрема із застосуванням алгоритмів машинного навчання, а також підвищення ефективності роботи спеціалістів з аналізу даних.

Розділ «Передпроєктне обстеження предметної області» присвячений постановці завдання ДП, аналізу предметної області та існуючих рішень, моделювання основних бізнес-процесів програмного забезпечення.

У розділі «Розроблення вимог до програмного забезпечення» розглянуто варіанти використання ПЗ, розроблено функціональні та нефункціональні вимоги, виконано аналіз системних вимог та економічних показників ПЗ та сформовано постановку завдання на розробку.

У розділі «Конструювання та розроблення програмного забезпечення» розроблено архітектуру вебсервісу та прийнято ключові архітектурні рішення, обґрунтовано вибір засобів розробки. По завершенню даної підготовки, виконано конструювання програмного забезпечення та аналіз безпеки даних.

Розділ «Аналіз якості та тестування програмного забезпечення» містить результати аналізу якості розробленого ПЗ, опис процесів тестування та контрольного прикладу.

Розділ «Розгортання та супровід програмного забезпечення» охоплює процеси розгортання та супроводу вебсервісу.

Програмне забезпечення впроваджено на хмарній платформі вебхостингу PythonAnywhere.

КЛЮЧОВІ СЛОВА: ВЕБСЕРВІС, ДАНІ, ПОПЕРЕДНЯ ОБРОБКА, АВТОМАТИЗАЦІЯ, ЗВІТ, РЕКОМЕНДАЦІЇ, PYTHON, API, СХОВИЩЕ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 37 tables, 28 figures and 36 sources – in total 81 pages.

The diploma project is dedicated to the development of a web service for automated data analysis and report generation.

The purpose of the diploma project is to accelerate and simplify the process of data preparation for analysis, including the use of machine learning algorithms, and to increase the efficiency of data scientists.

The section "Pre-project research of the subject area" presents the problem statement, analysis of the subject area and existing solutions, and modelling of the main business processes of the software.

The section "Software requirements development" covers use cases of the service, functional and non-functional requirements, system requirements and economic performance analysis, and the final development task formulation.

The section "Software design and developmente" contains the architecture of the web service, justified development tools, implementation details, and data security analysis.

The section "Software quality analysis and testing" presents the quality assessment results, testing procedures, and a control example.

The section "Software deployment and maintenance" describes the processes of deploying and supporting the web service.

The software is deployed on the PythonAnywhere cloud web hosting platform.

KEYWORDS: WEB SERVICE, DATA, PREPROCESSING, AUTOMATION, REPORT, RECOMMENDATIONS, PYTHON, API, STORAGE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1118.045440.01.91	Технічне завдання	12	
3	А4	КПІ.ІП-1118.045440.02.81	Пояснювальна записка	81	
4	А4	КПІ.ІП-1118.045440.03.12	Текст програми	29	
5	А4	КПІ.ІП-1118.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-1118.045440.05.34	Керівництво користувача	6	
7	А3	КПІ.ІП-1118.045440.06.99	Графічний матеріал	3	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1118.045440.00.90		
Розроб.		Ляля І.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Ліхоузова Т.А.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11	
Н.контр.		Ліщук К.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

ВЕБСЕРВІС ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ДАНИХ ТА ГЕНЕРАЦІЇ ЗВІТІВ

Технічне завдання

КПІ.ІП-1118.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Іван ЛЯЛЯ

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебсервіс для автоматизованого аналізу даних і генерації звітів.

Галузь застосування: підготовка даних до аналізу.

Наведене технічне завдання поширюється на розробку програмного забезпечення «Вебсервіс для автоматизованого аналізу даних і генерації звітів», котра використовується для автоматизованої обробки структурованих даних, отримання корисних статистичних відомостей і надання звітів рекомендаційного характеру та призначена для аналітиків, бізнес-користувачів і дослідників, які працюють зі структурованими наборами даних у різних сферах, включаючи маркетинг, економіку, наукові дослідження та освіту.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки вебсервісу для автоматизованого аналізу даних та генерації звітів є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та полегшення процесу підготовки даних до аналізу, а також для формування узагальнених рекомендацій на основі виявлених закономірностей у даних.

Метою розробки є прискорення та спрощення процесу підготовки даних до аналізу, зменшення впливу людського фактора на етапі попередньої підготовки даних, підвищення якості виконання даного етапу, а також автоматизація формування базових аналітичних висновків і рекомендацій на основі виявлених закономірностей у даних шляхом реалізації високорівневого програмного інтерфейсу з підтримкою необхідного функціоналу.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Програмного інтерфейсу:

Вебсервіс повинен забезпечувати обробку наступних HTTP-запитів:

– GET / – отримання посилань на API-документацію сервісу;

– POST /datasets – завантаження файлу з набором даних на сервер;

– GET /datasets/{dataset_id} – перевірка наявності набору даних на сервері та отримання метаданих про нього;

– POST /datasets/{dataset_id}/preprocess – попередня обробка даних набору даних на сервері;

– GET /datasets/{dataset_id}/download – завантаження набору з сервера;

– GET /datasets/{dataset_id}/report – генерація та отримання рекомендаційного звіту;

– POST /datasets/full_pipeline – завантаження набору даних, його попередня обробка, генерація та отримання рекомендаційного звіту в одному запиті.

4.1.2 Для користувача:

4.1.2.1 Обмін даними:

– завантаження набору даних на сервер;

– вказання параметрів зчитування набору даних;

– отримання метаінформації про набір даних;

– завантаження набору даних з сервера на пристрій;

– вибір формату завантаження набору даних.

4.1.2.2 Попередня обробка даних:

– вказання параметрів попередньої обробки;
– збереження результату як копії;
– виконання попередньої обробки.

4.1.2.3 Аналітика та звітність:

– вказання параметрів аналізу;
– загальне дослідження;
– генерація рекомендацій;
– генерація PDF звіту;
– отримання звіту з рекомендаціями;
– виконання повного циклу операцій одним запитом.

4.1.3 Для системи:

4.1.3.1 Управління сховищем даних:

– збереження набору даних в сховищі;
– отримання набору даних зі сховища;
– періодична очистка сховища даних;
– захист наборів даних ключем доступу.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в сховищі даних. Забезпечити коректну роботу сервісу та структуровану відповідь на запити у випадку виникнення помилок. Забезпечити стабільну роботу при взаємодії з різними HTTP-клієнтами.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно бути доступним за мережевим протоколом HTTP/HTTPS, підтримуючи REST архітектуру взаємодії з клієнтом.

Мінімальна конфігурація технічних засобів:

– підключення до мережі Інтернет зі швидкістю від 32 Мбіт/с.

Рекомендована конфігурація технічних засобів:

– підключення до мережі Інтернет зі швидкістю від 100 Мбіт/с.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення має бути доступним для інтеграції з усіма клієнтами, що підтримують HTTP/HTTPS запити.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:

– файл повинен бути у одному з форматів: CSV, XLS, XLSX, JSON, DB (SQLite);

– файл повинен містити табличні дані з заголовками стовпців;

– максимальний розмір файлу – 100 МБ;

– кодування текстових даних: UTF-8 або Windows-1251.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:

– рекомендаційний звіт у форматі PDF;

– набір даних у одному з форматів: CSV, XLSX, JSON, PICKLE.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища PyCharm.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторія на GitHub.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– архітектура програмного забезпечення;
– моделі бізнес-процесів.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	21.02	
2.	Розробка технічного завдання	15.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації функціональних та нефункціональних вимог
4.	Проєктування структури програмного забезпечення, проєктування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	20.05	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебсервіс для автоматизованого аналізу даних та генерації звітів**

КПІ.ІП-1118.045440.02.81

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application programming interface, прикладний програмний інтерфейс.
IoT	–	Internet of Things, інтернет речей.
ETL	–	Extract, Transform, Load; витяг, перетворення, завантаження даних.
ELT	–	Extract, Load, Transform; витяг, завантаження, перетворення даних.
СУБД	–	Система управління базами даних.
ML	–	Machine Learning, машинне навчання.
BPMN	–	Business Process Model and Notation, модель та нотація бізнес-процесів.
JSON	–	JavaScript Object Notation, текстовий формат обміну даними між комп'ютерами.
CSV	–	Comma-Separated Values; значення, розділені комами.
PDF	–	Portable Document Format, міжплатформний формат електронних документів.
DoS	–	Denial of Service, відмова в обслуговуванні.
HTTPS	–	HyperText Transfer Protocol Secure; розширення до протоколу HTTP: дані передаються в зашифрованому вигляді.
HTTP	–	HyperText Transfer Protocol, протокол передачі гіпертексту.
REST	–	Representational State Transfer, передача репрезентативного стану.
ООП	–	Об’єктно-орієнтоване програмування.
IDE	–	Integrated Development Environment, інтегроване середовище розробки.
URL	–	Uniform Resource Locator, уніфікована адреса ресурсу.

ВСТУП

Актуальність тематики даного дипломного проєкту зумовлена стрімким зростанням обсягів даних, що збираються у різних сферах – від бізнесу та медицини до державного управління й освіти. Для отримання цінної інформації з цих даних необхідне проведення якісного аналізу, ефективність якого значною мірою залежить від рівня підготовки вхідних даних. На практиці етап попередньої обробки може займати до 80% загального часу аналізу даних, особливо коли він виконується вручну. Ручна підготовка даних є трудомісткою, вимагає технічної обізнаності та схильна до помилок, що знижує якість та достовірність результатів всього аналізу. Автоматизація даного процесу дозволяє прискорити отримання результатів, зменшити вплив людського фактора та забезпечити кращу узгодженість і повторюваність дій. Таким чином, автоматизована підготовка даних є важливою складовою сучасного аналізу даних.

Серед провідних тенденцій розв’язання проблеми варто виділити активне використання інструментів автоматизації та рекомендаційних систем у сфері Data Science. Сучасні дослідження зосереджені на підвищенні якості підготовки даних, адаптивності обробки до різних типів задач, а також на створенні простих у використанні інтерфейсів для аналітиків без глибокої технічної підготовки. Провідні компанії та наукові установи активно розвивають напрям AutoML (automated machine learning), одним з кроків якого є і автоматизована підготовка даних.

Станом на сьогодні існує низка інструментів, які частково вирішують окремі аспекти цієї задачі, однак більшість із них вимагає встановлення та налаштування середовища, що ускладнює використання для непідготовлених користувачів. Наявні рішення рідко поєднують гнучкість обробки з рекомендаційною підтримкою та простотою інтеграції в інші системи.

У рамках даної роботи буде розроблено вебзастосунок, що надає програмний інтерфейс для автоматизованої підготовки даних. Користувач матиме змогу надіслати набір даних, виконати його типові перетворення, а

також отримати рекомендації щодо покращення його якості та можливих подальших дій. Такий сервіс не потребуватиме встановлення програмного забезпечення та буде доступним через стандартний HTTP-інтерфейс, що значно спростить його використання.

Мета роботи – прискорити та спростити процес підготовки даних до аналізу, зменшити вплив людського фактора на даному етапі роботи, підвищити якість його виконання та надати користувачам без глибокого досвіду у сфері аналізу даних можливість отримати зрозумілі підказки щодо необхідних дій для покращення якості даних і підвищення точності подальшого аналізу.

Можливими сферами застосування розробки є бізнес-аналітика, освітні дослідження, маркетинговий аналіз, наукові дослідження, а також інші галузі, що використовують аналіз табличних даних для прийняття рішень.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

– аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;

– аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;

– аналіз та моделювання бізнес-процесів;

– розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;

– постановка завдання на розробку програмного забезпечення ДП;

– розроблення архітектури програмного забезпечення;

– розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;

– конструювання та розроблення програмного забезпечення;

– аналіз безпеки даних програмного забезпечення;

– аналіз якості та тестування програмного забезпечення;

– розгортання та супровід програмного забезпечення;

– створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

Дані – це сукупність фактів, чисел, слів, спостережень або іншої корисної інформації. Вони є основою для отримання знань, які допомагають організаціям приймати стратегічні рішення, підвищувати ефективність процесів та сприяти інноваціям. [1]

Найпоширеніші сфери застосування даних включають [1]:

– прогностичний аналіз;

– генеративний штучний інтелект;
– інновації в галузі охорони здоров’я;
– соціальні дослідження;
– кібербезпеку та управління ризиками;
– оперативну ефективність;
– покращення клієнтського досвіду;
– державні ініціативи та бізнес-аналітику.

Станом на березень 2025 року, щодня створюється понад 400 мільйонів терабайт даних, а річний обсяг досяг близько 147 зетабайт (1 зетабайт = 10^{12} гігабайт), що у 74 рази перевищує показники 2010 року [2]. На рисунку 1.1 можна переглянути графік обсягу створених даних щорічно. Такий стрімкий ріст обсягів даних відкриває нові можливості для аналітики, але водночас створює й виклики, зокрема – необхідність у більш ефективних та масштабованих підходах до обробки інформації.

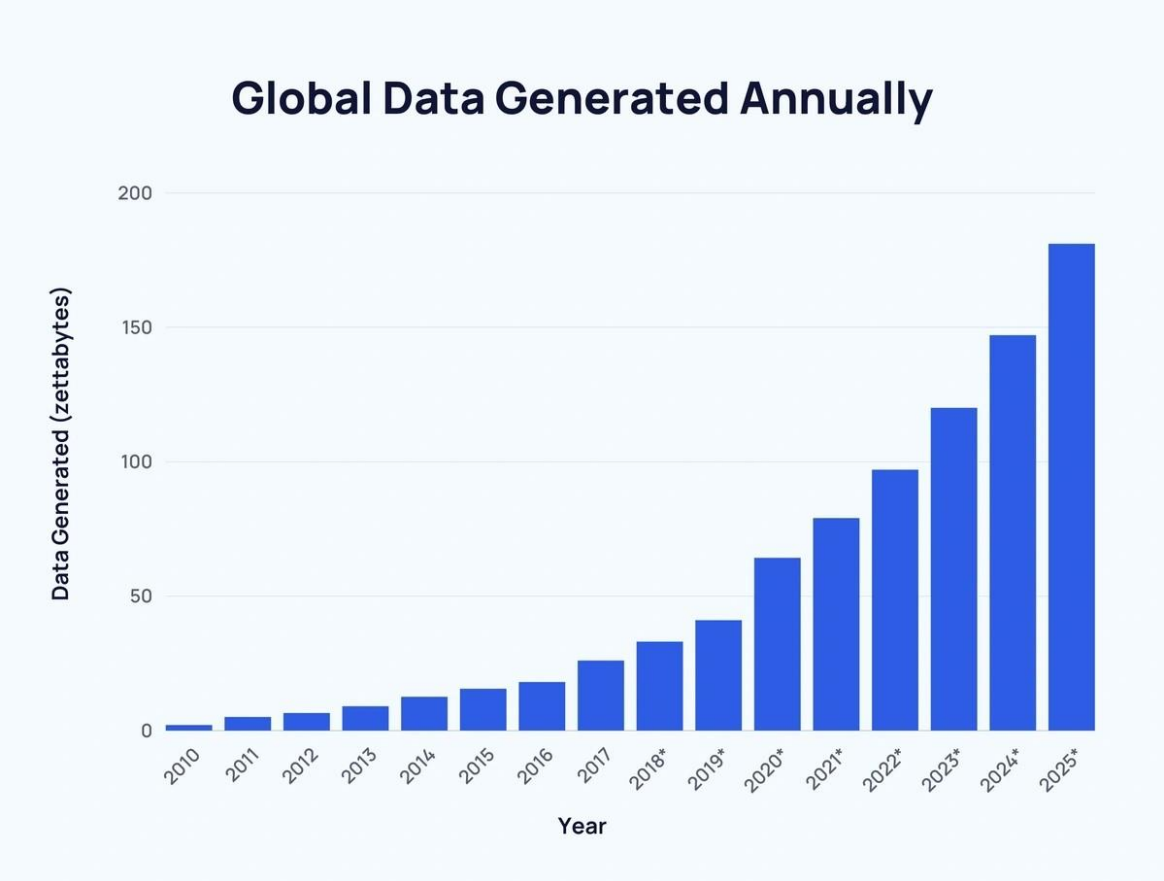


Рисунок 1.1 – Кількість даних, згенерованих за рік у світі [2]

Одним із таких підходів є автоматизований аналіз даних – процес збору, підготовки, аналізування або візуалізації даних за допомогою програмних інструментів і автоматизованих робочих процесів. Метою використання такого підходу є спрощення аналітики, підвищення точності, масштабованості та ефективності аналізу. Особливо корисним автоматизований аналіз даних може стати у випадку великого об’єму даних, при потребі отримувати корисну інформацію з даних у реальному часі або дуже швидко, за наявності повторюваних задач або в умовах обмежених ресурсів для проведення аналізу даних. [3]

На поточному етапі розвитку інформаційних технологій, автоматизація аналізу даних наразі має два основних напрями [3]:

1) автоматизація збору даних – передбачає:

– використання ботів або скриптів для систематичного отримання інформації з вебсайтів (вебскрейпінг);

– інтеграцію з API для доступу до структурованих джерел даних;

– отримання даних з IoT-пристроїв або датчиків, підключених до мережі.

2) автоматизація обробки даних (по суті, ключового етапу підготовки даних до аналізу), завдання якої можуть включати:

– валідацію даних – перевірку на помилки, дублікати, пропущені значення або неправильні формати;

– очистку даних – усунення виявлених помилок та невідповідностей;

– нормалізацію – приведення даних до єдиного формату для подальшого аналізу.

Для реалізації автоматизованого підходу застосовується низка інструментів [3]:

– ETL/ELT-рішення (наприклад, Zuar Runner), які автоматизують повний цикл: від збору та трансформації до передачі даних між системами.

– мови програмування (Python, R) та бібліотеки (наприклад, Pandas) забезпечують гнучку автоматизацію окремих етапів аналізу.

– BI-платформи (такі як Zuar Portal) дозволяють створювати аналітичні панелі, налаштовувати автоматичне оновлення та сповіщення.

– хмарні платформи (AWS, GCP) забезпечують побудову надійних, масштабованих робочих процесів із використанням оркестрації та планування завдань.

Незважаючи на значний прогрес, в існуючих підходах до аналізу даних все ще є певні обмеження. До ключових недоліків можна віднести високу складність налаштування інструментів, потребу у технічній експертизі, відсутність універсальних рішень для широкого кола задач та недостатню гнучкість деяких платформ у роботі зі специфічними даними.

Для покращення ситуації у сфері автоматизованого аналізу даних важливо спростити інтерфейси для користувачів без технічної підготовки, поширювати low-code рішення, підтримувати open-source інструменти та сприяти розвитку цифрової грамотності. Крім того, перспективним напрямом є впровадження інтелектуальних систем, що здатні самостійно виявляти проблеми або пропонувати варіанти аналізу.

У рамках даного дипломного проєкту було обрано напрям автоматизації аналізу даних шляхом спрощення процесу їх попередньої обробки та прискорення отримання ключових відомостей з них. Розроблюваний сервіс надасть можливість в декілька кроків завантажити дані у різних форматах, вказати параметри попередньої обробки, цільову задачу аналізу та одразу отримати структурований звіт з численними статистичними характеристиками, візуалізаціями та рекомендаціями щодо подальшого виконання задач з машинного навчання. Такий підхід дозволить користувачеві отримати корисну інформацію, зекономити час, при цьому без потреби у глибоких технічних знаннях.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації вебсервісу для автоматизованого аналізу даних та генерації звітів. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

Однією з провідних платформ для підготовки, обробки та очищення даних є Talend. Вона має зручний вебінтерфейс з функцією «point-and-click», що робить її доступною навіть для користувачів без досвіду програмування (рисунок 1.2). Разом із цим платформа підтримує створення кастомних сценаріїв ETL з можливістю роботи з кодом. Talend дозволяє створювати та зберігати правила трансформації даних, які можна повторно використовувати в командній роботі.

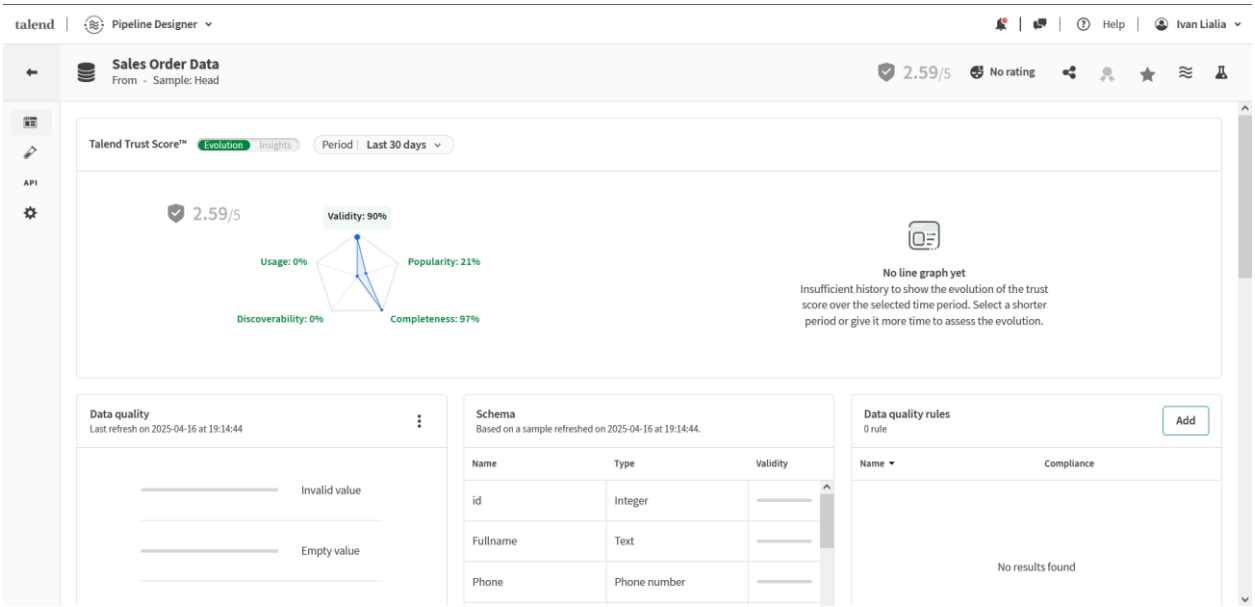


Рисунок 1.2 – Інтерфейс Talend

Перевагами даної платформи є наявність потужних інструментів для обробки даних, підтримка інтеграції з різними джерелами даних та можливість запуску пайплайнів будь-де. Недоліки – обмежена підтримка задач машинного навчання, високе споживання пам’яті та наявність багів у роботі платформи [4].

Alteryx APA – це потужна платформа для аналітики, яка поєднує можливості обробки даних, побудови моделей та візуалізації. Пропонує понад 100 інструментів для трансформації даних, зокрема fuzzy matching, профілювання, заміну значень тощо. Alteryx Designer є ключовим інструментом цієї платформи, який дозволяє будувати візуальні робочі процеси для обробки даних без потреби у програмуванні. Саме цей інструмент використаємо для порівняння, а його інтерфейс наведемо на рисунку 1.3.

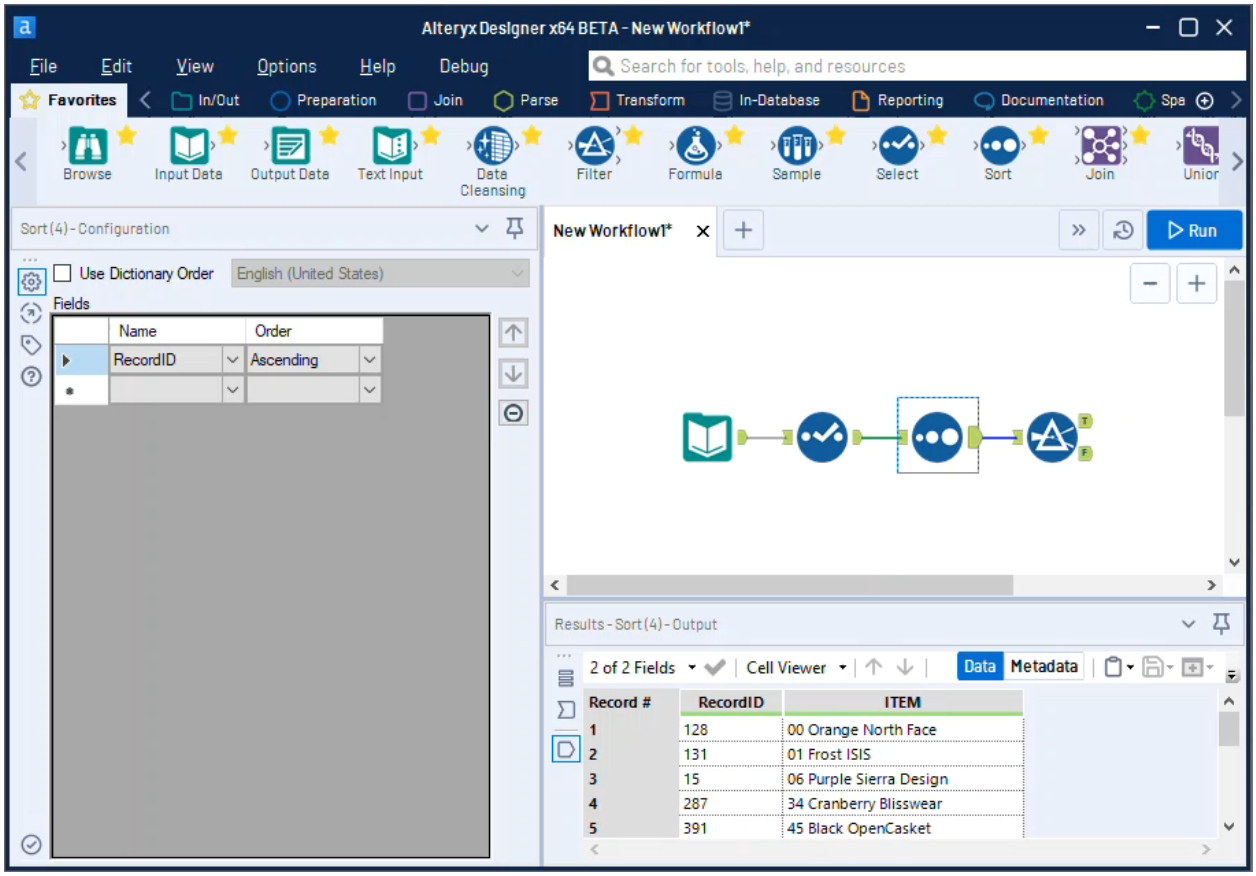


Рисунок 1.3 – Інтерфейс Alteryx Designer [5]

Перевагами даної платформи є підтримка великої кількості джерел даних – від електронних таблиць до вебресурсів та соцмереж, висока швидкість обробки, можливість створення моделей та їх повторного використання. Серед недоліків – застарілий інтерфейс, що може ускладнювати роботу та користувацький досвід, та висока вартість ліцензії [4].

Altair Monarch спеціалізується на перетворенні складних, неструктурованих джерел, таких як PDF-документи та текстові звіти, у структуровані таблиці. Інструмент дозволяє налаштовувати правила обробки даних та експортувати результати безпосередньо до СУБД. Має особливу популярність у фінансовій та медичній галузях [4]. Наведемо його інтерфейс на рисунку 1.4.

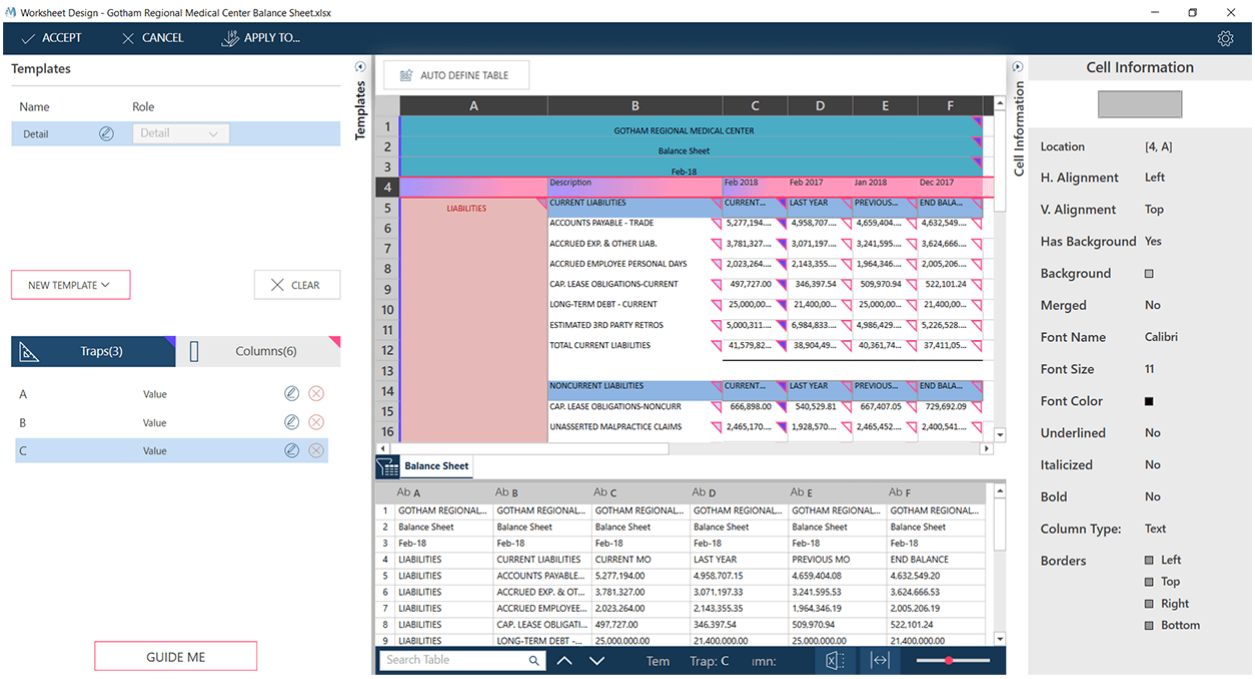


Рисунок 1.4 – Інтерфейс Altair Monarch [6]

Перевагами інструмента є підтримка неструктурованих джерел, широкі можливості інтеграції та експорту, підтримка складної логіки обробки. Серед недоліків – висока складність для новачків, падіння продуктивності зі зростанням обсягів даних. [4]

Для порівняння проєкту з аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал та особливості	Вебсервіс для автоматизованого аналізу даних та генерації звітів	Talend	Alteryx APA	Altair Monarch

Продовження таблиці 1.1

Ліцензування	Безкоштовне користування	14-денна пробна версія	30-денна пробна версія	30-денна пробна версія
Простота для новачків	Висока	Середня	Середня	Низька
Підтримка ML-моделей	Немає (можлива інтеграція)	Обмежено	Повноцінно	Немає
Генерація звітів	Автоматично	Вручну	Вручну	Відсутня
Рекомендації у звітах	+	-	+	-
Тип інтерфейсу	Веб	Веб	Десктоп	Десктоп
Необхідність реєстрації	Ні	Так	Так	Так
Системні вимоги	Низькі	Середні	Високі	Високі
Обробка великих обсягів даних	До 100 МБ на файл	Добре масштабується	Добре масштабується	Обмежено

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

У рамках даної розробки не передбачено використання складних математичних або обчислювальних алгоритмів, що потребують глибокого теоретичного аналізу або оптимізації. Основні задачі сервісу вимагають базової статистичної агрегації або простих математичних операцій, що планується реалізовувати з використанням готових бібліотечних рішень.

Таким чином, розробка або модифікація існуючих алгоритмів в межах дипломної роботи не планується.

Як і всі вебзастосунки, сервіс для автоматизованого аналізу даних та генерації звітів матиме клієнт-серверну архітектуру, що передбачає розділення програмного коду на два компоненти – скрипти клієнтської та серверної сторін [7]. У рамках серверної частини можуть застосовуватись різні архітектурні стилі. Розглянемо основні з них [8].

Моноліт – передбачає розробку всіх компонентів програмного забезпечення як єдиного цілого. Всі модулі взаємопов’язані та розгортаються разом. Такий підхід спрощує початкову реалізацію і добре підходить для невеликих проєктів із обмеженими ресурсами. Основними недоліками є складність у масштабуванні та супроводі з часом. Також існує ризик повної зупинки системи у разі збою одного з компонентів.

Мікросервіси – застосунок розбивається на набір незалежних сервісів, кожен із яких реалізує окрему бізнес-логіку та має власну базу даних. Комунікація між сервісами відбувається через REST API. Такий підхід забезпечує легкість масштабування, незалежність змін та підтримку безперервного розгортання. Проте реалізація та підтримка такої архітектури вимагає складнішої інфраструктури та координації.

Безсерверна (Serverless) – базується на використанні хмарних функцій, інфраструктура яких повністю керується провайдером (наприклад, AWS Lambda). Оплата відбувається лише за час виконання коду, що знижує витрати на сервіс. Даний стиль зручний для обробки подій, обмежених у часі, або нерегулярного завантаження. З недоліків – залежність від провайдера функцій, обмеження на час виконання та складність в налагодженні додатка.

Для порівняння архітектурних стилів у рамках серверної частини клієнт-серверної архітектури можна скористатися таблицею 1.2.

Таблиця 1.2 – Порівняння архітектурних стилів

Критерій	Моноліт	Мікросервіси	Serverless
Початок розробки	Простий	Складніший	Простий
Масштабованість	Обмежена	Висока	Автоматична
Складність підтримки	Зростає з часом	Висока	Помірна
Продуктивність	Висока при малому обсязі	Висока, вимагає оптимізації	Залежить від навантаження
Вартість інфраструктури	Низька на старті	Порівняно висока	Плата за використання
Гнучкість	Низька	Висока	Висока
Залежність від провайдера	Немає	Часткова	Висока

Іншим важливим технічним рішенням є вибір платформи для розгортання вебсервісу. Можна виділити два основних типи платформ [9]:

Вебхостинг – це розміщення вебзастосунку на одному фізичному сервері в дата-центрі, має кілька видів:

– спільний хостинг (Shared Hosting): кілька сайтів працюють на одному сервері. Дешево, але знижена продуктивність та безпека.

– віртуальний приватний сервер (VPS): окремий віртуальний сервер на фізичному пристрої. Краща ізоляція та швидкість.

– виділений сервер (Dedicated Hosting): весь сервер у вашому розпорядженні. Максимальна потужність, але й висока вартість і складність адміністрування.

Хмарний хостинг використовує мережу взаємопов’язаних серверів замість одного фізичного. Також поділяється на різні типи:

– публічна хмара (Public cloud).

– приватна хмара (Private cloud).

– гібридна хмара (Hybrid cloud).

– мультихмара (Multi-cloud).

Для порівняння типів платформ можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняльний аналіз вебхостингу та хмарного хостингу

Критерій	Вебхостинг	Хмарний хостинг
Продуктивність	Обмежена продуктивністю одного сервера	Висока; використовує балансування навантаження
Безпека	Зазвичай нижча; ресурси розподіляються між користувачами	Висока; ізоляція, резервне копіювання, автоматичне оновлення
Масштабованість	Обмежена; для збільшення потужності потрібен перехід на інший тариф	Висока; масштабування може бути автоматичним або ручним
Надійність	Залежність від одного фізичного сервера; у разі збою — простій	Висока; трафік перенаправляється на інші сервери у разі збоїв
Розподіл ресурсів	Фіксований; можливі збої через перевантаження	Динамічний; ресурси виділяються за потребою
Архітектура	Прив’язана до одного дата-центра або сервера	Глобальна хмарна інфраструктура з можливістю розміщення у різних регіонах
Вартість	Зазвичай нижча; фіксована ціна	Гнучка; сплачуєш за використані ресурси (може бути вигідніше при нестабільному трафіку)

Для реалізації вебсервісу буде використано монолітну архітектуру, що дозволяє швидше розробити програмний продукт, спростити його підтримку та зменшити витрати на інфраструктуру. Розгортання сервісу

відбуватиметься на основі хмарного хостингу, що забезпечить високу масштабованість, надійність та доступність без потреби в ручному керуванні інфраструктурою.

1.4 Аналіз та моделювання бізнес-процесів

Для моделювання бізнес-процесів використаємо BPMN нотацію [10]. На рисунку 1.5 наведемо модель ключового бізнес-процесу вебсервісу для автоматизованого аналізу даних та генерації звітів – отримання рекомендаційного звіту.

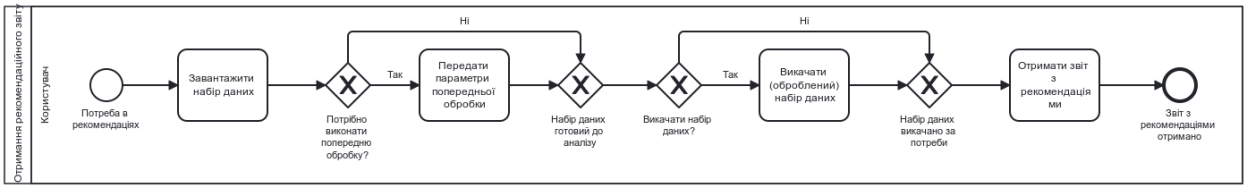


Рисунок 1.5 – Модель бізнес-процесу «Отримання рекомендаційного звіту»

Опис моделі бізнес-процесу отримання рекомендаційного звіту:

– користувач завантажує набір даних на сервер;

– якщо користувач бажає виконати попередню обробку даних – передає параметри обробки на сервер, інакше – переходить до наступного кроку;

– користувач може за бажанням викачати з сервера готовий до аналізу набір даних;

– користувач надсилає запит на отримання рекомендацій з указанням ML-задачі, яка планується виконуватись над набором даних та отримує рекомендаційний звіт.

Ще одним важливим бізнес-процесом є автоматизована попередня обробка даних. Наведемо модель процесу на рисунку 1.6.

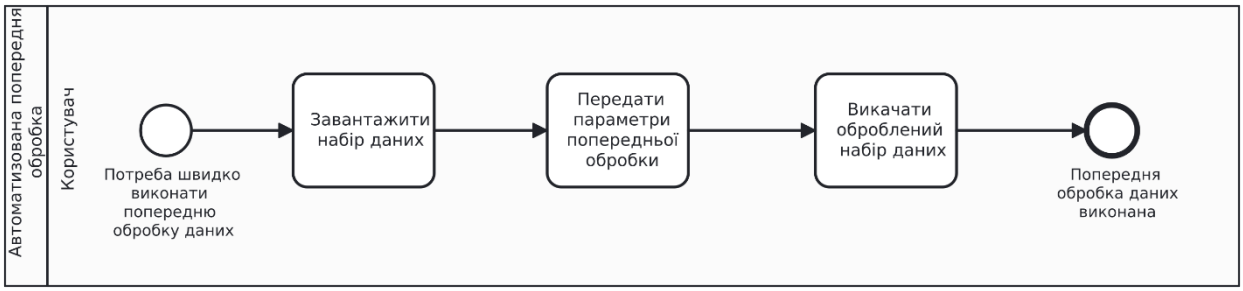


Рисунок 1.6 – Модель бізнес-процесу «Автоматизована попередня обробка»

Опис моделі бізнес-процесу автоматизованої попередньої обробки:

– користувач завантажує набір даних на сервер;

– користувач передає параметри попередньої обробки на сервер;

– користувач викачує з сервера автоматично оброблений набір даних.

Висновки до розділу

У ході передпроєктного обстеження предметної області було сформульовано основні завдання дипломного проєктування, деталізовано ключові поняття, розглянуто основні напрями та інструменти автоматизації аналізу даних, визначено проблеми в існуючих підходах та запропоновано шляхи їх вирішення. В результаті було окреслено напрям розробки.

Наступним кроком проведено порівняльний аналіз наявних програмних (Talend, Alteryx APA, Altair Monarch) та технічних рішень, визначено їх переваги та недоліки. На основі аналізу зроблено висновок про доцільність розробки власного рішення, обрано архітектуру та платформу для розгортання.

Останнім етапом у розділі став аналіз бізнес-процесів розроблюваного програмного забезпечення та їх моделювання з використанням BPMN.

Таким чином, визначено концепцію створення програмного забезпечення, що спрощує попередню обробку даних та отримання з них корисних відомостей, зосереджуючись на простоті використання для кінцевого користувача.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Діаграма варіантів використання з ключовими акторами та основними функціями показана на рисунку 2.1.

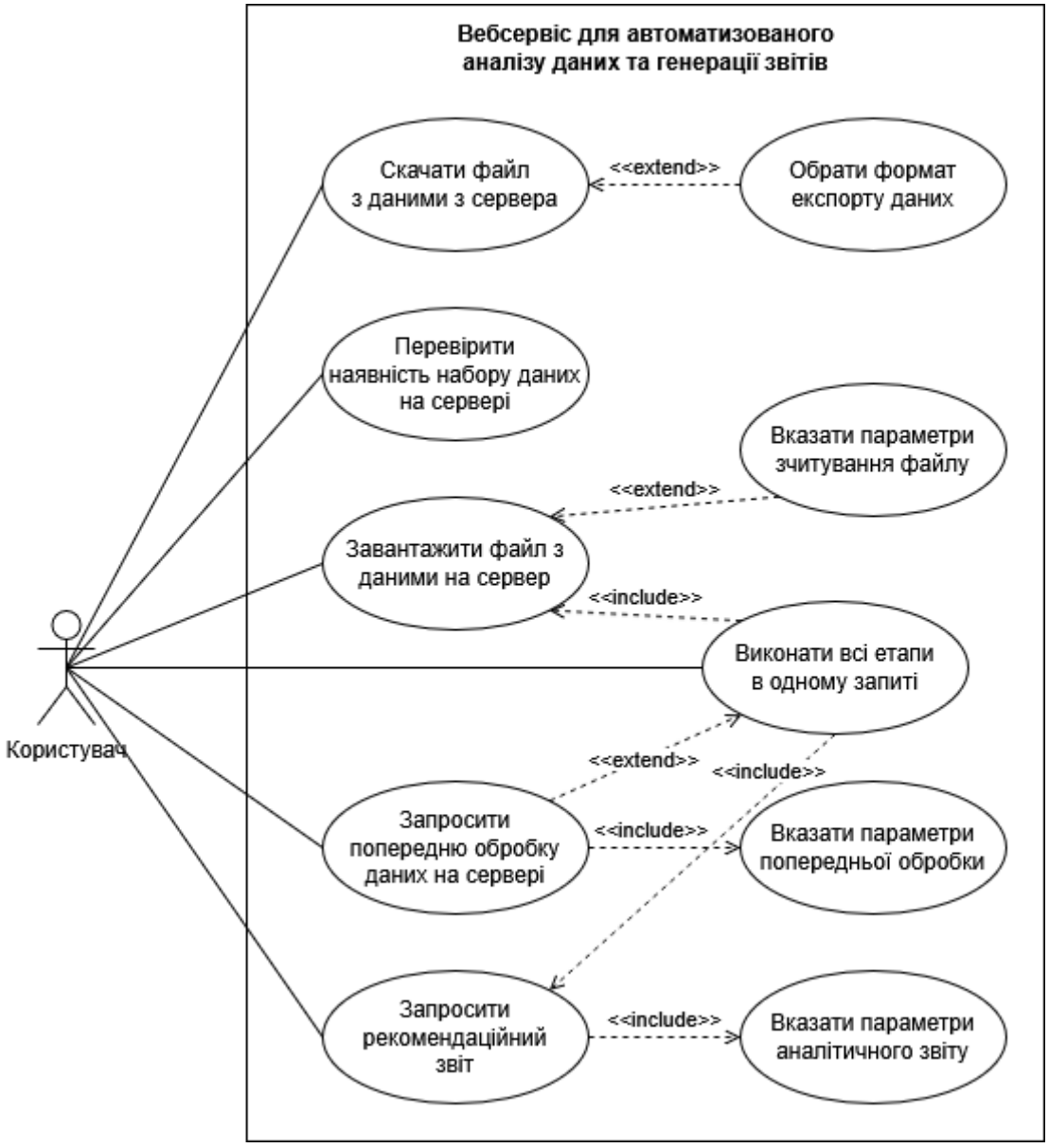


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1 – 2.6 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Завантажити файл з даними на сервер
Use case ID	UC-01
Goals	Надіслати локальний файл користувача на сервер для подальшої обробки
Actors	Користувач
Trigger	Користувач бажає розпочати аналіз набору даних
Pre-conditions	-
Flow of Events	Користувач виконує запит до сервісу, в якому додає файл з даними та, за потреби, явно вказує параметри зчитування файлу (наприклад, роздільник, кодування). Система приймає файл, зчитує набір даних, зберігає його в сховищі та повертає повідомлення про успішне завантаження, що містить також базову інформацію про набір даних, його ідентифікатор та ключ доступу, пропозицію щодо наступного кроку роботи.
Extension	У випадку передачі некоректних параметрів зчитування або файлу, а також у разі виникнення інших проблем при обробці запиту на сервері, користувачеві повертається інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Набір даних збережений на сервері для подальшої обробки

Таблиця 2.2 – Варіант використання UC-02

Use case name	Перевірити наявність набору даних на сервері
Use case ID	UC-02
Goals	Перевірити, чи є завантажений раніше файл на сервері
Actors	Користувач
Trigger	Користувач хоче переконатися, що файл завантажено
Pre-conditions	Попередньо файл мав бути завантажений на сервер
Flow of Events	Користувач виконує запит до сервісу, що включає ідентифікатор набору даних та ключ доступу до набору даних у відповідному заголовку. Система шукає набір даних у сховищі та валідує ключ доступу, повертає повідомлення про наявність, а також базову інформацію про набір даних, його ідентифікатор, пропозицію щодо наступного кроку роботи.

Продовження таблиці 2.2

Extension	Якщо набір даних не знайдено або наданий ключ доступу невірний, користувачеві повертається відповідне інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Користувач отримав підтвердження наявності набору даних на сервері або повідомлення про його відсутність

Таблиця 2.3 – Варіант використання UC-03

Use case name	Скачати файл з даними з сервера
Use case ID	UC-03
Goals	Отримати копію завантаженого (та, можливо, обробленого) файлу назад на локальний пристрій
Actors	Користувач
Trigger	Користувач бажає зберегти набір даних локально
Pre-conditions	Набір даних присутній у сховищі на сервері
Flow of Events	Користувач виконує запит до сервісу, що включає ідентифікатор набору даних, формат експорту за потреби та ключ доступу до набору даних у відповідному заголовку. Система шукає набір даних у сховищі та валідує ключ доступу, після чого надсилає його користувачеві як бінарний файл в обраному форматі або CSV за замовчуванням.
Extension	Якщо набір даних не знайдено або наданий ключ доступу невірний, користувачеві повертається відповідне інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Файл з набором даних завантажений на локальний пристрій користувача

Таблиця 2.4 – Варіант використання UC-04

Use case name	Запросити попередню обробку даних на сервері
Use case ID	UC-04
Goals	Провести автоматизовану обробку даних
Actors	Користувач
Trigger	Користувач бажає виконати певну очистку або трансформацію даних на сервері

Продовження таблиці 2.4

Pre-conditions	Набір даних присутній у сховищі на сервері
Flow of Events	Користувач виконує запит до сервісу, що включає ідентифікатор набору даних та ключ доступу до набору даних у відповідному заголовку, а також необхідні параметри обробки з доступних. Система шукає набір даних у сховищі та валідує ключ доступу, виконує його попередню обробку відповідно до переданих параметрів та зберігає зміни у сховищі. Після цього, повертається повідомлення про успішність виконання даного етапу, що містить також оновлену базову інформацію про набір даних, його ідентифікатор та пропозицію щодо наступного кроку роботи.
Extension	Якщо набір даних не знайдено або наданий ключ доступу невірний, а також у випадку передачі некоректних параметрів попередньої обробки або виникнення інших проблем на сервері, користувачеві повертається відповідне інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Дані на сервері зберігаються у попередньо обробленому вигляді

Таблиця 2.5 – Варіант використання UC-05

Use case name	Запросити рекомендаційний звіт
Use case ID	UC-05
Goals	Отримати рекомендації щодо подальшого аналізу
Actors	Користувач
Trigger	Користувач бажає отримати інструкції та рекомендації для аналізу даних
Pre-conditions	Набір даних присутній та, за потреби, попередньо оброблений у сховищі на сервері
Flow of Events	Користувач виконує запит до сервісу, що включає ідентифікатор набору даних та ключ доступу до набору даних у відповідному заголовку, а також параметри аналізу даних та формування звіту. Система шукає набір даних у сховищі та валідує ключ доступу, виконує його аналіз та формує рекомендаційний звіт у форматі PDF і надсилає його користувачеві.

Продовження таблиці 2.5

Extension	Якщо набір даних не знайдено або наданий ключ доступу невірний, а також у випадку передачі некоректних параметрів подальшого аналізу або виникнення інших проблем на сервері, користувачеві повертається відповідне інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Користувач отримав звіт з рекомендаціями

Таблиця 2.6 – Варіант використання UC-06

Use case name	Виконати всі етапи в одному запиті
Use case ID	UC-06
Goals	Передати набір даних, автоматизовано виконати його обробку та отримати рекомендаційний звіт за один запит
Actors	Користувач
Trigger	Користувач бажає швидко отримати рекомендації без виконання окремих етапів, або не має змоги зберігати набір даних між запитами (наприклад, через обмеження місця в сховищі)
Pre-conditions	-
Flow of Events	Користувач виконує запит до сервісу, в якому додає файл з даними та, за потреби, явно вказує параметри зчитування файлу, попередньої обробки даних, які містяться в ньому, та рекомендаційного звіту. Система приймає файл, зчитує набір даних, виконує його попередню обробку та аналіз, формує рекомендаційний звіт у форматі PDF та надсилає його користувачеві.
Extension	У випадку передачі некоректних параметрів чи їх значень, або файлу з даними, а також у разі виникнення інших проблем при обробці запиту на сервері, користувачеві повертається інформативне повідомлення про помилку у JSON форматі.
Post-Condition	Користувач отримав звіт з рекомендаціями

2.2 Розроблення функціональних вимог

Програмне забезпечення умовно розділене на модулі. Кожен модуль має певний набір функцій. В таблиці 2.7 наведено загальну модель вимог, а в таблиці 2.8 – опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог представлено на рисунку 2.2.

Таблиця 2.7 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Управління сховищем даних	-	Високий	Низький
1.1	Збереження набору даних в сховищі	FR-1	Високий	Низький
1.2	Отримання набору даних зі сховища	FR-2	Високий	Низький
1.3	Періодична очистка сховища даних	FR-3	Середній	Середній
1.4	Захист наборів даних ключем доступу	FR-4	Середній	Середній
2	Обмін даними	-	Високий	Середній
2.1	Завантаження набору даних на сервер	FR-5	Високий	Середній
2.2	Вказання параметрів зчитування набору даних	FR-6	Низький	Середній
2.3	Отримання метаінформації про набір даних	FR-7	Низький	Низький
2.4	Завантаження набору даних з сервера на пристрій	FR-8	Середній	Низький
2.5	Вибір формату завантаження набору даних	FR-9	Низький	Низький
3	Попередня обробка даних	-	Високий	Середній

Продовження таблиці 2.7

3.1	Вказання параметрів попередньої обробки	FR-10	Високий	Високий
3.2	Збереження результату як копії	FR-11	Середній	Низький
3.3	Виконання попередньої обробки	FR-12	Високий	Високий
4	Аналітика та звітність	-	Високий	Середній
4.1	Вказання параметрів аналізу	FR-13	Середній	Середній
4.2	Загальне дослідження	FR-14	Високий	Низький
4.3	Генерація рекомендацій	FR-15	Середній	Високий
4.4	Генерація PDF звіту	FR-16	Високий	Середній
4.5	Отримання звіту з рекомендаціями	FR-17	Високий	Високий
4.6	Виконання повного циклу операцій одним запитом	FR-18	Середній	Високий

Таблиця 2.8 – Опис функціональних вимог

ID	Опис
FR-1	Збереження набору даних в сховищі. Система автоматично зберігає набори даних у сховище для подальшої обробки або аналізу.
FR-2	Отримання набору даних зі сховища. Система надає доступ до раніше збережених наборів даних у сховищі.
FR-3	Періодична очистка сховища даних. Система автоматично очищає тимчасове сховище від застарілих наборів даних.
FR-4	Захист наборів даних ключем доступу. Система обмежує доступ до раніше збережених наборів даних через унікальний ключ доступу.

Продовження таблиці 2.8

FR-5	Завантаження набору даних на сервер. Користувач може завантажити табличний файл на сервер.
FR-6	Вказання параметрів зчитування набору даних. Користувач може вказати параметри зчитування файлу з даними при його завантаженні на сервер.
FR-7	Отримання метаінформації про набір даних. Користувач може отримати метаінформацію про набір даних, що знаходиться у сховищі.
FR-8	Завантаження набору даних з сервера на пристрій. Користувач може завантажити набір даних зі сховища.
FR-9	Вибір формату завантаження набору даних. Користувач може обрати формат, в якому завантажити дані з сервера.
FR-10	Вказання параметрів попередньої обробки. Користувач може вказати параметри попередньої обробки даних у запиті на її виконання.
FR-11	Збереження результату як копії. Користувач може зберігати результат попередньої обробки як новий набір даних у сховищі без перезапису оригінального.
FR-12	Виконання попередньої обробки. Система виконує попередню обробку даних відповідно до вказаних параметрів.
FR-13	Вказання параметрів аналізу. Користувач може вказати параметри аналізу даних для отримання більш точних рекомендацій та кастомізації вигляду звіту.
FR-14	Загальне дослідження. Система виконує базовий аналіз даних, включаючи описову статистику, пошук пропущених значень тощо.

Продовження таблиці 2.8

FR-15	Генерація рекомендацій. Система генерує рекомендації на основі параметрів аналізу.
FR-16	Генерація PDF звіту. Система формує PDF звіт за результатами аналізу даних.
FR-17	Отримання звіту з рекомендаціями. Користувач може отримати рекомендаційний аналітичний звіт.
FR-18	Виконання повного циклу операцій одним запитом. Система дозволяє виконати повний цикл зчитування, попередньої обробки та аналізу набору даних в одному запиті.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18
UC-01	+			+	+	+	+											
UC-02		+		+			+											
UC-03		+		+				+	+									
UC-04	+	+		+			+			+	+	+						
UC-05		+		+									+	+	+	+	+	
UC-06					+	+				+		+	+	+	+	+	+	+

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

1) Доступність:

– рішення є працезданим не менше ніж 98% часу в місяць.

– допустимий час відновлення після відмови не перевищує 4 години.

– відстеження працездатності здійснюється через базову перевірку відповіді API на запити.

2) Продуктивність:

– API підтримує обробку до 10 одночасних запитів без помітного погіршення продуктивності.

– час відповіді на запит API не перевищує 3 секунди для невеликих операцій.

– генерація звіту на основі набору даних розміром до 100 МБ триває не довше 5 хвилин.

3) Обмеження:

– рішення реалізується мовою програмування Python з використанням фреймворку Flask.

– дані у сховищі зберігаються в форматі pickle у файловій системі сервера.

– мінімальна конфігурація сервера є наступною: 2 CPU, 4 GB RAM, 5 GB SSD.

4) Безпека:

– для запобігання атакам типу DoS через великі файли або некоректні формати передбачена валідація вхідних даних.

– захист потенційно конфіденційної інформації у наборах забезпечується шифруванням під час передачі (через протокол HTTPS) та обмеженням доступу до даних у сховищі.

– розмежування доступу до наборів даних реалізується шляхом використання унікального ключа доступу, який необхідний для їх отримання.

5) Вимоги до зберігання даних:

– гарантується збереження завантажених наборів даних у сховищі протягом 24 годин.

– автоматичне видалення даних здійснюється після закінчення терміну зберігання.

6) Масштабованість:

– архітектура рішення передбачає можливість вертикального масштабування.

7) Конфігурованість:

– основні параметри роботи сервісу можна налаштувати через конфігураційний файл.

– зміна конфігураційного файлу не повинна потребувати змін у програмному коді.

8) Зручність використання:

– максимальна глибина вкладеності у структурах URL не перевищує 3 рівні, що полегшує навігацію та інтеграцію.

– об’єкти у JSON-відповідях мають не більше 3 рівнів вкладеності для збереження читабельності та спрощення обробки на клієнтській стороні.

– усі повідомлення про помилки мають уніфікований формат (назва, код та опис помилки), що спрощує їх автоматичну обробку клієнтами.

– API є самодокументованим у специфікації OpenAPI.

– іменування ендпоінтів та параметрів відповідає суті виконуваних операцій, що забезпечує зрозумілість.

– API реалізовано відповідно до REST-архітектури.

2.4 Аналіз системних вимог

Для взаємодії з вебсервісом достатньо наявності:

– HTTP-клієнта (наприклад, curl, Postman або будь-яка програмна бібліотека з підтримкою HTTP-запитів);

– підключення до мережі Інтернет.

За мінімальну потрібну швидкість підключення вважатимемо таку, яка забезпечить передачу файлу з даними максимально допустимого розміру (100 МБ) за 25 секунд:

$$\frac{100\ \text{МБ}}{25\ \text{с}} * \frac{8\ \text{біт}}{\text{Б}} = 32\frac{\text{Мбіт}}{c}.$$

Рекомендованою конфігурацією вважатимемо ту, на якій виконуватимемо розробку $\sim 100\frac{\text{Мбіт}}{c}$ (рисунок 2.3).

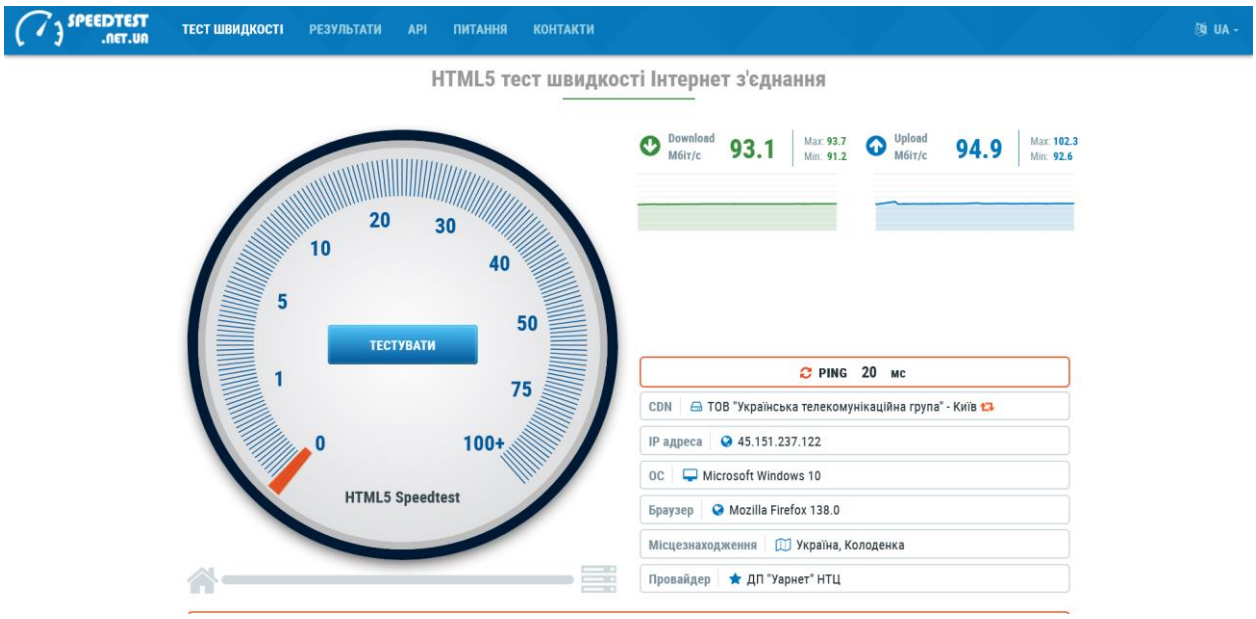


Рисунок 2.3 – Тестування швидкості Інтернет з’єднання на пристрої, де буде виконуватись розробка [11]

2.5 Аналіз економічних показників програмного забезпечення

Для аналізу економічних показників вебсервісу для автоматизованого аналізу даних та генерації звітів буде використано метод підрахунку точок варіантів використання (UCP) [12].

В таблиці 2.9 наведемо результати оцінки складності для кожного з варіантів використання.

Таблиця 2.9 – Складність варіантів використання

Use case ID	Use case name	Кількість транзакцій	Складність
UC-01	Завантажити файл з даними на сервер	5	Середній
UC-02	Перевірити наявність набору даних на сервері	3	Простий
UC-03	Скачати файл з даними з сервера	4	Середній
UC-04	Запросити попередню обробку даних на сервері	7	Середній
UC-05	Запросити рекомендаційний звіт	7	Середній
UC-06	Виконати всі етапи в одному запиті	10	Складний

На основі зробленої оцінки обчислимо нескориговану оцінку варіантів використання (UUCW) в таблиці 2.10.

Таблиця 2.10 – Нескоригована оцінка варіантів використання (UUCW)

Складність	Ваговий коефіцієнт	Кількість варіантів використання	Програмний продукт
Простий	5	1	5
Середній	10	4	40
Складний	15	1	15
Всього			60

Далі розрахуємо нескориговану оцінку акторів (UAW). Розроблювана система має лише одного актора – користувача, що взаємодіє через API, і тому згідно класифікації вважається середнім за складністю. Розрахунок наведемо в таблиці 2.11.

Таблиця 2.11 – Нескоригована оцінка акторів (UAW)

Складність	Ваговий коефіцієнт	Кількість акторів	Програмний продукт
Простий	1	0	0
Середній	2	1	2
Складний	3	0	0
Всього			2

Обчислимо нескориговані точки варіантів використання (UUCP):

$$UUCP = UUCW + UAW = 60 + 2 = 62\,.$$

Наступним кроком є розрахунок фактора технічної складності. Для цього потрібно оцінити вплив на розробку програмного продукту кожного з 13 технічних факторів від 0 (неважливий) до 5 (критично важливий) та обчислити зважену суму наведених факторів. Наведемо відповідні розрахунки в таблиці 2.12.

Таблиця 2.12 – Зважена сума технічних факторів (TFactor)

Фактор	Ваговий коефіцієнт	Оцінка	Вплив
Розподілена система	2	3	6
Вимоги до швидкодії	2	3	6
Ефективність для користувача	1	2	2
Складність внутрішньої обробки	1	3	3
Повторне використання коду	1	3	3
Простота встановлення	0.5	2	1
Простота використання	0.5	4	2
Портативність	2	4	8
Простота зміни	1	3	3
Паралельне використання	1	2	2
Безпека	1	3	3
Доступ третіх сторін	1	3	3
Потреба в навчанні	1	3	3
Всього (TFactor)			45

На основі розрахованого впливу факторів проведемо розрахунок фактора технічної складності (TCF):

$$TCF = 0.6 + (0.01 * TFactor) = 0.6 + (0.01 * 45) = 1.05 .$$

Далі потрібно оцінити складність середовища. Щоб це зробити, спочатку потрібно оцінити вплив кожного з 8 факторів середовища від 0 до 5 та обчислити їх зважену суму (EFactor). Відповідні розрахунки наведемо у таблиці 2.13.

Таблиця 2.13 – Зважена сума факторів середовища (EFactor)

Фактор	Ваговий коефіцієнт	Оцінка	Вплив
Досвід з процесом розробки	1.5	3	4.5
Досвід у предметній області	0.5	4	2
Досвід з ООП	1	4	4
Компетентність аналітика	0.5	3	1.5
Мотивація	1	3	3
Стабільність вимог	2	3	6

Продовження таблиці 2.13

Часткова зайнятість розробників	-1	0	0
Складність мови програмування	-1	0	0
Всього (EFactor)			21

На основі розрахованого впливу проведемо розрахунок фактора середовища (EF):

$$EF = 1.4 + (-0.03 * EFactor) = 1.4 + (-0.03 * 21) = 0.77 .$$

Щоб обчислити точки варіантів використання, потрібно скоригувати UUCP обчисленими значеннями TCF та EF:

$$UCP = UUCP * TCF * EF = 62 * 1.05 * 0.77 \approx 50 .$$

З урахуванням того, що система є вебсервісом із мінімальним інтерфейсом користувача, а основна функціональність полягає в обробці даних та генерації звітів, більшість операцій є автоматизованими. Крім того, використання сучасних фреймворків та бібліотек дозволяє суттєво скоротити витрати часу на реалізацію окремих компонентів. Таким чином, доцільно прийняти 1 людино-день на одну точку варіантів використання (UCP).

За обчисленим значенням UCP = 50, загальна трудомісткість реалізації програмного забезпечення становить 50 людино-днів.

Вважатимемо, що зарплата розробника складає 1000 грн/день. Таким чином, орієнтовна вартість реалізації даного ПЗ складає 50000 грн.

2.6 Постановка завдання на розробку програмного забезпечення

Головною метою розробки є прискорення та спрощення процесу підготовки даних до подальшого аналізу, зокрема із застосуванням алгоритмів машинного навчання, а також підвищення ефективності роботи спеціалістів з аналізу даних.

Досягнення мети розробки планується шляхом створення вебсервісу, що дозволить автоматизувати аналіз табличних даних у багатьох форматах та генерувати рекомендаційні звіти на основі цілей користувача.

Програмне забезпечення буде реалізовано у вигляді вебсервісу з клієнт-серверною архітектурою, що має обмежений перелік доступних операцій та не зберігає стан між запитами, що дозволяє працювати без дотримання фіксованого сценарію.

Функціональні задачі, що підлягають вирішенню в ході реалізації програмного забезпечення на дипломний проєкт, включають обмін даними між користувачем та сервером, попередню обробку даних, генерацію рекомендацій у вигяді PDF-звіту та ефективне управління сховищем даних на сервері.

У результаті розробки очікується створення стабільного, масштабованого та легкорозширюваного вебсервісу, який може бути використаний в якості допоміжного інструмента при виконанні задач, пов’язаних з аналізом даних у різноманітних сферах, включаючи освіту, бізнес, маркетинг та науку.

Висновки до розділу

У ході розроблення вимог до вебсервісу для автоматизованого аналізу даних та генерації звітів було виділено 6 варіантів використання, побудовано UML-діаграму з ключовими акторами та основними функціями, а також детально описано кожен із варіантів.

На основі цього сформульовано 18 функціональних вимог і побудовано матрицю трасування, що демонструє зв’язок між варіантами використання та вимогами.

Також були визначені критерії якості у вигляді нефункціональних вимог, класифікованих за групами, і розроблено специфікацію системних вимог до користувача.

Для оцінки вартості розробки застосовано метод підрахунку точок варіантів використання: загальне значення склало 50 одиниць, а оціночна вартість – 50 тисяч гривень.

Під час постановки завдання визначено мету, цілі, задачі та загальні характеристики програмного забезпечення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Ще на етапі аналізу відомих технічних рішень було прийнято рішення про використання архітектурного патерну клієнт-сервер у поєднанні з монолітною архітектурою програмного забезпечення. Такий підхід дозволяє централізовано реалізувати всю бізнес-логіку на сервері, забезпечити простоту розгортання та ефективне обслуговування API без необхідності розділення сервісу на окремі підсистеми.

Високорівневу архітектуру представлено у вигляді C4 Model [13]. На рисунку 3.1 наведено рівень 1 моделі – контекстна діаграма. На рисунку 3.2 наведено рівень 2 моделі – діаграма контейнерів.

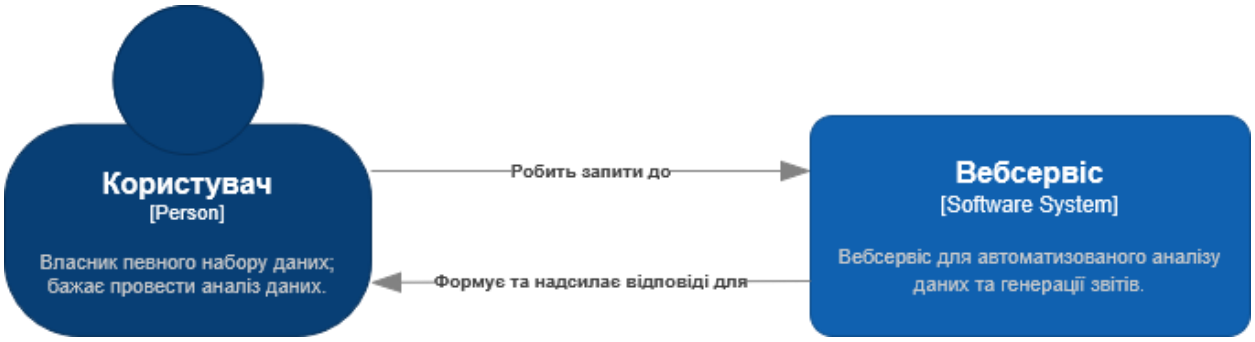


Рисунок 3.1 – Контекстна діаграма (C4 Model рівень 1)

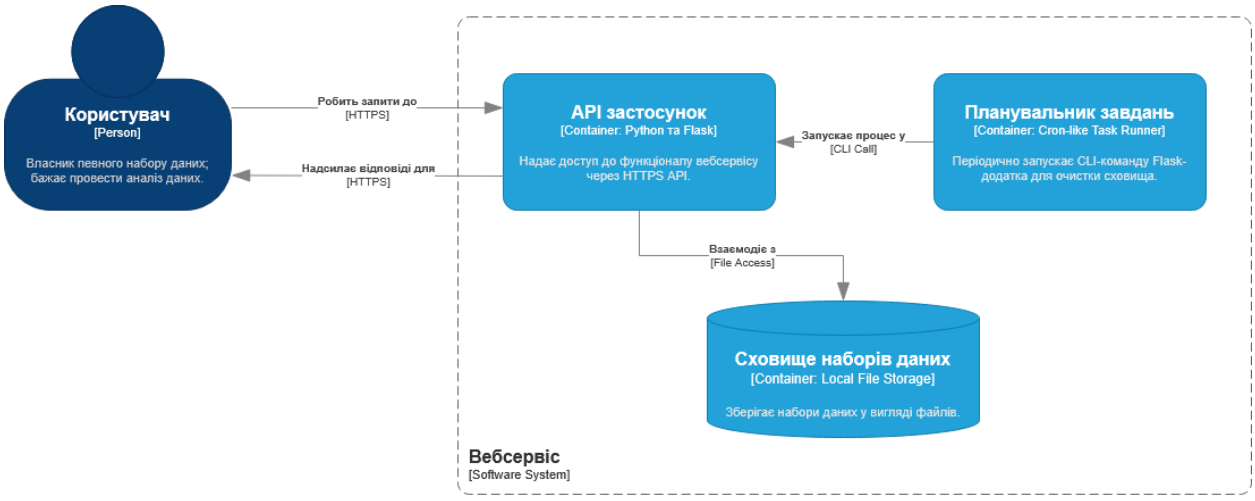


Рисунок 3.2 – Діаграма контейнерів (C4 Model рівень 2)

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

На даному етапі розробки виникає необхідність ухвалення низки архітектурних рішень, спрямованих на забезпечення гнучкості, простоти підтримки, а також відповідності функціональним і нефункціональним вимогам до системи.

Першим кроком є вибір типу бази даних. З метою спрощення процесу розробки та розгортання сервісу в якості сховища наборів даних доцільно використати файлове сховище. Таке рішення забезпечує швидкий, прямий та безпечний доступ до необхідних даних, а також не вимагає налаштування повноцінної СУБД.

Далі важливо розмежувати доступ до даних для кінцевих користувачів. Концептуальна модель системи передбачає одноразове використання сервісу для конкретного набору даних (як правило, один файл використовується для створення одного звіту або виконання одного циклу попередньої обробки), не зберігає стан між запитами та, відповідно, не потребує зберігання сесій або облікових записів. Як наслідок, доступ до завантаженого набору даних здійснюється за унікальним складним ключем доступу, який генерується під час завантаження файлу та надається користувачеві для подальшої обробки. Таке рішення забезпечує високий рівень безпеки, а також простоту як у використанні, так і в реалізації.

Сервіс не потребує інтеграції із зовнішніми системами, що дозволяє розгортати його як повністю автономний програмний компонент.

Система управління сховищем даних дозволяє видалення лише застарілих наборів даних (створених більше 24-х годин тому), таким чином забезпечуючи збереження файлів протягом даного періоду часу. Для запуску процесу очищення сховища реалізується окрема CLI-команда, що може бути легко інтегрована у cron-подібний планувальник завдань.

Для централізованого управління параметрами сервісу передбачено використання єдиного конфігураційного файлу, де будуть зберігатися важливі налаштування.

Відповіді на запити клієнта передбачено формувати у JSON з обмеженням глибини вкладеності до трьох рівнів для спрощення розуміння.

Для створення серверної частини вебзастосунків часто використовуються мови Python, Java та PHP, що вирізняються зрозумілим синтаксисом, масштабованістю та гнучкістю [14]. Для обґрунтування вибору проведено порівняльний аналіз (табл. 3.1).

Таблиця 3.1 – Порівняння популярних серверних мов програмування

Критерій	Python	Java	PHP
Платформи	Windows, macOS, Linux	Будь-який пристрій з JVM	Windows, macOS, Linux, Unix
Швидкість розробки	Висока	Середня	Висока
Підтримка сторонніх бібліотек	Дуже багато	Широка екосистема	Багато рішень для веб
Інтеграція	Легка з іншими технологіями	Добра, особливо у JVM-середовищі	Проста з HTML, СУБД, хостингами
Продуктивність	Низька	Висока	Низька
Споживання пам’яті	Високе споживання	Середнє (контроль JVM)	Низьке
Багатопоточність	Обмежена (GIL)	Повна підтримка	Обмежена
Типізація	Динамічна	Статична	Динамічна
Призначення	Розробка вебзастосунків, аналіз даних та машинне навчання, автоматизація та скрипти, розробка ігор та мережевих додатків	Корпоративні рішення, мобільна розробка, наукові обчислення, розподілені системи та обробка великих даних	Розробка динамічних вебсторінок та застосунків, електронна комерція, серверні скрипти, API та вебсервіси
Наявність досвіду розробки	Значний	Мінімальний	Відсутній

З огляду на цілі вебсервісу, для розробки обрано мову Python [15].

Порівняльний аналіз середовищ розробки для Python представлено на рисунку 3.3.

Назва	PyCharm	Visual Studio Code (VS Code)	Spyder	Jupyter Notebook	Python IDLE
Платформа	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS	Будь-яка	Windows, macOS, Linux
Вартість	Безкоштовна Community Edition, Платна Professional Edition	Безкоштовна	Безкоштовна	Безкоштовна	Безкоштовна
Розширення	Має велику кількість розширень та плагінів, але багато з них доступні тільки у версії Professional	Має велику кількість розширень та плагінів, доступних безкоштовно	Має певну кількість розширень та плагінів, але менше порівняно з іншими IDE	Обмежена	Має обмежений функціонал та розширення
Інтеграція з Git	Так	Так	Так	Ні	Ні
Дебагер	Так	Так	Так	Так	Ні
Підтримка веброзробки	Так	Так	Ні	Ні	Ні
Легкість використання	Середня/ висока	Висока	Висока	Середня	Висока
Орієнтація на наукові обчислення	Ні	Ні	Так	Так	Ні

Рисунок 3.3 – Порівняння популярних IDE для Python [16]

Враховуючи об’єктивні переваги та особистий досвід використання, було обрано PyCharm в якості середовища розробки.

Для реалізації вебфункціоналу застосунку обрано фреймворк Flask. Його переваги: [17]

– підходить як для малих, так і для масштабованих проєктів;

– залишає розробнику свободу у виборі бібліотек та інструментів;

– має просту архітектуру, хорошу документацію та велику спільноту;

– підтримується багатьма хостингами та легко розгортається.

3.3 Конструювання програмного забезпечення

3.3.1 Моделювання та валідація вхідних даних

Для структурованого представлення вхідних параметрів запитів та забезпечення їхньої коректності використовуються моделі, реалізовані за допомогою бібліотеки Pydantic. Дана бібліотека дозволяє явно задавати типи даних для полів, значення за замовчуванням, обмеження та правила валідації, що істотно спрощує перевірку запитів на рівні API.

Для кожної операції в сервісі, виконання якої передбачає можливість задання параметрів користувачем, було створено окрему модель, яка описує допустиму структуру даних.

Під час надходження запиту моделі автоматично перевіряють типи значень, обов’язковість полів та виконують вбудовану валідацію. У разі помилки бібліотека надає відповідь зі зрозумілим описом, що дозволяє зекономити велику кількість часу порівняно зі створенням повідомлень вручну. Завдяки типізації та механізмам, які надає Pydantic, вдається зменшити кількість ручної обробки помилок і забезпечити чистий та підтримуваний код.

У таблицях 3.2 – 3.5 можна переглянути опис моделей користувацьких параметрів у запитах.

Таблиця 3.2 – Pydantic модель параметрів зчитування файлу

Назва поля	Тип даних	Обов’язкове	Опис
separator	Рядок	ні	Роздільник значень у CSV
thousands	Символ	ні	Роздільник тисяч у CSV або Excel
decimal	Символ	ні	Десятковий роздільник у CSV або Excel
sheet_name	Рядок або ціле число	ні	Назва аркуша у Excel файлі
table_name	Рядок	ні	Назва таблиці у SQLite файлі

Таблиця 3.3 – Pydantic модель параметрів попередньої обробки

Назва поля	Тип даних	Обов’язкове	Опис
make_copy	Логічний	ні	Чи зберігати результат як новий набір даних
case_insensitive_columns	ColumnList	ні	Стовпці, які треба привести до одного регістру
clear_punct_columns	ColumnList	ні	Стовпці, які треба очистити від пунктуації
clear_digits_columns	ColumnList	ні	Стовпці, які треба очистити від цифр
row_range_start	Додатне ціле число	ні	Початковий індекс вибірки з даних
row_range_end	Додатне ціле число	ні	Кінцевий індекс вибірки з даних
row_range_step	Додатне ціле число	ні	Крок для вибірки з даних
index_cols	ColumnList	ні	Стовпці, які є індексом в наборі даних
fill_na_values	JSON значення	ні	Значення для заповнення пропусків по стовпцях
mfill	Логічний	ні	Чи заповнювати пропуски медіаною/модою
ffill	Логічний	ні	Чи заповнювати пропуски попереднім дійсним значенням у стовпці
bfill	Логічний	ні	Чи заповнювати наступним дійсним значенням у стовпці
drop_na	Перелічуване (рядок)	ні	Вісь видалення пропусків: rows / columns
drop_outliers	Логічний	ні	Чи видаляти викиди
outliers_threshold	Додатне дійсне число	ні	Поріг визначення викидів
drop_duplicates	Логічний	ні	Чи видаляти дублікати
duplicate_subset	ColumnList	ні	Стовпці, збіги в яких вважати дублікатами

Продовження таблиці 3.3

duplicate_keep	Перелічуване (рядок)	ні	Який дублікат залишити: first / last / none
datetime_columns	ColumnList	ні	Стовпці, які треба привести до datetime типу
category_columns	ColumnList	ні	Стовпці, які треба привести до category типу
join_small_cat	Логічний	ні	Чи об’єднувати малі категорії даних
joined_category_name	Рядок	ні	Назва стовпця, утвореного об’єднанням малих категорій
categories_threshold	Додатне дійсне число	ні	Максимальна частка, при якій категорія вважається малою
scale_numeric	Логічний	ні	Чи масштабувати числові дані
scaling_method	Перелічуване (рядок)	ні	Метод масштабування: max_abs_scaling / min_max_scaling / z_score

Примітка. ColumnList – це рядок, список рядків або спец. символ *.

Таблиця 3.4 – Pydantic модель параметрів рекомендаційного звіту

Назва поля	Тип даних	Обов’язкове	Опис
analysis_task	Перелічуване (рядок)	так	Тип задачі аналізу: regression / classification / clusterization
target_col	Рядок	ні	Назва цільової змінної
include_basic_stats	Логічний	ні	Чи додавати базові статистики у звіт
include_visualizations	Логічний	ні	Чи додавати візуалізації у звіт
dpi	Додатне ціле число	ні	Роздільна здатність візуалізацій у dpi
theme	Перелічуване (рядок)	ні	Тема звіту: light / dark
show_time	Логічний	ні	Чи додавати дату і час генерації у звіт

Таблиця 3.5 – Pydantic модель параметрів експорту

Назва поля	Тип даних	Обов’язкове	Опис
format	Перелічуване (рядок)	ні	Формат експорту набору даних: csv / json / excel / pickle

3.3.2 Розробка алгоритмів обробки даних

У вебсервісі відсутні обчислювальні алгоритми в класичному розумінні, проте важливо формалізувати процеси, що відбуваються в ході виконання ключових операцій з обробки даних.

Процес зчитування вхідного набору даних реалізується у класі DataFrameLoader і передбачає поетапне виконання таких кроків:

1) визначення типу файлу – здійснюється шляхом аналізу розширення (підтримувані формати: .csv, .xls, .xlsx, .json, .db).

2) вибір методу зчитування – для кожного типу файлу реалізовано окремий метод зчитування, що використовує відповідну функціональність бібліотеки pandas.

3) формування параметрів зчитування – використовується об’єкт класу LoadingParams, який містить усі необхідні налаштування, визначені користувачем (див. табл. 3.2). Ці параметри передаються у відповідний метод зчитування.

4) зчитування набору даних – викликається функція pandas, що відповідає формату файлу, з переданими параметрами.

5) перевірка результату – здійснюється перевірка наявності даних у зчитаній таблиці. У разі порожнього результату формується повідомлення про помилку.

Попередня обробка даних реалізується у класі DataFrameLoader. Алгоритм реалізовано у вигляді гнучкого, параметризованого конвеєра умовних операцій. Основні етапи обробки включають:

1) нормалізація текстових даних, яка може охоплювати:
   – приведення до нижнього регістру;
   – видалення розділових знаків;
   – видалення цифр.
2) фільтрація рядків за допомогою зрізу (вказуються початок, кінець та крок).
3) встановлення індексу – визначення однієї або кількох колонок як індексу таблиці.
4) заповнення пропущених значень:
   – заданими користувачем значеннями;
   – статистичними характеристиками (медіана – для числових, мода – для нечислових);
   – методами прямого (ffill) або зворотного (bfill) заповнення.
5) видалення рядків/стовпців з пропущеними значеннями (за заданою віссю).
6) видалення викидів – за допомогою Z-оцінки (Z-score [29]) за заданим порогом.
7) видалення дублікатів – із зазначенням стовпців та стратегії (first, last, False) збереження.
8) перетворення типів даних:
   – приведення стовпців до типу datetime;
   – приведення стовпців до типу category.
9) об’єднання рідкісних категорій – із можливістю вказати порогову частку частоти та назву об'єднаної категорії.
10) масштабування числових ознак – із вибором методу (max_abs_scaling, min_max_scaling, z_score).

Параметри для налаштування кожного етапу отримуються з об’єкта PreprocessingParams (див. таблицю 3.3). Деструктивні операції (які можуть зменшити розмір набору даних) супроводжуються перевіркою на порожність після виконання.

Рекомендаційний аналіз здійснюється у класі DataFrameAnalyzer, отримує на вхід об’єкт AnalysisParams (див. таблицю 3.4) і передбачає виконання наступних етапів:

1) додавання у звіт загальної структури та базових статистик набору даних – якщо користувач не вказав явно, що не потрібно додавати.

2) створення та додавання у звіт рекомендацій, що є специфічними для конкретної задачі машинного навчання.

3) створення та додавання у звіт рекомендацій щодо відбору важливих ознак для подальшого аналізу.

4) створення та додавання у звіт рекомендацій щодо конструювання ознак.

Процес експорту набору даних не потребує створення окремого класу, оскільки є достатньо простим і виконується безпосередньо в модулі обробки відповідного запиту. Для управління параметрами експорту використовується модель ExportParams (див. таблицю 3.5), яка містить вхідний параметр – очікуваний формат файлу, а також вихідні параметри: функцію перетворення в послідовність байтів, відповідне розширення та тип змісту (Content-Type) для HTTP-відповіді.

3.3.3 Опис структури сховища даних

У вебсервісі реалізовано локальне файлове сховище, яке використовується для тимчасового зберігання наборів даних, завантажених користувачем. Це сховище ізольоване від основної логіки застосунку та використовується як кеш для швидкого доступу до попередньо оброблених або частково аналізованих таблиць.

Усі файли зберігаються у каталозі, шлях до якого визначається змінною конфігурації DATASET_STORAGE (за замовчуванням, datasets/). Структура даного каталогу є плоскою, тобто всі файли зберігаються без вкладених директорій.

Кожен збережений набір даних серіалізується у форматі pickle [30] та зберігається у вигляді файлу з іменем, що формується за схемою “<ідентифікатор_набору_даних>__<ключ_доступу_до_набору_даних>”. Таке іменування дозволяє забезпечити ізоляцію доступу та швидкий пошук даних за відповідними атрибутами.

3.3.4 Модульність та структура вебзастосунку

Для забезпечення читабельності, підтримуваності та розширюваності коду, у процесі його написання дотримано певних архітектурних принципів та рішень.

Одним з архітектурних рішень є використання фабрики – шаблону, що дозволяє створювати та використовувати кілька екземплярів застосунку одночасно:

```
def create_app(config_class: object = Config) -> Flask:
    app: Flask = Flask(__name__)
    app.config.from_object(config_class)
    … логіка застосунку
    return app
```

Наступним важливим рішенням є використання концепту ескізів у Flask (Blueprints), що дозволяє розділити обробники запитів у великому застосунку на логічні групи [21]. Виділено наступні ескізи:

– data_exchange – обробка запитів, пов’язаних з обміном даними між користувачем та сервером;

– preprocessing – обробка запитів, пов’язаних з попередньою обробкою даних;

– reporting – обробка запитів, пов’язаних з рекомендаційним аналізом даних;

– system – обробка запитів, пов’язаних із службовою логікою.

Реєстрація ескізів у головному Flask застосунку:

app.register_blueprint(system_bp)

app.register_blueprint(data_exchange_bp)

app.register_blueprint(preprocessing_bp)

app.register_blueprint(reporting_bp)

Окремо реалізовано систему управління сховищем даних у вигляді внутрішнього розширення Flask. Такий підхід дає змогу інкапсулювати логіку збереження, завантаження та очищення файлів, підтримуючи чисту архітектуру застосунку. Крім того, це дозволяє використовувати спільне сховище для кількох екземплярів застосунку, що особливо корисно в умовах масштабування або роботи в різних середовищах. Створення системи управління сховищем та інтеграція у Flask застосунок:

storage = Storage()

app = Flask(__name__)

storage.init_app(app)

Для чіткого розділення логіки моделі параметрів винесено в окремий пакет models, допоміжні класи – в controllers, розширення – в extensions, а помилки та їх обробку – до errors.py та handlers.py відповідно (рисунок 3.4).

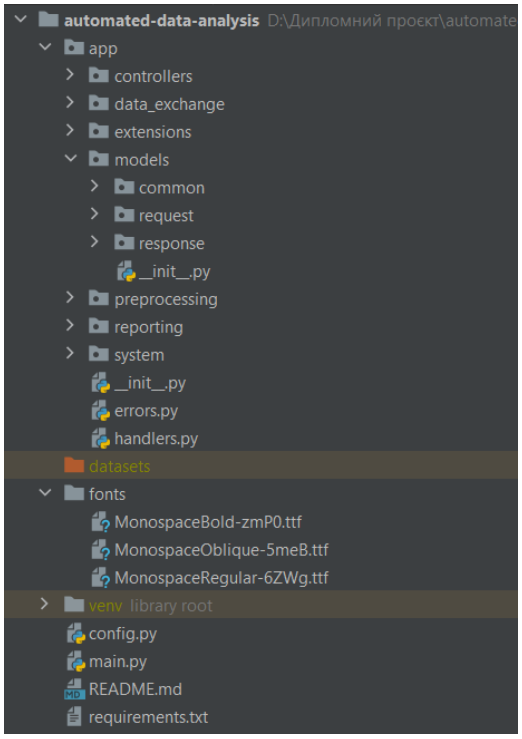


Рисунок 3.4 – Модульна структура застосунку

3.3.5 Розробка механізму обробки помилок

У вебзастосунку реалізовано централізовану систему обробки помилок шляхом визначення власних класів винятків для типових помилкових ситуацій (наприклад, відсутність даних, помилкові параметри запиту, помилка при зчитуванні файлу з даними) та підключення глобальних обробників через механізм errorhandler у Flask. Це дозволяє не лише логічно відокремити обробку виняткових ситуацій від основної бізнес-логіки, а й гарантувати формування стандартизованих HTTP-відповідей з описом помилки у форматі JSON.

Усі власні класи винятків є нащадками готових винятків, визначених у модулі exceptions програмної бібліотеки Werkzeug [35]. Такий підхід дозволяє мінімізувати зусилля, необхідні для коректної реалізації поведінки винятків згідно зі стандартами HTTP-протоколу (включно з кодами статусу, повідомленнями тощо), а також уніфікувати структуру помилкових відповідей. Таким чином, власні та системні HTTP-винятки є нащадками класу HTTPException і можуть оброблятися єдиним глобальним обробником, що значно спрощує підтримку та забезпечує послідовність у відповідях:

```
@app.errorhandler(HTTPException)
def handle_http_exception(e: HTTPException) -> tuple[FlaskResponse, int]:
    response = {
        "error": e.name,
        "code": e.code,
        "description": e.description
    }
    return jsonify(response), e.code
```

Використання моделей Pydantic моделей для валідації параметрів запитів дозволяє централізовано виконувати перевірку типів і значень без необхідності писати додаткову логіку. Оскільки всі помилки валідації у Pydantic є екземплярами класу ValidationError, їх можна зручно та уніфіковано обробляти на рівні всього застосунку:

```
@app.errorhandler(ValidationError)
def handle_validation_error(e: ValidationError) -> tuple[FlaskResponse, int]:
    simplified_errors = [
        {
            "parameter": err["loc"][0],
            "message": err["msg"],
            "value": err["input"]
        }
        for err in e.errors()
    ]
    return handle_http_exception(ValidationFailed(simplified_errors))
```

Для перехоплення неочікуваних помилок реалізовано окремий обробник загального винятку Exception, який дозволяє повертати уніфіковану відповідь навіть у випадках непередбачених збоїв. Такий обробник логуватиме помилку для подальшого її виправлення розробником та повертатиме стандартне повідомлення про внутрішню помилку сервера:

```
@app.errorhandler(Exception)
def handle_unexpected_error(e: Exception) -> tuple[FlaskResponse, int]:
    service_name = request.blueprint or "APP"
    tb_str = ''.join(traceback.format_exception(type(e), e, e.__traceback__))
    print(f"[{service_name}] Unexpected error:\n{tb_str}")
    return handle_http_exception(InternalServerError())
```

Обидва обробники інтегруються з єдиною системою обробки HTTP-помилок, що забезпечує послідовну структуру відповіді незалежно від джерела помилки. Такий підхід сприяє покращенню досвіду користувача та забезпечує зручність і узгодженість в реалізації.

3.3.6 Інструменти розробки

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.6.

Таблиця 3.6 – Опис утиліт

№ п/п	Назва утиліти	Опис утиліти
1	PyCharm [18]	IDE для розробки вебсервісу
2	Postman [19]	Платформа для тестування API
3	draw.io [20]	Сервіс для побудови діаграм
4	Flask [21]	Мікрофреймворк для розробки вебдодатків на Python
5	Pydantic [22]	Бібліотека для валідації даних на Python
6	pandas [23]	Бібліотека для обробки та аналізу даних на Python
7	NumPy [24]	Бібліотека для наукових обчислень на Python
8	Matplotlib [25]	Бібліотека для створення візуалізацій на Python
9	Seaborn [26]	Бібліотека для візуалізації даних на Python, побудована на основі Matplotlib
10	FPDF (fpdf2) [27]	Бібліотека для програмного формування PDF документів на Python
11	Scikit-learn [28]	Бібліотека для виконання задач, пов’язаних з машинним навчанням, на Python
12	Werkzeug [35]	Бібліотека для створення WSGI-сумісних вебдодатків на Python, що є невід’ємною частиною Flask
13	flask-pydantic-spec [36]	Бібліотека для інтеграції валідації параметрів через Pydantic та автоматичної генерації документації OpenAPI у Flask-застосунках
14	sqlite3	Стандартний модуль мови Python для роботи з sqlite базами даних
15	typing	Стандартний модуль мови Python для підтримки підказок типів

Продовження таблиці 3.6

16	os	Стандартний модуль мови Python для використання залежного від ОС функціоналу
17	io	Стандартний модуль мови Python для робити з різними типами вводу/виводу

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Під час розробки вебсервісу важливо враховувати загрози безпеки, особливо при обробці файлів користувача чи параметрів запитів. Типові загрози вебзастосунків [31]:

– ін’єкції (Injection) – виникають, коли дані користувача передаються у внутрішні команди без належної перевірки, що дозволяє виконання шкідливих інструкцій.

– відмова в обслуговуванні (DoS) – масове створення запитів до сервера з метою його перевантаження чи блокування доступу.

– міжсайтова підробка запиту (CSRF) – використання чинної автентифікації користувача для здійснення небажаних запитів.

– міжсайтове скриптування (XSS) – вставлення зловмисного коду у вебсторінки для викрадення даних або сеансів користувачів.

– помилкова конфігурація безпеки (Security Misconfiguration) – неналежне налаштування захисту в застосунку або інфраструктурі.

– зовнішні сутності в XML (XXE) – уразливість при неправильному налаштуванні XML-парсингу, що може дати доступ до внутрішніх файлів.

– небезпечна десеріалізація (Insecure Deserialization) – експлуатація вразливостей механізмів десеріалізації, що дозволяє виконання довільного коду.

У таблиці 3.7 наведено результати аналізу загроз безпеки з урахуванням поточної реалізації сервісу.

Таблиця 3.7 – Аналіз загроз безпеки вебсервісу

Загроза	Ризик виникнення	Вжиті заходи
Injection	Відсутній – дані від користувача не використовуються напряму в SQL або shell	–
DoS	Помірний – можливе масове надсилання запитів або великих файлів	Обмеження розміру файлів (100 МБ), часу виконання запиту (300 с)
CSRF	Відсутній – автентифікація не базується на cookie або сесіях	–
XSS	Відсутній – сервіс не генерує HTML і не виконує JavaScript	–
Security Misconfig.	Помірний – можливі наслідки при неправильному розгортанні	Встановлено DEBUG=False у production, окремі середовища dev/prod
XXE	Відсутній – не використовується парсинг XML	–
Insecure Deserializ.	Помірний – збереження даних між запитами здійснюється у форматі Pickle	Неможливість передачі або зміни файлів Pickle користувачем

Висновки до розділу

У ході конструювання програмного забезпечення було реалізовано та описано процес створення вебсервісу для автоматизованого аналізу даних і генерації звітів.

Робота над даним етапом розпочалася з розробки архітектури системи та її високорівневого представлення за допомогою діаграм першого та другого рівнів C4-моделі.

Для переходу до етапу програмування було ухвалено низку архітектурних рішень, зокрема щодо вибору типу бази даних, розмежування доступу до її вмісту, а також способів дотримання окремих нефункціональних вимог.

Після цього проведено порівняльний аналіз мов програмування, у результаті чого обрано Python завдяки її перевагам для завдань аналізу даних. Як інтегроване середовище розробки використано PyCharm з огляду на його функціональні можливості.

Під час реалізації функціоналу вебсервісу описано ключові аспекти програмного забезпечення: моделі даних, алгоритми обробки, структуру сховища, модульну організацію коду, механізми обробки помилок та використані інструменти.

Завершальним етапом став попередній аналіз потенційних вразливостей на основі поширених типів загроз, а також опис заходів безпеки, спрямованих на їх запобігання.

У результаті виконання даного розділу реалізовано основну частину програмного коду розроблюваного вебзастосунку.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Для оцінки якості розробленого програмного забезпечення використано онлайн-інструмент SonarQube Cloud [32], який автоматично аналізує вихідний код та обчислює низку метрик, що відображають його якість, складність, дублювання та потенційні проблеми.

Метрики, за якими було здійснено аналіз:

– контрольна точка якості (Quality Gate) – узагальнена контрольна точка якості, яка базується на ключових метриках та визначає, чи відповідає код вимогам до якості.

– оцінка безпеки (Security rating) – показник, що оцінює потенційні вразливості в коді, зокрема наявність небезпечних конструкцій чи практик.

– гарячі точки безпеки (Security Hotspots) – ділянки коду, які потребують ручного перегляду на предмет потенційних загроз безпеці.

– оцінка надійності (Reliability rating) – показник, що оцінює місця в коді, де поведінка може не відповідати очікуваній.

– оцінка підтримуваності (Maintainability rating) – метрика, що визначає легкість модифікації та підтримки коду на основі наявних поганих практик.

– дублювання коду (Code Duplications) – частка коду, що повторюється, що може ускладнювати підтримку та підвищувати ймовірність помилок.

– цикломатична складність (Cyclomatic Complexity) – оцінка мінімальної кількості тест кейсів, потрібних для повного покриття коду.

– когнітивна складність (Cognitive Complexity) – метрика, що оцінює зусилля, необхідні для розуміння коду людиною.

Зведені результати аналізу представлено в таблиці 4.1, а скріншот його підсумку – на рисунку 4.1.

Таблиця 4.1 – Метрики оцінки якості вебсервісу за SonarQube Cloud

Метрика	Значення	Примітка
Quality Gate	Passed	ПЗ відповідає стандартам якості
Security rating	A	Відсутні вразливості
Security Hotspots	1	Потрібний власний перегляд
Reliability rating	A	Відсутні проблеми
Maintainability rating	A	Висока підтримуваність
Code Duplications	0.0%	Відсутні повторення коду
Cyclomatic Complexity	191 (~2.08/функція)	Низька розгалуженість логіки
Cognitive Complexity	134 (~1.46/функція)	Висока зрозумілість коду

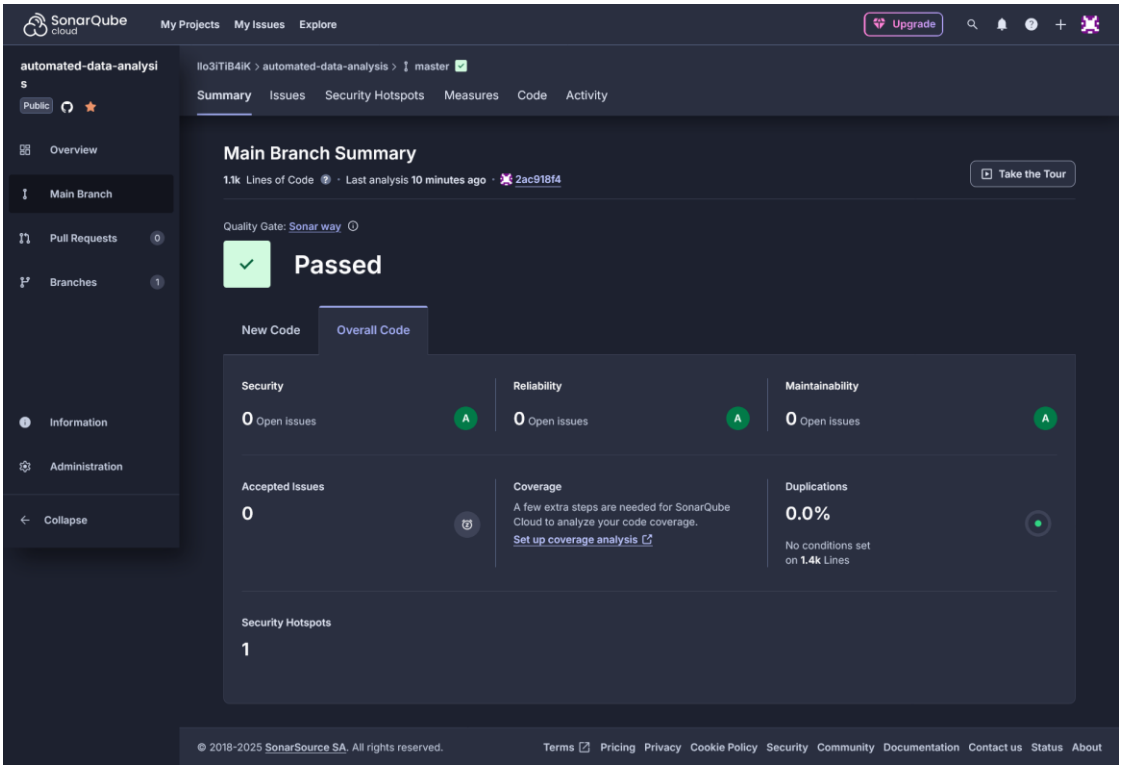


Рисунок 4.1 – Підсумок статичного аналізу коду проєкта

Крім технічної якості коду, проєкт було проаналізовано на відповідність сформульованим нефункціональним вимогам, наведеним у розділі 2.3. Перевірку проведено шляхом емпіричних тестів, а також на основі результатів спостереження за поведінкою системи під час її роботи:

– доступність: сервіс функціонував стабільно впродовж усіх тестових сесій. Працездатність підтверджувалася реакцією сервера на запити. Збоїв не зафіксовано.

– продуктивність: API успішно обробляв до 10 одночасних запитів без помітного зниження швидкодії. Генерація звіту на наборі даних розміром 80 МБ тривала ~3 хвилини, що повністю відповідає вимогам (до 5 хвилин).

– безпека: реалізовано валідацію вхідних даних (розміру, формату), що запобігає атакам типу DoS. Обмін даними відбувається через HTTPS, що забезпечує конфіденційність.

– зберігання даних: реалізовано тимчасове зберігання завантажених файлів на сервері з обмеженням за часом (24 години), після чого дані автоматично видаляються.

– масштабованість: структура додатку дозволяє перенесення на більш продуктивне середовище без зміни логіки чи архітектури за рахунок використання масштабованих технологій та бібліотек.

– конфігурованість: параметри (максимальний допустимий розмір файлу, час зберігання файлу, заголовок ключа доступу, шлях до сховища наборів даних, середовище) задаються через конфігураційний файл (config.py) і можуть бути змінені без втручання в логіку програми.

– зручність використання: реалізовано REST-підхід із семантичними HTTP-методами. Формат JSON-відповідей стандартизовано, вкладеність обмежена трьома рівнями. Помилки повертаються у структурованому вигляді. Документація формується автоматично та доступна за адресою кореневого маршруту (/).

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Проведено мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.2 – 4.14.

Таблиця 4.2 – Тест 1. Завантаження валідного набору даних на сервер

Початковий стан системи	У сховищі на сервері є вільне місце (~100 КБ)
Вхідні дані	Файл: Financial Sample.xlsx Тип контенту: multipart/form-data
Опис проведення тесту	1. Надіслати POST запит до /datasets 2. Додати у заголовки тип контенту за ключем "Content-Type" 3. Додати у тіло файл за ключем "file"
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить message ("Dataset uploaded successfully"), dataset_id (рядок), access_key (рядок), next_step (URL), metadata (базова інформація про набір даних).
Фактичний результат	Збігається з очікуваним

Таблиця 4.3 – Тест 2. Завантаження пустого набору даних на сервер

Початковий стан системи	-
Вхідні дані	Файл: empty dataset.csv Тип контенту: multipart/form-data
Опис проведення тесту	1. Надіслати POST запит до /datasets 2. Додати у заголовки тип контенту за ключем "Content-Type" 3. Додати у тіло файл за ключем "file"
Очікуваний результат	Отримано статус-код 422 UNPROCESSABLE ENTITY. Відповідь містить error ("Empty Dataset"), code (422), description ("The uploaded file 'empty dataset.csv' contains no data").
Фактичний результат	Збігається з очікуваним

Таблиця 4.4 – Тест 3. Передача коректних параметрів зчитування

Початковий стан системи	У сховищі на сервері є вільне місце (~100 КБ)
Вхідні дані	Файл: Financial Sample.xlsx Тип контенту: multipart/form-data Параметри зчитування: {"decimal": ".", "sheet_name": "Sheet1"}
Опис проведення тесту	1. Надіслати POST запит до /datasets 2. Додати у заголовки тип контенту за ключем "Content-Type" 3. Додати у тіло файл за ключем "file" та параметри зчит.
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить message ("Dataset uploaded successfully"), dataset_id (рядок), access_key (рядок), next_step (URL), metadata (базова інформація про набір даних).
Фактичний результат	Збігається з очікуваним

Таблиця 4.5 – Тест 4. Передача некоректних параметрів зчитування

Початковий стан системи	У сховищі на сервері є вільне місце (~100 КБ)
Вхідні дані	Файл: Financial Sample.xlsx Тип контенту: multipart/form-data Параметри зчитування: {"decimal": ",", "sheet_name": "123"}
Опис проведення тесту	1. Надіслати POST запит до /datasets 2. Додати у заголовки тип контенту за ключем "Content-Type" 3. Додати у тіло файл за ключем "file" та параметри зчит.
Очікуваний результат	Отримано статус-код 422 UNPROCESSABLE ENTITY. Відповідь містить error ("Reading Error"), code (422), description (детальний опис помилки).
Фактичний результат	Збігається з очікуваним

Таблиця 4.6 – Тест 5. Запит інформації про існуючий набір даних

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id> 2. Додати у заголовки access_key за ключем "x-dataset-token"
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить message ("Dataset found successfully"), dataset_id (рядок), next_step (URL), metadata (базова інформація про набір даних).
Фактичний результат	Збігається з очікуваним

Таблиця 4.7 – Тест 6. Запит інформації про неіснуючий набір даних

Початковий стан системи	У сховищі відсутній набір даних з dataset_id="abc123"
Вхідні дані	dataset_id: "abc123" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id> 2. Додати у заголовки access_key за ключем "x-dataset-token"
Очікуваний результат	Отримано статус-код 404 NOT FOUND. Відповідь містить error ("Not Found"), code (404), description ("Dataset not found or invalid access key.").
Фактичний результат	Збігається з очікуваним

Таблиця 4.8 – Тест 7. Запит інформації про дані без додавання ключа

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>
Очікуваний результат	Отримано статус-код 400 BAD REQUEST. Відповідь містить error ("Bad Request"), code (400), description ("Missing access key in headers.").
Фактичний результат	Збігається з очікуваним

Таблиця 4.9 – Тест 8. Завантаження набору даних з сервера

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/download 2. Додати у заголовки access_key за ключем "x-dataset-token"
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить заголовки Content-Type (text/csv; charset=utf-8) та Content-Length (82967).
Фактичний результат	Збігається з очікуваним

Таблиця 4.10 – Тест 9. Автоматичне видалення старих наборів даних

Початковий стан системи	Набір даних, якому присвоєно ідентифікатор "0e644c7b-2b49-4aa5-a916-67a77ff3dac7" завантажено понад 48 годин тому
Вхідні дані	dataset_id: "0e644c7b-2b49-4aa5-a916-67a77ff3dac7" access_key: "5cff58a2-6d73-4b4a-b95a-ecdd65fa7acf"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/download 2. Додати у заголовки access_key за ключем "x-dataset-token"
Очікуваний результат	Отримано статус-код 404 NOT FOUND. Відповідь містить error ("Not Found"), code (404), description ("Dataset not found or invalid access key.").
Фактичний результат	Збігається з очікуваним

Таблиця 4.11 – Тест 10. Запит рекомендаційного PDF-звіту

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea" Параметри аналізу: "analysis_task=regression", "target_col=Sales"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/report 2. Додати у заголовки access_key за ключем "x-dataset-token" 3. Додати до запиту параметри аналізу у URL або окремо
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить заголовки Content-Type (application/pdf) та Content-Length (809250).
Фактичний результат	Збігається з очікуваним

Таблиця 4.12 – Тест 11. Передача зайвих параметрів аналізу даних

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea" Параметри аналізу: "analysis_task=regression", "target_col= Sales", "invalid_param=abc123"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/report 2. Додати у заголовки access_key за ключем "x-dataset-token" 3. Додати до запиту параметри аналізу у URL або окремо
Очікуваний результат	Отримано статус-код 200 OK. Відповідь містить заголовки Content-Type (application/pdf) та Content-Length (809250).
Фактичний результат	Збігається з очікуваним

Таблиця 4.13 – Тест 12. Вказання некоректної задачі аналізу

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea" Параметри аналізу: "analysis_task=mltask", "target_col= Sales"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/report 2. Додати у заголовки access_key за ключем "x-dataset-token" 3. Додати до запиту параметри аналізу у URL або окремо
Очікуваний результат	Отримано статус-код 422 UNPROCESSABLE ENTITY. Відповідь містить error ("Validation Failed"), code (422), description (деталі помилки).
Фактичний результат	Збігається з очікуваним

Таблиця 4.14 – Тест 13. Нечислова цільова змінна для регресії

Початковий стан системи	Набір даних з dataset_id="4abedb7b-5939-49de-9def-6c370bb6c98f" був попередньо завантажений на сервер
Вхідні дані	dataset_id: "4abedb7b-5939-49de-9def-6c370bb6c98f" access_key: "e3463597-6633-4c7e-b611-8cfa135707ea" Параметри аналізу: "analysis_task=regression", "target_col=Product"
Опис проведення тесту	1. Надіслати GET запит до /datasets/<dataset_id>/report 2. Додати у заголовки access_key за ключем "x-dataset-token" 3. Додати до запиту параметри аналізу у URL або окремо
Очікуваний результат	Отримано статус-код 200 OK. Звіт містить примітку "* Target column 'Product' is not numeric. * No further reporting can be performed. Consider encoding or converting it.".
Фактичний результат	Збігається з очікуваним

4.3 Опис контрольного прикладу

Опис контрольного прикладу зроблено на основі бізнес-процесу «Отримання рекомендаційного звіту» (див. розділ 1.4), що демонструє базовий сценарій роботи користувача з вебсервісом. Для виконання сценарію використано довільний набір даних heart_2020_cleaned.csv [33].

Послідовність дій охоплює основний функціонал системи: обмін даними, автоматизована попередня обробка та генерація рекомендаційного звіту.

Для моделювання HTTP-запитів використано інструмент Postman, який забезпечує зручну взаємодію з API-інтерфейсом вебсервісу та демонстрацію відповіді у зрозумілому форматі.

Наведемо кроки виконання контрольного прикладу:

1) Завантаження набору даних на сервер:

Надіслано POST-запит на ендпоінт /datasets з додаванням файлу heart_2020_cleaned.csv у форматі form-data. У відповідь отримано статус-код «200 OK» та потрібну для наступних етапів роботи інформацію, що наведено на рисунку 4.2.

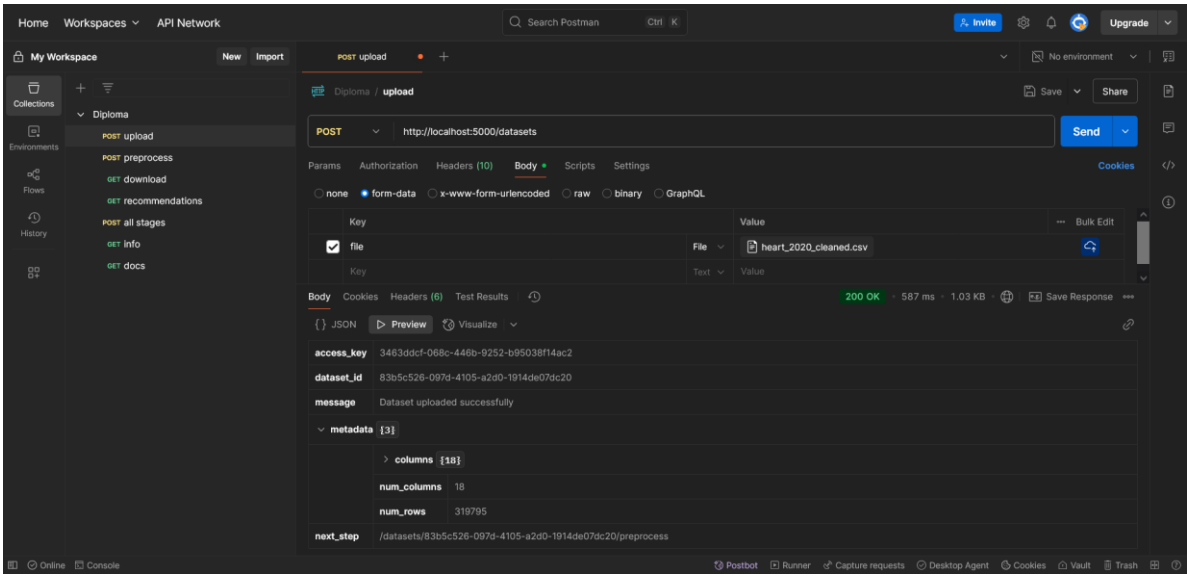


Рисунок 4.2 – Приклад відповіді сервісу на завантаження набору даних

2) (Опціонально) Попередня обробка даних:

Маючи dataset_id та access_key, надсилається POST-запит на ендпоінт /datasets/<dataset_id>/preprocess з довільними валідними параметрами попередньої обробки у тілі запиту. У відповідь отримано статус-код «200 OK» та оновлену інформацію про набір даних, що наведено на рисунку 4.3.

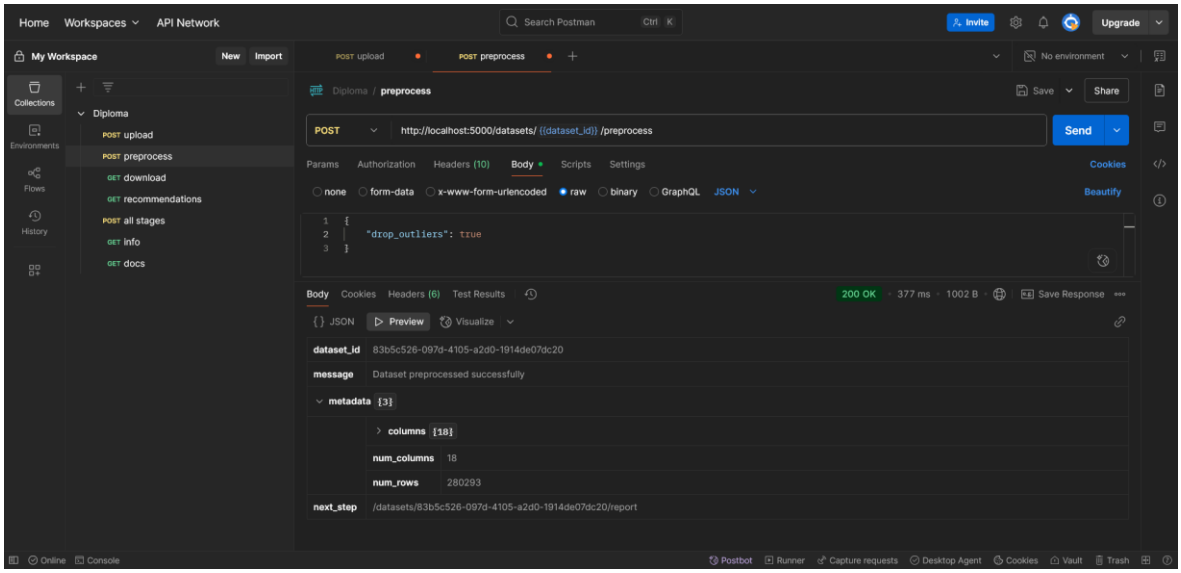


Рисунок 4.3 – Приклад відповіді сервісу на запит попередньої обробки даних

3) (Опціонально) Завантаження підготовленого набору:

Користувач має змогу скачати набір даних з сервера. Для цього надсилається GET-запит на ендпоінт /datasets/<dataset_id>/download. У відповідь отримано статус-код «200 OK» та дані у форматі CSV, що наведено на рисунку 4.4.

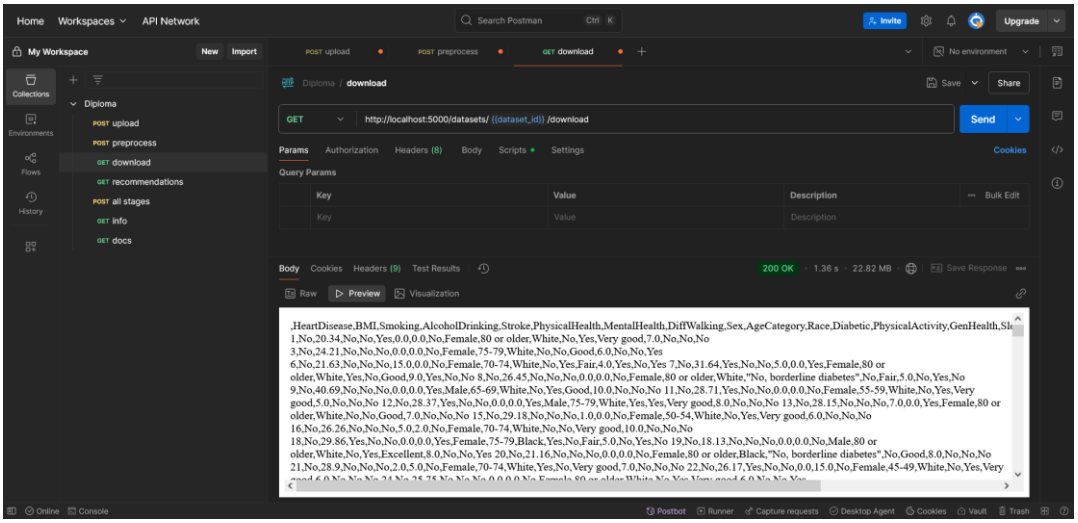


Рисунок 4.4 – Приклад відповіді на запит скачування набору даних

4) Отримання рекомендаційного звіту:

На даному етапі користувач може досягнути кінцевої мети використання сервісу шляхом надсилання POST-запиту на ендпоінт /datasets/<dataset_id>/report з необхідними параметрами аналізу у тілі запиту. У даному випадку задачею машинного навчання є класифікація, а цільовим стовпцем HeartDisease. У відповідь отримано статус-код «200 OK» та PDF-звіт, що наведено на рисунку 4.5.

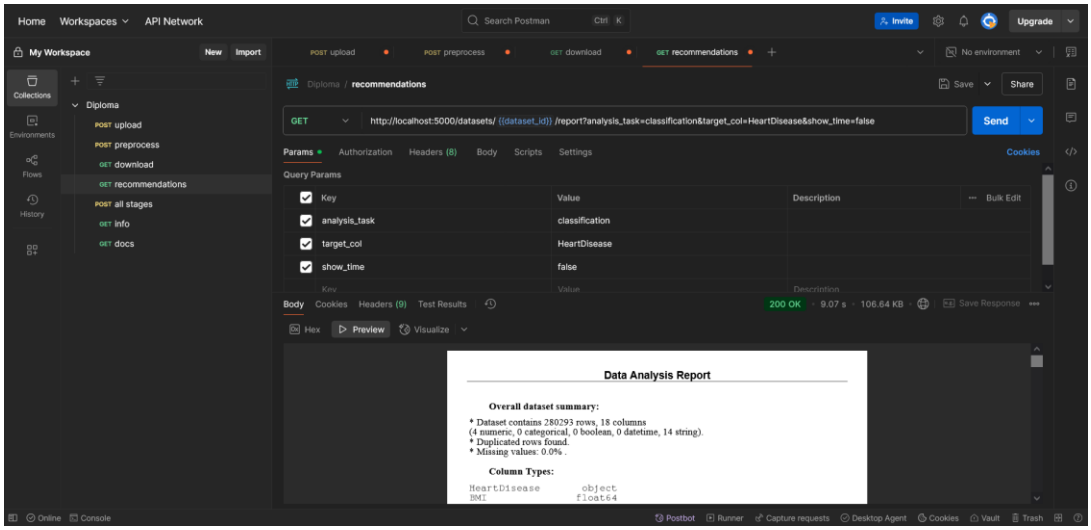


Рисунок 4.5 – Приклад відповіді на запит рекомендаційного звіту

Висновки до розділу

У розділі аналізу якості програмного забезпечення описано процес перевірки працездатності та тестування розробленого вебсервісу.

Першим кроком було використання інструменту статичного аналізу коду SonarQube Cloud. Оцінювання відбувалося за низкою ключових метрик, які відображають такі характеристики ПЗ, як безпека, надійність, доступність, підтримуваність та читабельність. Усі метрики продемонстрували високі показники якості, що підтверджує відповідність коду сучасним стандартам розробки. Проведено емпіричне тестування, в ході якого підтверджено виконання деяких нефункціональних вимог, сформульованих на попередніх етапах проєкту.

Далі було проведено мануальне тестування основного функціоналу вебсервісу. Було розроблено та виконано 13 позитивних та негативних тест-кейсів, кожен з яких описано в уніфікованому форматі. Усі тести пройдено успішно – фактичні результати відповідали очікуваним.

Завершальним етапом тестування стало виконання контрольного прикладу, що ілюструє типовий сценарій використання вебсервісу та охоплює ключовий функціонал програмного забезпечення. Даний приклад підтвердив мету та цілі розробки проєкту.

У результаті виконання зазначених етапів було протестовано вебсервіс для автоматизованого аналізу даних та генерації звітів. У ході тестування було виявлено та усунуто окремі помилки та недоліки ПЗ, що додатково покращило його якість.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання вебсервісу обрано хмарну платформу PythonAnywhere – спеціалізоване середовище для запуску Python застосунків у хмарі. Сервіс, зокрема, надає інструменти для створення, редагування та розгортання вебдодатків просто через браузер – без необхідності вручну налаштовувати вебсервер, встановлювати Python чи керувати інфраструктурою, що значно спрощує процес розгортання та робить його зрозумілим навіть для користувачів без досвіду роботи з DevOps інструментами. Середовище є доступним з будь-якого пристрою, оскільки є звичайним вебінтерфейсом. Значною перевагою можна вважати підтримку популярних Python фреймворків «з коробки», таких як Flask, Django, Bottle та інших, та великого переліку попередньо встановлених бібліотек. На вибір даної платформи також вплинула наявність безкоштовного тарифного плану, що дозволяє розгорнути вебсервіс на стандартному домені платформи без додаткової плати. [34]

Для початку потрібно перейти на сайт PythonAnywhere [34], зареєструватись або увійти в існуючий профіль користувача, після чого відкрита панель управління (рисунок 5.1).

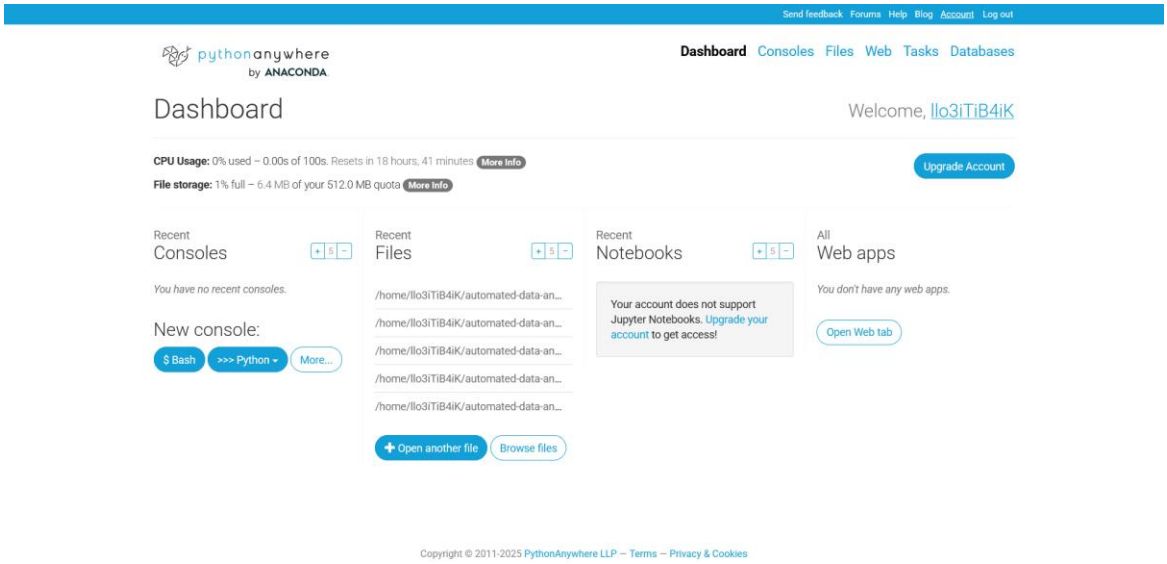


Рисунок 5.1 – Панель управління PythonAnywhere

Спершу потрібно налаштувати віртуальне середовище. Для цього переходимо до вкладки «Consoles» (рисунок 5.2), відкриваємо нову Bash консоль, натиснувши відповідну кнопку та дочекавшись її створення (рисунок 5.3). У новоствореній консолі потрібно виконати по черзі команди:

– mkvirtualenv myflaskenv --python=python3.10 ;

– git clone --branch master --single-branch https://github.com/llo3iTiB4iK/automated-data-analysis.git ;

– cd automated-data-analysis ;

– pip install --no-cache-dir -r requirements.txt – може потребувати повторного виконання в умовах обмеженого дискового простору (необхідний мінімум 500 МБ) через особливості встановлення пакетів.

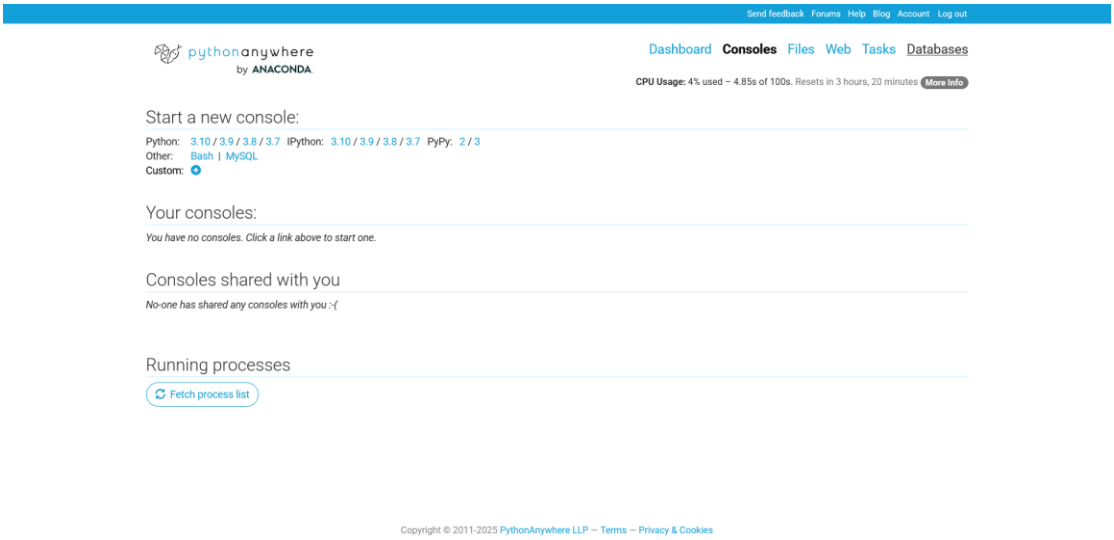


Рисунок 5.2 – Сторінка консолей PythonAnywhere

Рисунок 5.3 – Новостворена Bash консоль

Наступним кроком є створення та налаштування вебзастосунку. Для цього потрібно перейти до вкладки «Web» (рисунок 5.4) та створити новий вебзастосунок з доменом за замовчуванням, ручним налаштуванням середовища та версією інтерпретатора Python 3.10 (рисунок 5.5). Після вибору відповідних налаштувань потрібно дочекатись створення застосунку (рисунок 5.6). На сторінці новоствореного застосунку потрібно застосувати наступні значення для параметрів:

– source code: automated-data-analysis ;

– working directory: automated-data-analysis ;

– WSGI configuration file – змінити вміст файлу за посиланням та зберегти зміни (рисунок 5.7):

```
import sys
import os

path = '/home/<Ім’я користувача>/automated-data-analysis'
if path not in sys.path:
    sys.path.append(path)

os.environ['ENV'] = 'prod'

from main import app as application  # noqa
```

– virtualenv: myflaskenv ;

– force HTTPS: Enabled .

Після застосування наведених конфігурацій потрібно натиснути кнопку перезавантаження вебзастосунку:

– reload <Ім’я користувача>.pythonanywhere.com ;

Вигляд сторінки правильно налаштованого вебдодатку наведено на рисунку 5.8.

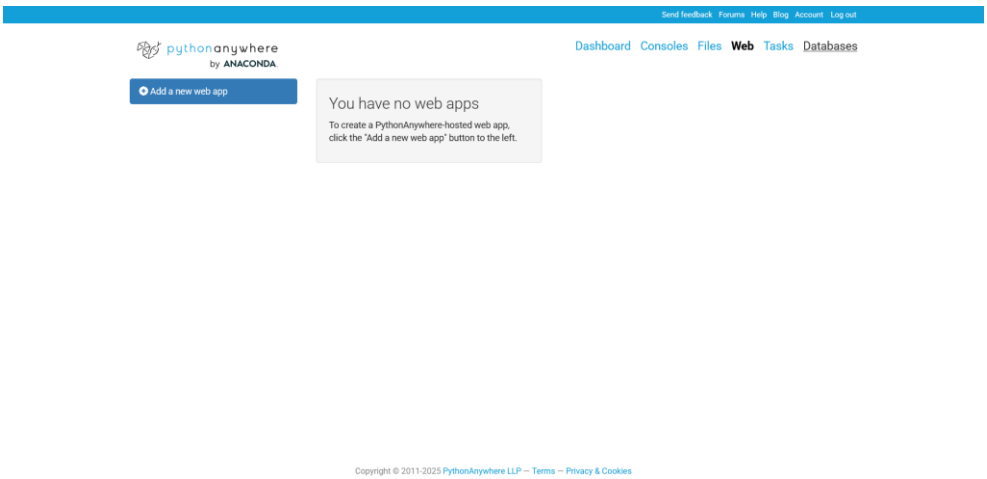


Рисунок 5.4 – Сторінка вебзастосунків PythonAnywhere

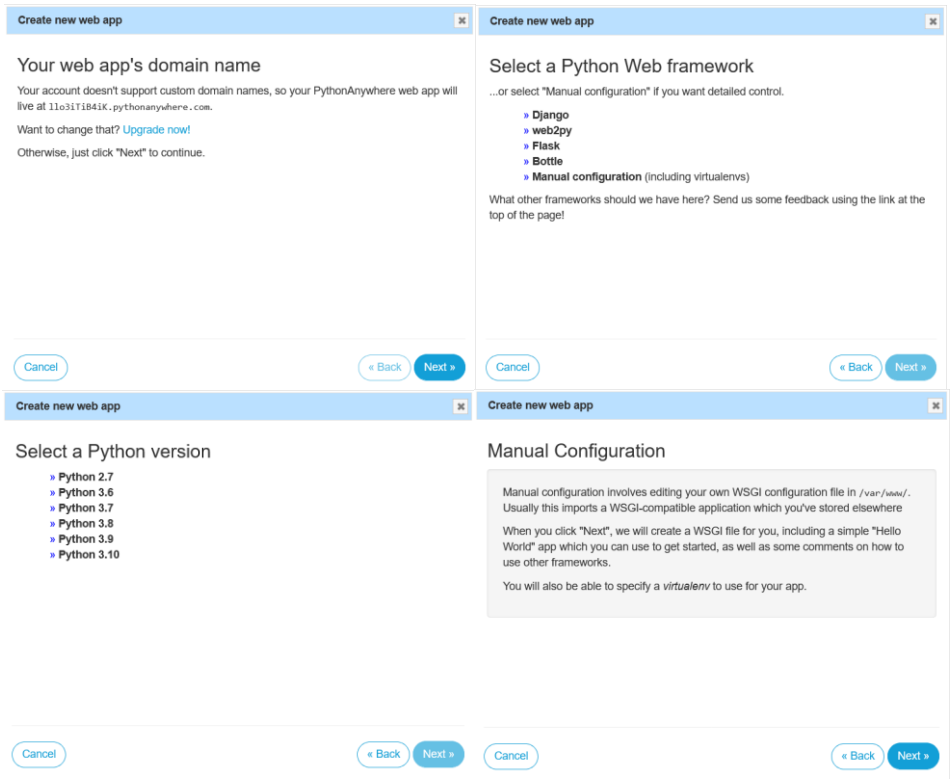


Рисунок 5.5 – Кроки створення вебзастосунку

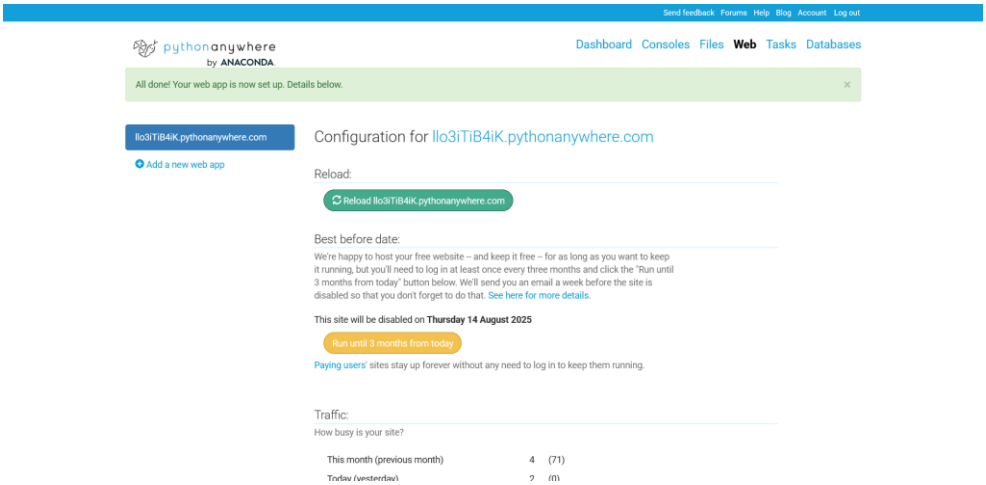


Рисунок 5.6 – Успішно створений вебзастосунок

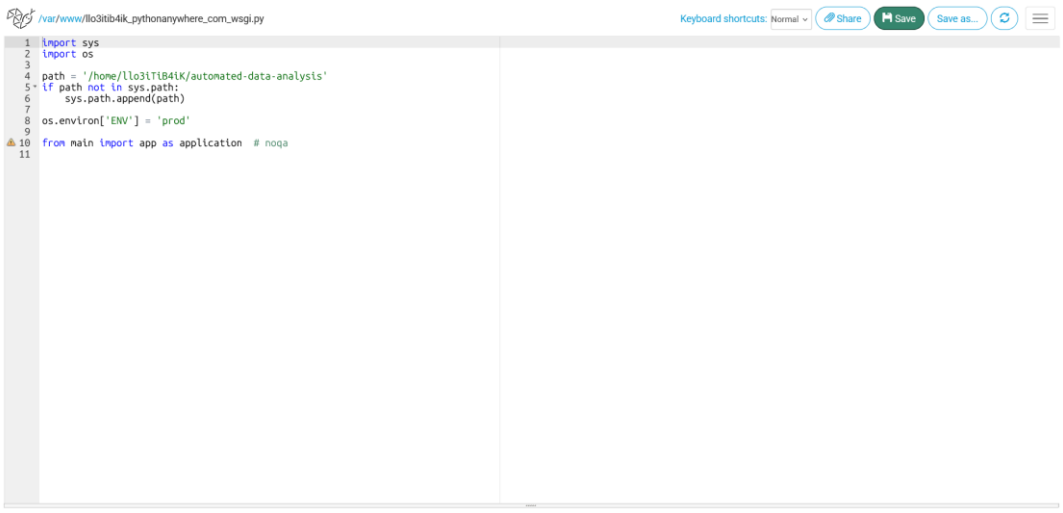


Рисунок 5.7 – Файл з налаштуваннями вебсервера

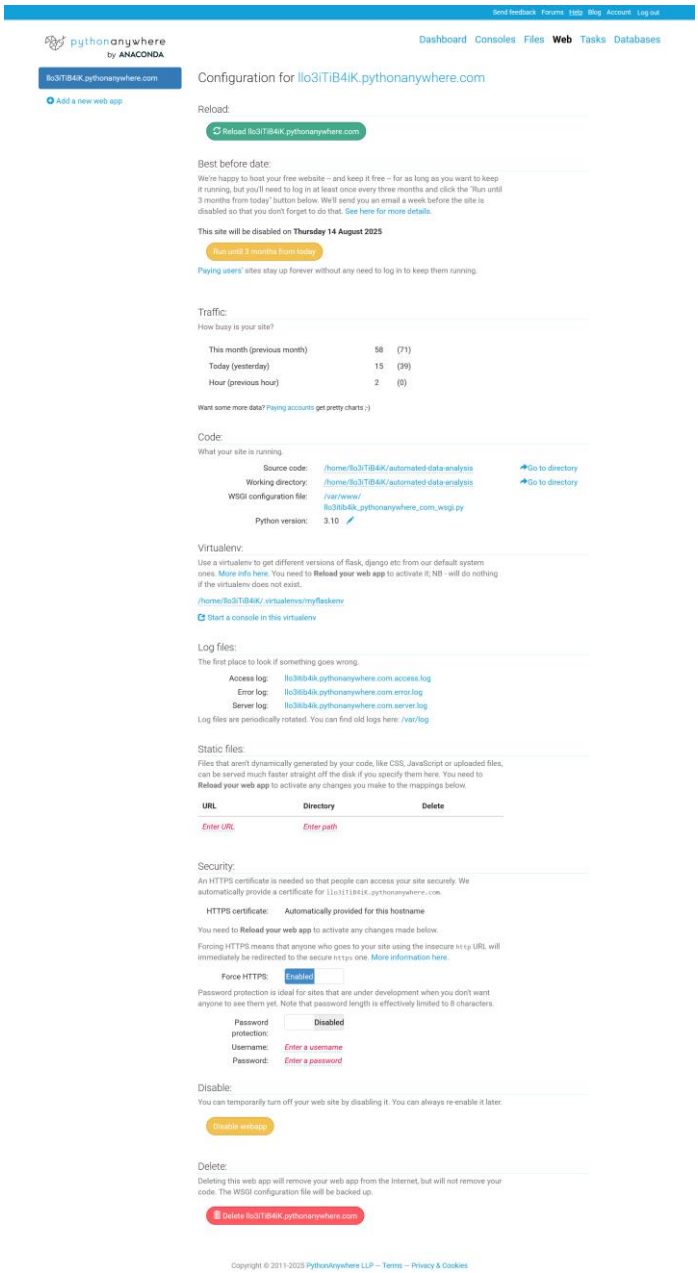


Рисунок 5.8 – Правильно налаштований вебзастосунок

Останнім кроком в налаштуванні роботи сервісу є налаштування періодичної очистки сховища. Для цього потрібно перейти до вкладки «Tasks» та у розділі «Scheduled tasks» створити нове заплановане завдання, яке щоденно в заданий час виконуватиме наступну команду (рисунок 5.9):

– workon myflaskenv && cd automated-data-analysis && flask cleanup

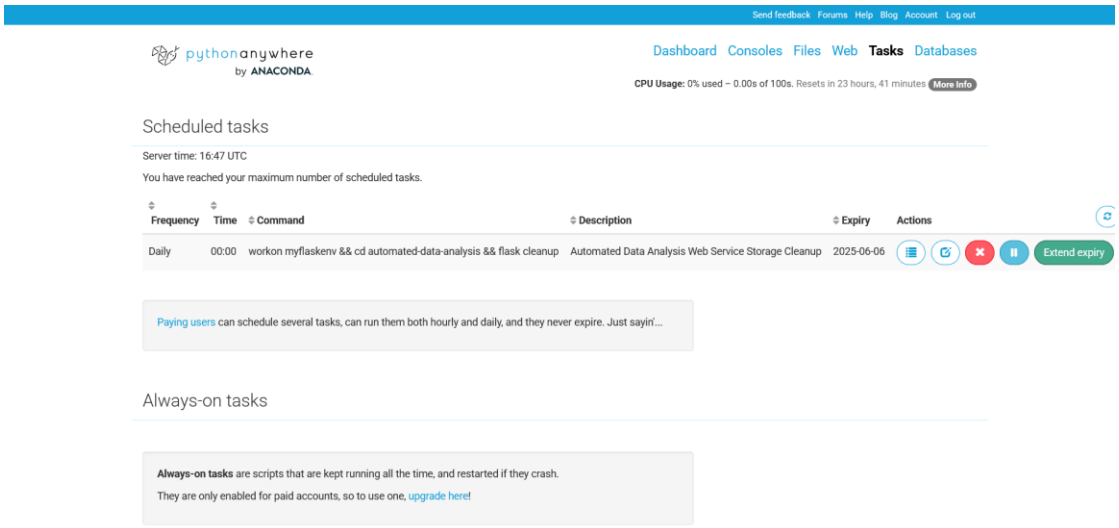


Рисунок 5.9 – Створене заплановане завдання очистки сховища

Після виконання усіх наведених кроків сервіс доступний по мережі за посиланням <Ім’я користувача>.pythonanywhere.com. За потреби, PythonAnywhere дозволяє використовувати власне доменне ім’я для користувачів платного тарифного плану.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво користувача».

Оскільки доступ до вебсервісу здійснюється через API, кінцевий користувач отримує останню версію одразу при наступному запиті до сервера без виконання додаткових дій.

Для оновлення коду серверної частини сервісу потрібно запушити зміни в гілку master віддаленого GitHub репозиторія. Далі потрібно застосувати ці зміни на вебсервері. Для цього здійснюється вхід в PythonAnywhere, на сторінці консолей (рисунок 5.2) створюється нова Bash консоль (рисунок 5.3) та виконується набір інструкцій:

cd automated-data-analysis

git pull

workon myflaskenv

pip install --no-cache-dir -r requirements.txt

touch /var/www/<Ім’я користувача>_pythonanywhere_com_wsgi.py

Після успішного виконання даного набору інструкцій (рисунок 5.10) оновлено код та відбувається автоматичний перезапуск вебсервісу.

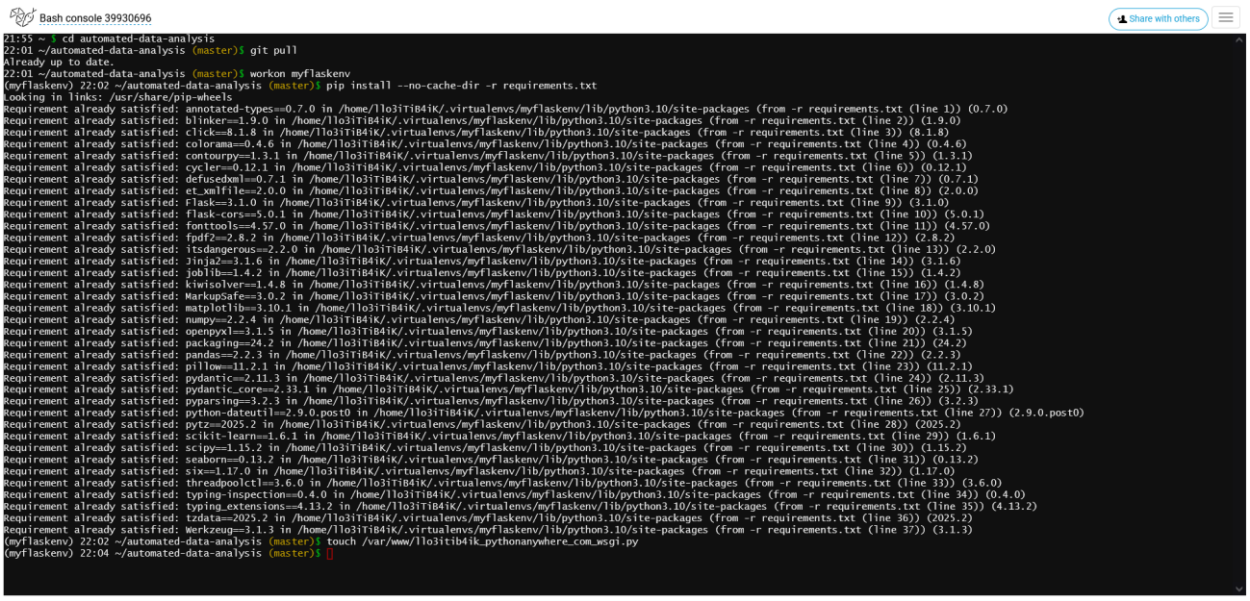


Рисунок 5.10 – Успішне оновлення вебсервісу

У разі переходу на платний тарифний план PythonAnywhere є можливість автоматизувати процес оновлення програмного забезпечення за допомогою GitHub Actions та PythonAnywhere SSH [34]. Таким чином, випуск нових версій ПЗ відбуватиметься при кожному пуші змін в master гілку GitHub репозиторія без потреби в ручному втручанні.

Висновки до розділу

У даному розділі було описано процеси випуску та оновлення вебсервісу для автоматизованого аналізу даних та генерації звітів.

Спершу було зроблено та обґрунтовано вибір хмарної платформи PythonAnywhere для хостингу розробленого вебсервісу та детально описано кроки розгортання в даному середовищі. Основними етапами розгортання є:

– створення та налаштування віртуального середовища для ізоляції залежностей вебсервісу;

– створення та повна конфігурація вебдодатку для правильної та безпечної роботи сервісу через мережу Інтернет;

– створення запланованого завдання для ефективного управління вільним місцем у сховищі незалежно від роботи вебсервера.

В наступному підрозділі було описано процес підтримки програмного забезпечення та отримання користувачем нових версій продукту. Власне супровід вебсервісу є досить простим та повністю задокументованим, але було описано потенційні шляхи його покращення.

Дані процеси були описані з огляду на використання безкоштовного тарифного плану PythonAnywhere. Однак, для забезпечення безвідмовної роботи сервісу, простоти його супроводу та найкращого користувацького досвіду варто розглянути вигідні платні плани, ціна яких коливається від 5 до 500 доларів на місяць в залежності від потреб [34].

ВИСНОВКИ

У ході виконання дипломного проєкту було розроблено вебзастосунок для автоматизованої підготовки табличних даних до аналізу, який надає користувачеві можливість швидко та ефективно виконати попередню обробку даних, отримати рекомендації щодо покращення їх якості та експортувати результати у зручному форматі.

Поставлену мету досягнуто у повному обсязі. Спростити та прискорити підготовку даних вдалося завдяки тому, що користувач більше не потребує самостійного налаштування середовища або опанування складних бібліотек чи програмних інструментів. Тепер достатньо просто передати дані та високорівневі інструкції з їх обробки для вебсервісу. Такий підхід значно зменшує ризики, пов’язані з людським фактором, а також робить процес доступним навіть для фахівців без глибоких знань у сфері аналізу даних. До того ж, для вказаної категорії користувачів передбачено рекомендації щодо покращення якості даних перед застосуванням до них алгоритмів машинного навчання.

Усі поставлені завдання розробки ПЗ успішно вирішено:

– реалізовано безпечний механізм обміну даними між користувачем і сервером без необхідності впровадження механізмів авторизації;

– забезпечено автоматизований процес попередньої обробки даних із використанням типових трансформацій;

– реалізовано модуль формування рекомендаційного звіту у форматі PDF з підказками щодо покращення якості даних і підвищення ефективності подальшого аналізу;

– створено систему управління сховищем із підтримкою автоматичного видалення застарілих наборів даних, а також передбачений сценарій користування сервісом у випадку переповнення серверного сховища даних;

– розроблено самодокументоване REST API з централізованою обробкою помилок у стандартизованому форматі.

Розроблене програмне рішення є завершеним, модульним та розширюваним. Застосунок не має зовнішніх залежностей (наприклад, сторонні API для забезпечення частини функціональності), підтримує централізовану конфігурацію його параметрів, а також може бути інтегроване як сервіс в іншу програмну систему.

У подальшому можливий розвиток системи в напрямках:

– створення графічного інтерфейсу для взаємодії через веббраузер;

– розширення переліку трансформацій та типових рекомендацій;

– підтримка додаткових форматів даних (зокрема, нетабличних) і сценаріїв аналізу;

– створення можливості вибору формату звіту;

– підтримка багатомовного інтерфейсу та звітів для залучення ширшої аудиторії користувачів;

– підтримка застосування алгоритмів машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Badman A., Kosinski M. What is data?. URL: https://www.ibm.com/think/topics/data (дата звернення: 15.04.2025).

2) Duarte F. Amount of data created daily (2025). URL: https://explodingtopics.com/blog/data-generated-per-day (дата звернення: 15.04.2025).

3) Pearson N. What is automated data analytics? (with examples). URL: https://www.zuar.com/blog/data-automation-improve-analysis-productivity/ (дата звернення: 16.04.2025).

4) Varshney H. Top 12 data wrangling tools in 2025. URL: https://hevodata.com/learn/data-wrangling-tools/ (дата звернення: 16.04.2025).

5) Alteryx. Get started with designer. URL: https://help.alteryx.com/2021.1/Getting_Started/GetStarted.htm (дата звернення: 16.04.2025).

6) Altair. Self-service data preparation solution | altair monarch. URL: https://altair.com/monarch/ (дата звернення: 16.04.2025).

7) Amazon Web Services, Inc. What is a web app?. URL: https://aws.amazon.com/what-is/web-application/ (дата звернення: 17.04.2025).

8) William. Web application architecture: the latest guide for 2025. URL: https://www.clickittech.com/software-development/web-application-architecture/ (дата звернення: 17.04.2025).

9) Nikita S. Web hosting vs. cloud hosting. URL: https://www.cloudpanel.io/blog/web-hosting-vs-cloud-hosting/ (дата звернення: 17.04.2025).

10) Object Management Group. Business process model and notation (BPMN). URL: https://www.omg.org/spec/BPMN/2.0/PDF (дата звернення: 18.04.2025).

11) Український SPEEDTEST. HTML5 тест швидкості Інтернет з'єднання. URL: https://speedtest.net.ua/ua (дата звернення: 29.04.2025).

12) Cohn M. Estimating with use case points. URL: https://www.mountaingoatsoftware.com/articles/estimating-with-use-case-points (дата звернення: 30.04.2025).

13) The C4 model for visualising software architecture. URL: https://c4model.com/ (дата звернення: 01.05.2025).

14) BrowserStack. 13 best languages for web development in 2025. URL: https://www.browserstack.com/guide/best-language-for-web-development (дата звернення: 02.05.2025).

15) Python Software Foundation. Python. URL: https://www.python.org/ (дата звернення: 03.05.2025).

16) Sigma Software University. Топ-5 кращих IDE для python розробки у 2024. Sigma Software University. URL: https://university.sigma.software/best-python-ide-and-code-editors/ (дата звернення: 02.05.2025).

17) Verbina E. Which is the best Python web framework: Django, Flask, or FastAPI? | The PyCharm Blog. URL: https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/ (дата звернення: 02.05.2025).

18) JetBrains. PyCharm: The only Python IDE you need. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 03.05.2025).

19) Postman: your complete API platform, from design to delivery. URL: https://www.postman.com/ (дата звернення: 03.05.2025).

20) Draw.io. Security-first diagramming for teams. URL: https://www.drawio.com/ (дата звернення: 03.05.2025).

21) Pallets. Flask documentation. URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 03.05.2025).

22) Pydantic. Pydantic. URL: https://docs.pydantic.dev/latest/ (дата звернення: 03.05.2025).

23) pandas. Python data analysis library. URL: https://pandas.pydata.org/ (дата звернення: 03.05.2025).

24) NumPy. The fundamental package for scientific computing with Python. URL: https://numpy.org/ (дата звернення: 03.05.2025).

25) Matplotlib. Visualization with Python. URL: https://matplotlib.org/ (дата звернення: 03.05.2025).

26) Seaborn. Statistical data visualization. URL: https://seaborn.pydata.org/ (дата звернення: 03.05.2025).

27) py-pdf. fpdf2. URL: https://py-pdf.github.io/fpdf2/ (дата звернення: 03.05.2025).

28) Scikit-learn. Machine Learning in Python. URL: https://scikit-learn.org/stable/ (дата звернення: 03.05.2025).

29) Nevil S. Z-Score: meaning and formula. URL: https://www.investopedia.com/terms/z/zscore.asp (дата звернення: 04.05.2025).

30) Agrawal A. Python pickling: what it is and how to use it securely. URL: https://www.blackduck.com/blog/python-pickling.html (дата звернення: 06.05.2025).

31) Dizdar A. Web application security: threats and 6 defensive methods. URL: https://www.brightsec.com/blog/web-application-security/ (дата звернення: 06.05.2025).

32) Sonar. SonarQube Cloud online code review as a service tool. URL: https://www.sonarsource.com/products/sonarcloud/ (дата звернення: 09.05.2025).

33) Zhang L. heart_2020_cleaned. Kaggle. URL: https://www.kaggle.com/datasets/luyezhang/heart-2020-cleaned (дата звернення: 11.05.2025).

34) PythonAnywhere. Host, run, and code Python in the cloud. URL: https://www.pythonanywhere.com/ (дата звернення: 14.05.2025).

35) Pallets. Werkzeug documentation. URL: https://werkzeug.palletsprojects.com/en/stable/ (дата звернення: 20.05.2025).

36) PyPI. Flask Pydantic Spec. URL: https://pypi.org/project/flask-pydantic-spec/ (дата звернення: 25.05.2025).

ДОДАТКИ

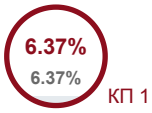
Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-11_Ляля_ПЗ
Автор Науковий керівник / Експерт
ЛяляЛіхоузова Т.А.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

10	12708	95420
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки		30
Парафрази (SmartMarks)	a	44

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП-з11_Босенко_ПЗ [redacted] 5/28/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	67 0.53 %
2	ІП-з11_Босенко_ПЗ [redacted] 5/28/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	48 0.38 %

3	Вебзастосунок для допомоги у діагностиці захворювань у професійній медичній практиці [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	32 0.25 %
4	ІП-13_Качмар_ПЗ [redacted] 5/29/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	26 0.20 %
5	ІП-15_Химич_ПЗ [redacted] 5/28/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	26 0.20 %
6	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	24 0.19 %
7	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	24 0.19 %
8	ІП-з11_Босенко_ПЗ [redacted] 5/28/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	22 0.17 %
9	Вебсервіс для онлайн-виставок картин [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	21 0.17 %
10	ІП-з11_Босенко_ПЗ [redacted] 5/28/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)	21 0.17 %

з домашньої бази даних (4.71 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП-з11_Босенко_ПЗ [redacted] 5/28/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**211 (9) 1.66 %**
2	ІП-13_Качмар_ПЗ [redacted] 5/29/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**59 (4) 0.46 %**
3	Вебзастосунок для допомоги у діагностиці захворювань у професійній медичній практиці [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**43 (3) 0.34 %**
4	Бібліотека для визначення контурів рельєфу з супутникових фотографій [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**41 (5) 0.32 %**
5	Вебсервіс для онлайн-виставок картин [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**33 (2) 0.26 %**

6	Вебсервіс для підтримки діяльності служби доставки [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**32 (3) 0.25 %**
7	Сервіс агрегації та аналізу даних про оренду житла [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**31 (3) 0.24 %**
8	ІП-15_Химич_ПЗ [redacted] 5/28/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**26 (1) 0.20 %**
9	ІП-11_Трикош_ПЗ [redacted] 5/28/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**24 (2) 0.19 %**
10	Вебзастосунок для ведення шкільного щоденника для учнів та викладачів [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**20 (1) 0.16 %**
11	Плагін браузера Google Chrome для автоматизації тестування вебзастосунків [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	17 (3) 0.13 %
12	Програмне забезпечення для виявлення аномалій на МРТ знімках [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**16 (2) 0.13 %**
13	Комп'ютера гра у жанрі Roguelike на рушії Godot [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**11 (1) 0.09 %**
14	ІП-13_Макарчук_ПЗ [redacted] 5/27/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-а інформатики та програмної інженерії)**	**11 (1) 0.09 %**
15	Мобільний застосунок для тайм-менеджменту [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	7 (1) 0.06 %
16	Програмне забезпечення реалізації можливостей Unreal Engine 5 для прикладних задач з використанням Oculus quest 2 [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	6 (1) 0.05 %
17	Веб-застосунок з надання та отримання послуг для власників домашніх тварин [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	5 (1) 0.04 %
18	Вебзастосунок для громадських організацій наукових спільнот [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	5 (1) 0.04 %

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (1.66 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download**	**71 (6) 0.56 %**
2	https://ela.kpi.ua/bitstream/123456789/29228/1/Diachenko_bakalavr.pdf	31 (5) 0.24 %
3	**https://ela.kpi.ua/bitstreams/a5cb95ec-11a8-40e1-9b06-d3455c267a2d/download**	**19 (2) 0.15 %**
4	**https://ela.kpi.ua/server/api/core/bitstreams/f463af88-756b-424a-bca4-3da434526966/content**	**18 (1) 0.14 %**
5	**https://ami.lnu.edu.ua/wp-content/uploads/2020/11/Sendziuk_MahRob_2020.pdf**	**17 (1) 0.13 %**
6	**https://www.pythonanywhere.com/forums/topic/35195/**	**16 (2) 0.13 %**
7	**http://tk-its.kpi.ua/sites/default/files/2020-07/Rachenchuk_bakalavr-%D0%BF%D0%B5%D1%80%D0%B5%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BE.pdf**	**15 (1) 0.12 %**
8	https://ela.kpi.ua/bitstream/123456789/63780/1/Kulyk_bakalavr.pdf	13 (2) 0.10 %
9	https://sci.ldubgd.edu.ua/jspui/bitstream/123456789/14400/1/%D0%9A%D1%83%D0%BF%D1%80%D1%96%D0%BA%D0%BE%D0%B2.pdf	11 (2) 0.09 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБСЕРВІС ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ДАНИХ ТА ГЕНЕРАЦІЇ ЗВІТІВ

Текст програми

КПІ.ІП-1118.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Іван ЛЯЛЯ

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
https://github.com/llo3iTiB4iK/automated-data-analysis

Файл app/controllers/dataframe_loader.py
Реалізація функціональної задачі обміну даними між користувачем та сервером.

```
import os
import sqlite3
import tempfile
from io import BytesIO

import pandas as pd
from werkzeug.datastructures.file_storage import FileStorage

from app.errors import EmptyDataset, ReadingError
from app.models import LoadingParams


class DataFrameLoader:

    def __init__(self, file: FileStorage, params: LoadingParams) -> None:
        self.file = file
        self.params = params

    def __error(self, desc: str) -> ReadingError:
        raise ReadingError(self.file.filename, desc)

    def _load_csv(self) -> pd.DataFrame:
        return pd.read_csv(self.file.stream, sep=self.params.separator, thousands=self.params.thousands,
                           decimal=self.params.decimal)

    def _load_excel(self) -> pd.DataFrame:
        sheet_name = self.params.sheet_name
        xls = pd.ExcelFile(BytesIO(self.file.stream.read()))
        available = xls.sheet_names
        if isinstance(sheet_name, str) and sheet_name not in available:
            raise self.__error(f"Sheet '{sheet_name}' not found in Excel file. Available sheets: {available}")

        return xls.parse(sheet_name=sheet_name, thousands=self.params.thousands, decimal=self.params.decimal)

    def _load_json(self) -> pd.DataFrame:
        return pd.read_json(self.file.stream)

    def _load_sqlite(self) -> pd.DataFrame:
        table_name = self.params.table_name

        with tempfile.NamedTemporaryFile(delete=False, suffix=".db") as temp_file:
```

```
            temp_file.write(self.file.read())
            temp_file_path = temp_file.name

        cnx = sqlite3.connect(temp_file_path)
        try:
            return pd.read_sql(f"SELECT * FROM {table_name}", cnx)
        except pd.errors.DatabaseError:
            cursor = cnx.execute("SELECT name FROM sqlite_master WHERE type='table';")
            available = [record[0] for record in cursor.fetchall()]
            raise self.__error(f"Table '{table_name}' not found in the SQLite file. Available tables: {available}'")
        finally:
            cnx.close()
            os.remove(temp_file_path)

    def load_data(self) -> pd.DataFrame:
        LOADERS = {
            'csv': self._load_csv,
            'xls': self._load_excel,
            'xlsx': self._load_excel,
            'json': self._load_json,
            'db': self._load_sqlite
        }

        extension = self.file.filename.split('.')[-1].lower()

        if extension not in LOADERS:
            raise self.__error(f"Unsupported file extension '{extension}'. Supported types: {list(LOADERS.keys())}")

        try:
            data = LOADERS[extension]()
        except ReadingError:
            raise
        except Exception:
            raise self.__error("The data may be corrupted or incorrectly formatted. "
                               "Please check the file and try again later.")

        if data.empty:
            raise EmptyDataset(f"The uploaded file '{self.file.filename}' contains no data")

        return data
```

Файл app/data_exchange/routes.py

Реалізація функціональної задачі обміну даними між користувачем та сервером.

```
from io import BytesIO


from flask import request, url_for, send_file, jsonify

from flask_pydantic_spec import Response, FileResponse, MultipartFormRequest
```

```
from app.data_exchange import bp
from app.errors import ParameterMissing
from app.extensions import storage, spec
from app.controllers import DataFrameLoader, DataFrameAnalyzer
from app.models import LoadingParams, UploadResponse, DatasetTokenHeader, InfoResponse,
ExportParams


@bp.route("/datasets", methods=["POST"])
@spec.validate(
    body=MultipartFormRequest(model=LoadingParams),
    resp=Response(HTTP_200=UploadResponse),
    tags=["Upload dataset"]
)
def upload_file() -> Response:
    file = request.files.get('file')
    if file is None or file.filename == '':
        raise ParameterMissing("file")

    params = request.context.body  # noqa

    data = DataFrameLoader(file, params).load_data()
    dataset_id, access_key = storage.save_dataset(data)

    response_data = UploadResponse(
        message="Dataset uploaded successfully",
        dataset_id=dataset_id,
        access_key=access_key,
        next_step=url_for("preprocessing.preprocess_dataset", dataset_id=dataset_id),
        metadata=DataFrameAnalyzer.get_metadata(data)
    )

    return jsonify(response_data.dict())
```

```
@bp.route("/datasets/<dataset_id>")
@spec.validate(
    headers=DatasetTokenHeader,
    resp=Response(HTTP_200=InfoResponse),
    tags=["Dataset info"]
)
def get_info(dataset_id: str) -> Response:
    data = storage.get_dataset(dataset_id)

    response_data = InfoResponse(
        message="Dataset found successfully",
        dataset_id=dataset_id,
        next_step=url_for("preprocessing.preprocess_dataset", dataset_id=dataset_id),
        metadata=DataFrameAnalyzer.get_metadata(data)
    )

    return jsonify(response_data.dict())



@bp.route("/datasets/<dataset_id>/download")
@spec.validate(
    query=ExportParams,
    headers=DatasetTokenHeader,
    resp=FileResponse(),
    tags=["Download dataset"]
)
def download_dataset(dataset_id: str) -> FileResponse:
    params: ExportParams = request.context.query  # noqa
    data = storage.get_dataset(dataset_id)

    buffer = BytesIO()
    params.method(data, buffer)
    buffer.seek(0)
```

```
    return send_file(buffer, mimetype=params.mimetype, as_attachment=False,
download_name=f'{dataset_id}.{params.ext}')
```

Файл app/controllers/dataframe_preprocessor.py
Реалізація функціональної задачі попередньої обробки даних.

```
import string
from typing import Any, Callable, List

import numpy as np
import pandas as pd

from app.errors import EmptyDataset, ColumnNotFound, TransformationError
from app.models.request.preprocessing_params import ColumnList, PreprocessingParams



class DataFramePreprocessor:

    def __init__(self, data: pd.DataFrame) -> None:
        self.data = data

    def preprocess(self, params: PreprocessingParams) -> pd.DataFrame:
        """
        Execute preprocessing steps according to given parameters.
        """
        steps = [
            (params.case_insensitive_columns, self._lowercase_columns),
            (params.clear_punct_columns, self._remove_punctuation),
            (params.clear_digits_columns, self._remove_digits),
            (any([params.row_range_start, params.row_range_end, params.row_range_step]),
self._select_rows),
            (params.index_cols, self._set_index),
            (params.fill_na_values is not None, self._fill_missing_values),
            (params.mfill, self._fill_missing_with_median_mode),
            (params.ffill, self._forward_fill),
            (params.bfill, self._backward_fill),
            (params.drop_na, self._drop_na),
```

```
            (params.drop_outliers, self._drop_outliers),
            (params.drop_duplicates, self._drop_duplicates),
            (params.datetime_columns, self._convert_datetime),
            (params.category_columns, self._convert_category),
            (params.join_small_cat, self._combine_rare),
            (params.scale_numeric, self._scale_numeric),
        ]

        for condition, action in steps:
            if condition:
                action(params)

        return self.data

    def _resolve_columns(self, cols: ColumnList) -> List[str]:
        if cols == "*":
            return list(self.data.columns)
        elif isinstance(cols, str):
            cols = [cols]

        missing = [c for c in cols if c not in self.data.columns]
        if missing:
            raise ColumnNotFound(missing, list(self.data.columns))

        return list(cols)

    def _ensure_not_empty(self, operation: str) -> None:
        if self.data.empty:
            raise EmptyDataset(f"{operation.title()} resulted in an empty dataset. Review respective
request params.")

    # ========== String operations ==========
    def _lowercase_columns(self, params: PreprocessingParams) -> None:
        self._apply_str_op(params.case_insensitive_columns, lambda s: s.lower(), "Lowercasing")
```

```
    def _remove_punctuation(self, params: PreprocessingParams) -> None:
        self._apply_str_op(
            params.clear_punct_columns,
            lambda s: s.translate(str.maketrans('', '', string.punctuation)),
            "Punctuation Removal"
        )

    def _remove_digits(self, params: PreprocessingParams) -> None:
        self._apply_str_op(
            params.clear_digits_columns,
            lambda s: s.translate(str.maketrans('', '', string.digits)),
            "Digits Removal",
        )

    def _apply_str_op(self, cols: ColumnList, func: Callable[[Any], Any], operation: str, el_wise:
bool = True) -> None:
        columns = self._resolve_columns(cols)

        def elem_func(x: Any) -> Any:
            return func(x) if pd.notna(x) else x

        for col in columns:
            try:
                self.data[col] = self.data[col].apply(elem_func) if el_wise else func(self.data[col])
            except Exception:
                raise TransformationError(operation, col)

    # ========== Row/Index operations ==========
    def _select_rows(self, params: PreprocessingParams) -> None:
        start = (params.row_range_start or 1) - 1
        stop = params.row_range_end
        step = params.row_range_step
        self.data = self.data.iloc[start:stop:step]
        self._ensure_not_empty("row selection")
```

```
def _set_index(self, params: PreprocessingParams) -> None:
    cols = self._resolve_columns(params.index_cols)
    self.data.set_index(cols, inplace=True)
    self._ensure_not_empty("index setting")

# ========== Missing value operations ==========
def _fill_missing_values(self, params: PreprocessingParams) -> None:
    try:
        self.data.fillna(params.fill_na_values, inplace=True)
    except Exception:
        raise TransformationError("Filling missing values", "*")

def _fill_missing_with_median_mode(self, _: PreprocessingParams) -> None:
    stats = {}
    for col in self.data.columns:
        if self.data[col].isna().any():
            stats[col] = (
                self.data[col].median()
                if pd.api.types.is_numeric_dtype(self.data[col])
                else self.data[col].mode().iloc[0]
            )
    self.data.fillna(stats, inplace=True)

def _forward_fill(self, _: PreprocessingParams) -> None:
    self.data.ffill(inplace=True)

def _backward_fill(self, _: PreprocessingParams) -> None:
    self.data.bfill(inplace=True)

def _drop_na(self, params: PreprocessingParams) -> None:
    self.data.dropna(axis=params.drop_na, inplace=True)
    self._ensure_not_empty("dropping missing values")

# ========== Outliers & duplicates ==========
def _drop_outliers(self, params: PreprocessingParams) -> None:
```

```
        num = self.data.select_dtypes(include='number')
        z: pd.DataFrame = np.abs((num - num.mean()) / num.std())
        mask = (z > params.outliers_threshold).any(axis=1)
        self.data = self.data.loc[~mask]
        self._ensure_not_empty("dropping outliers")

    def _drop_duplicates(self, params: PreprocessingParams) -> None:
        subset = None if not params.duplicate_subset else
self._resolve_columns(params.duplicate_subset)
        self.data.drop_duplicates(subset=subset, keep=params.duplicate_keep, inplace=True)
        self._ensure_not_empty("dropping duplicates")

    # ========== Type conversions ==========
    def _convert_datetime(self, params: PreprocessingParams) -> None:
        self._apply_str_op(params.datetime_columns, pd.to_datetime, "Datetime conversion")

    def _convert_category(self, params: PreprocessingParams) -> None:
        self._apply_str_op(params.category_columns, lambda col: col.astype('category'), "Category
conversion", False)

    # ========== Category merging ==========
    def _combine_rare(self, params: PreprocessingParams) -> None:
        for col in self.data.select_dtypes(include='category').columns:
            counts = self.data[col].value_counts(normalize=True)
            threshold = params.categories_threshold or counts.quantile(0.2)
            rare_ = counts[counts <= threshold].index
            self.data[col] = self.data[col].apply(lambda x, rare=rare_: params.joined_category_name
if x in rare else x)

    # ========== Scaling ==========
    def _scale_numeric(self, params: PreprocessingParams) -> None:
        cols = self.data.select_dtypes(include='number').columns
        if cols.empty:
            return
        self.data[cols] = params.scaler.fit_transform(self.data[cols])
```

Файл app/preprocessing/routes.py

Реалізація функціональної задачі попередньої обробки даних.

```
from flask import request, url_for, jsonify
from flask_pydantic_spec import Response

from app.controllers import DataFramePreprocessor, DataFrameAnalyzer
from app.extensions import storage, spec
from app.models import PreprocessingParams, PreprocessingResponse, DatasetTokenHeader
from app.preprocessing import bp



@bp.route("/datasets/<dataset_id>/preprocess", methods=["POST"])
@spec.validate(
    body=PreprocessingParams,
    headers=DatasetTokenHeader,
    resp=Response(HTTP_200=PreprocessingResponse),
    tags=["Preprocessing"]
)
def preprocess_dataset(dataset_id: str) -> Response:
    params: PreprocessingParams = request.context.body  # noqa
    data = storage.get_dataset(dataset_id)

    preprocessor = DataFramePreprocessor(data)
    data = preprocessor.preprocess(params)

    new_dataset_id, new_access_key = storage.save_dataset(data, None if params.make_copy else
dataset_id)

    response_data = PreprocessingResponse(
        message="Dataset preprocessed successfully",
        dataset_id=dataset_id,
        next_step=url_for("reporting.get_recommendations", dataset_id=dataset_id),
        metadata=DataFrameAnalyzer.get_metadata(data),
        new_dataset_id=new_dataset_id if params.make_copy else None,
        new_dataset_access_key=new_access_key if params.make_copy else None
    )
```

```
    return jsonify(response_data.dict(exclude_none=True))
```

Файл app/controllers/dataframe_analyzer.py

Реалізація функціональної задачі генерації рекомендаційного звіту.

```
import sys
from contextlib import nullcontext
from dataclasses import dataclass
from typing import Callable

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.feature_selection import mutual_info_classif

from app.errors import ColumnNotFound
from app.models import MetadataResponse
from app.models.request.analysis_params import AnalysisParams, AnalysisTask,
DocumentTheme
from .dataframe_report import DataFrameReport

sns.set_style("darkgrid")



class DataFrameAnalyzer:

    @dataclass(frozen=True)
    class FeatureSelectionParams:
        metrics: pd.Series
        levels: dict[str, tuple[float, float]]
        name: str
        task: str
        plot_func: Callable[..., plt.Axes]
        plot_feature_wise: bool = True
```

```
    def __init__(self, data: pd.DataFrame) -> None:
        self._data = data
        self._report = None
        self._include_visualizations = True

    def __validate_target(self, col: str) -> pd.Series:
        if col not in self._data:
            raise ColumnNotFound([col], list(self._data.columns))
        series = self._data[col]
        missing = series.isna().sum()
        if missing:
            pct = round(missing / len(series) * 100, 2)
            self._report.add_text(
                f"* Target column '{col}' has {missing} missing values ({pct}% of all):\n"
                f"    - if missing values share is not too significant - "
                f"consider removal or using median/mode/mean value for imputation.\n"
                f"    - else - consider special methods of imputation "
                f"(Forward Fill, Backward Fill, using exact value or based on other columns)."
            )
        else:
            self._report.add_text(f"* Target column '{col}' has no missing values.")
        return series

    def __select_features(self, params: FeatureSelectionParams, target: str) -> None:
        self._report.add_heading("Feature Selection Recommendations:")
        significant_features = 0
        for label, (low, high) in params.levels.items():
            group = params.metrics[(params.metrics.abs() >= low) & (params.metrics.abs() < high)]
            if group.empty:
                self._report.add_text(f"* No {label} meaningful features found based on
{params.name}.")
                continue
            significant_features += len(group)
            self._report.add_text(f"* {len(group)} {label} meaningful features were found. Consider
using them in {params.task}:")
```

```
            # Visual summary
            if 3 <= len(group) <= 7 and self._include_visualizations:
                sns.heatmap(pd.DataFrame(group).T, annot=True, square=True, cbar=False,
vmin=low, vmax=high, cmap='coolwarm', linewidth=.5)
                self._report.add_plot(title=params.name.title())
            else:
                self._report.add_series(group)
            # Pairwise plots
            if self._include_visualizations:
                if params.plot_feature_wise:
                    plotters = [lambda ax, f=feat: params.plot_func(ax, f) for feat in group.index]
                    self._report.add_subplots(plotters, suptitle=f"Dependency between '{target}' and
{label} meaningful features")
                else:
                    params.plot_func(group.index)
                    self._report.add_plot(title=f"{label.title()} meaningful features chart")
        # Note on rest
        rest = params.metrics[params.metrics.abs() < min(l[0] for l in params.levels.values())]
        if significant_features == 0:
            self._report.add_text(
                f"* None of the features show significant relationship with target '{target}'.\n"
                f"- You might want to collect more informative features, engineer new ones, or
reassess data quality."
            )
        elif not rest.empty:
            self._report.add_text("* Remaining features have low relevance.")

    def __regression_recs(self, target: str) -> FeatureSelectionParams | None:
        y = self.__validate_target(target)
        if pd.api.types.is_numeric_dtype(y) and not pd.api.types.is_bool_dtype(y):
            self._report.add_text(f"* Target column '{target}' is numeric ({y.dtype}).")
        else:
            self._report.add_text(f"* Target column '{target}' is not numeric.\n"
                                  f"* No further reporting can be performed. Consider encoding or
converting it.")
```

```
            return

        if self._include_visualizations:
            self._report.add_subplots([
                lambda ax: sns.boxplot(x=y, ax=ax),
                lambda ax: sns.histplot(y, kde=True, ax=ax).lines[0].set_color('crimson')
            ], suptitle=f"'{target}' values distribution")
        # Outliers
        q1, q3 = y.quantile([0.25, 0.75])
        iqr = q3 - q1
        out = ((y < q1 - 1.5 * iqr) | (y > q3 + 1.5 * iqr)).sum()
        if out:
            self._report.add_text(f"* {out} potential outliers detected in '{target}'.\n"
                                  f"* Consider handling them or leave these values as-is if they are
important.\n")
        else:
            self._report.add_text("* No potential outliers were detected in target column, which is
perfect for building a "
                                  "stable predictive model and indicates good data quality!\n")
        self._report.add_text(f"* If the distribution of '{target}' is not normal (look at the chart
above), consider "
                              f"applying transformations such as log or Box-Cox to make the data more
suitable for reporting.\n"
                              f"A transformation can sometimes help stabilize variance and improve the
model's performance.")
        # Correlation

        def plot_corr(ax: plt.Axes, feature: str) -> plt.Axes:
            sns.regplot(x=y, y=self._data[feature], line_kws={"color": "orange"}, ax=ax)

        return self.FeatureSelectionParams(
            metrics=self._data.corr(numeric_only=True)[target],
            levels={'highly': (0.7, 1.0), 'moderately': (0.5, 0.7), 'low': (0.3, 0.5)},
            name='correlation',
            task='regression',
```

```
            plot_func=plot_corr
        )

    def __classification_recs(self, target: str) -> FeatureSelectionParams | None:
        y = self.__validate_target(target)
        if pd.api.types.is_bool_dtype(y) or y.dtype == 'category' or (y.dtype == 'object' and
y.nunique(True) <= 10):
            self._report.add_text(f"* Target column '{target}' seems to be discrete ({y.dtype}).")
            y = y.astype('category')
        else:
            self._report.add_text(f"* Target column '{target}' seems to be continuous or containing
raw text data.\n"
                                  f"- No classification analysis can be performed.\n* Number of unique
values: {y.nunique(True)}.")
            return

        if self._include_visualizations:
            sns.countplot(x=y, hue=y, legend=False)
            self._report.add_plot(f"'{target}' class distribution")
        # Class balance
        counts = y.value_counts(normalize=True)
        min_class, max_class = counts.min(), counts.max()
        if max_class / min_class > 3:
            self._report.add_text(
                f"* Target column is imbalanced:\n"
                f"    - the most frequent class appears {round(max_class / min_class, 2)}x more often
than the least frequent.\n"
                f"    - consider using techniques like oversampling (SMOTE), undersampling, or class
weighting in your model.")
        else:
            self._report.add_text("* Target column has a balanced distribution of classes.")
        # Rare categories
        rare_categories = counts[counts < 0.01]
        if not rare_categories.empty:
            self._report.add_text("* Some target classes are very rare (<1% of total data):")
```

```
            self._report.add_series(rare_categories)
            self._report.add_text("* Consider:\n"
                                  "   - grouping rare classes into an 'Other' category (if appropriate)\n"
                                  "   - collecting more data\n"
                                  "   - using stratified sampling during training.")
        # Mutual info

        def plot_mi(ax: plt.Axes, feature: str) -> plt.Axes:
            sns.boxplot(x=self._data[feature], y=y, hue=y, ax=ax, legend=False)

        nums = self._data.select_dtypes('number')
        if nums.shape[1] == 0:
            return

        valid_idx = pd.concat([nums, y], axis=1).dropna().index

        return self.FeatureSelectionParams(
            metrics=pd.Series(mutual_info_classif(nums.loc[valid_idx], y.loc[valid_idx],
random_state=42), index=nums.columns, name=target),
            levels={'highly': (0.1, 1.0), 'moderately': (0.05, 0.1), 'low': (0.01, 0.05)},
            name='mutual information',
            task='classification',
            plot_func=plot_mi
        )

    def __clustering_recs(self, target: str) -> FeatureSelectionParams | None:
        self._data.loc[:, target] = None
        n = len(self._data)
        if n < 100:
            self._report.add_text(f"* Small data ({n} rows): clustering may be unreliable.")
        else:
            self._report.add_text(
                f"* Considering your dataset size ({len(self._data)} rows), expected number of clusters
should not be"
```

```
            f" more than {n // 10}.\n    - Otherwise, clustering algorithms would have low performance.")
        nums = self._data.select_dtypes('number').dropna()
        if nums.shape[1] > 3:
            self._report.add_text(
                f"\n* The dataset consists of {nums.shape[1]} numeric columns:\n    - To facilitate clustering "
                f"and improve visualization, dimensionality reduction techniques like PCA or t-SNE should be applied."
            )
        if nums.shape[1] == 0:
            return
        # Weighted PCA score
        pca = PCA(random_state=42)
        pca.fit_transform(nums)
        weighted_pca = abs(pca.components_).T.dot(pca.explained_variance_ratio_)
        imp = pd.Series(weighted_pca, index=nums.columns, name=target).sort_values(ascending=False)
        self._report.add_text(
            "\n* In the next section, looking at the feature distribution chart, consider preprocessing decisions:\n"
            "    1) whether scaling should be applied, as most clustering algorithms are distance-based;\n"
            "        - actually, scaling can result in a completely different set of important features.\n"
            "    2) whether outliers should be handled properly, as they can significantly impact the results;\n"
            "        - this operation can also significantly impact the feature importance.")
        return self.FeatureSelectionParams(
            metrics=imp,
            levels={'highly': (0.01 * imp.sum(), imp.sum() + sys.float_info.epsilon)},
            name='weighted PCA score',
            task='clustering',
            plot_func=lambda features: sns.boxplot(self._data[features], orient='h'),
            plot_feature_wise=False
        )
```

```
    def __feature_engineering(self, target: str) -> None:
        self._report.add_heading("Feature Engineering Recommendations:")
        dtypes = self._data.drop(columns=[target]).dtypes
        recs_given = False

        strategies = {
            'bool': "Encode boolean features as 0/1 if needed:",
            'int': "Bin or treat small-cardinality integer features (having few unique values) as
categorical:",
            'float': "Transform highly skewed (|skewness| > 1) float features (log, sqrt, Box-Cox):",
            'datetime64[ns]': "Extract useful date parts (year, month, day, weekday etc) from
datetime features:",
            'category': "Encode category features using One-Hot, Target, Frequency or Ordinal
Encoding methods."
                        "Group rare categories to avoid sparsity:",
            'object': "Convert string features to categorical ones, apply text transformations or drop:"
        }
        for dt, msg in strategies.items():
            cols = dtypes[dtypes == dt].index.tolist()
            if not cols:
                continue
            if dt == 'int':
                cols = [c for c in cols if (self._data[c].nunique() <= 20)]
            if dt == 'float':
                cols = [c for c in cols if abs(self._data[c].skew()) > 1]
            if cols:
                recs_given = True
                self._report.add_text(f"* {msg}")
                self._report.add_series(cols)

        if not recs_given:
            self._report.add_text("* All the features seem to be prepared for further analysis.")

    def _basic_stats(self) -> None:
```

```
        df = self._data
        self._report.add_heading("Overall dataset summary:")

        # Basic counts
        missing_cells = df.isna().sum().sum()
        counts = {
            'rows': len(df),
            'columns': len(df.columns),
            'numeric': df.select_dtypes('number').shape[1],
            'categorical': df.select_dtypes('category').shape[1],
            'boolean': df.select_dtypes('bool').shape[1],
            'datetime': df.select_dtypes('datetime').shape[1],
            'string': df.select_dtypes('object').shape[1],
            'duplicates': int(df.duplicated().any()),
            'missing_pct': round(missing_cells / df.size * 100, 2),
        }
        summary = f"* Dataset contains {counts['rows']} rows, {counts['columns']} columns\n" \
                  f"({counts['numeric']} numeric, {counts['categorical']} categorical,
{counts['boolean']} boolean, " \
                  f"{counts['datetime']} datetime, {counts['string']} string).\n" \
                  f"* Duplicated rows {'found' if counts['duplicates'] else 'not found'}.\n" \
                  f"* Missing values: {counts['missing_pct']}% ."
        self._report.add_text(summary)

        # Data types series
        self._report.add_series(self._data.dtypes, title="Column Types:")

        # Missing value plot
        if missing_cells > 0 and self._include_visualizations:
            missing_df = pd.DataFrame({
                'Missing': df.isna().sum(),
                'Non-missing': df.notna().sum()
            })
            missing_df.plot(kind='barh', stacked=True, title="Missing values by column",
xlabel="Count")
```

```
            self._report.add_plot()

        # Descriptive statistics
        num_desc = df.describe()

        for col in df.select_dtypes(include='datetime').columns:
            num_desc[col] = num_desc[col].astype('datetime64[s]')

        self._report.add_dataframe(num_desc, title="Numeric Stats:")
        category_columns = df.select_dtypes(include=['object', 'category', 'bool'])
        if not category_columns.empty:
            self._report.add_dataframe(category_columns.describe(), title="Non-numeric Stats:")

    def _task_based_recs(self, analysis_task: AnalysisTask, target_column: str) -> None:
        TASK_RECOMMENDERS: dict[str, Callable[..., None]] = {
            AnalysisTask.REGRESSION: self.__regression_recs,
            AnalysisTask.CLASSIFICATION: self.__classification_recs,
            AnalysisTask.CLUSTERIZATION: self.__clustering_recs
        }
        self._report.add_heading(f"{analysis_task.title()} Recommendations for
'{target_column}'")
        selection_params = TASK_RECOMMENDERS[analysis_task](target_column)
        if selection_params:
            self.__select_features(selection_params, target_column)
            self.__feature_engineering(target_column)
        self._report.add_text(f"\n|====<  {analysis_task.title()} preparation completed !  >====|",
monospaced=True, style="B")

    def generate_report(self, params: AnalysisParams) -> DataFrameReport:
        with plt.style.context('dark_background') if params.theme == DocumentTheme.DARK else
nullcontext():
            self._report = DataFrameReport(dpi=params.dpi, theme=params.theme,
show_time=params.show_time)
            self._include_visualizations = params.include_visualizations
            if params.include_basic_stats:
```

```
            self._basic_stats()
        self._task_based_recs(params.analysis_task, params.target_col)
        return self._report

    @staticmethod
    def get_metadata(data: pd.DataFrame) -> MetadataResponse:
        return MetadataResponse(
            num_rows=len(data),
            num_columns=len(data.columns),
            columns=data.dtypes.astype(str).to_dict()
        )
```

Файл app/controllers/dataframe_report.py

Реалізація функціональної задачі генерації рекомендаційного звіту.

```
from datetime import datetime, timezone
from io import BytesIO

import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from fpdf import FPDF  # noqa

from app.models.request.analysis_params import DocumentTheme

matplotlib.use('Agg')
pd.set_option('display.precision', 4)



class DataFrameReport(FPDF):

    def __init__(self, dpi: int = 200, theme: DocumentTheme = DocumentTheme.LIGHT,
show_time: bool = True) -> None:
        super().__init__()
        self._dpi = dpi
        self._show_time = show_time
        self.create_time = datetime.now(timezone.utc).strftime("%Y-%m-%d %H:%M:%S UTC")
```

```
        if theme == DocumentTheme.DARK:
            self.set_page_background((0, 0, 0))
            self.set_text_color(255, 255, 255)
            self.set_draw_color(255, 255, 255)

        self.add_font("Monospace-Unicode", style="", fname="fonts/MonospaceRegular-
6ZWg.ttf")
        self.add_font("Monospace-Unicode", style="b", fname="fonts/MonospaceBold-zmP0.ttf")
        self.add_font("Monospace-Unicode", style="i", fname="fonts/MonospaceOblique-
5meB.ttf")

        self.add_page()

    def header(self) -> None:
        self.set_font('Arial', 'B', 15)
        self.cell(text="Data Analysis Report", center=True)
        if self._show_time:
            self.set_font('Arial', 'I', 10)
            self.cell(0, 10, f"Generated: {self.create_time}", align="R", ln=True)
        else:
            self.ln()
        self.line(self.l_margin, self.get_y(), self.w - self.r_margin, self.get_y())
        self.ln(5)

    def footer(self) -> None:
        self.set_y(-15)
        self.set_font('Arial', 'I', 10)
        self.cell(w=0, text=f'Page {self.page_no()}', align='C')

    def add_text(self, text: str = "", monospaced: bool = False, style: str = "") -> None:
        if monospaced:
            self.set_font("Monospace-Unicode", style, 14)
        else:
            self.set_font("Times", style, 14)
```

```
        self.write(text=text+"\n")

    def add_heading(self, text: str = "") -> None:
        self.ln(5)
        self.add_text(text="        " + text, style="B")
        self.ln(3)

    def add_series(self, s: pd.Series, title: str = "") -> None:
        self.add_dataframe(df=pd.DataFrame(s), title=title, col_names=False)

    def add_dataframe(self, df: pd.DataFrame, title: str = "", col_names: bool = True, max_cols:
int = 4,
                      max_col_width: int = 14) -> None:
        if title:
            self.add_heading(text=title)

        def truncate_column_name(name: str, max_length: int) -> str:
            return name if len(name) <= max_length else f"{name[:max_length//2-1]}...{name[-
(max_length//2-2):]}"

        df = df.rename(columns={col: truncate_column_name(str(col), max_col_width) for col in
df.columns})

        for start in range(0, len(df.columns), max_cols):
            chunk = df.iloc[:, start:start + max_cols]
            self.add_text(text=chunk.to_string(header=col_names)+"\n", monospaced=True)
        self.add_text()

    def add_plot(self, title: str = None) -> None:
        if title:
            plt.title(title)
        img_buf = BytesIO()
        plt.savefig(img_buf, dpi=self._dpi, bbox_inches='tight')
        self.image(img_buf, w=self.epw)
```

```
        img_buf.close()
        plt.close()

    def _add_subplots_row(self, plot_funcs: list[callable], cols: int = 2, suptitle: str = None) ->
None:
        fig, axes = plt.subplots(1, cols, figsize=(5 * cols, 5))
        axes = axes if cols > 1 else [axes]

        for ax, func in zip(axes, plot_funcs):
            func(ax)

        for ax in axes[len(plot_funcs):]:
            ax.axis('off')

        if suptitle:
            fig.suptitle(suptitle, fontsize=cols*7, fontweight='bold', y=1.05)

        plt.tight_layout(pad=1.0)
        self.add_plot()

    def add_subplots(self, plot_funcs: list[callable], cols: int = 2, suptitle: str = None) -> None:
        self._add_subplots_row(plot_funcs[:cols], cols, suptitle)
        for i in range(cols, len(plot_funcs), cols):
            self._add_subplots_row(plot_funcs[i:i + cols], cols)

    def to_bytes(self) -> BytesIO:
        return BytesIO(self.output())
```

Файл app/reporting/routes.py

Реалізація функціональної задачі генерації рекомендаційного звіту.

```
from flask import send_file, request
from flask_pydantic_spec import FileResponse

from app.controllers import DataFrameAnalyzer
from app.extensions import storage
from app.reporting import bp
```

```
from app.extensions import spec
from app.models import AnalysisParams, DatasetTokenHeader



@bp.route("/datasets/<dataset_id>/report")
@spec.validate(
    query=AnalysisParams,
    headers=DatasetTokenHeader,
    resp=FileResponse(content_type='application/pdf'),
    tags=["Recommendations report"]
)
def get_recommendations(dataset_id: str) -> FileResponse:
    params: AnalysisParams = request.context.query  # noqa
    data = storage.get_dataset(dataset_id)

    report = DataFrameAnalyzer(data).generate_report(params)
    return send_file(report.to_bytes(), mimetype='application/pdf', as_attachment=False,
download_name='report.pdf')
```

Файл app/extensions/storage.py

Реалізація функціональної задачі управління сховищем даних.

```
import os
import time
import uuid
from datetime import datetime, timezone

import pandas as pd
from flask import Flask, current_app, url_for, request
from werkzeug.exceptions import BadRequest, NotFound, InternalServerError



class Storage:

    def __init__(self, app: Flask = None) -> None:
        if app is not None:
            self.init_app(app)
```

```
    def init_app(self, app: Flask) -> None:
        if not hasattr(app, 'extensions'):
            app.extensions = {}
        app.extensions['storage'] = self

    @property
    def storage_location(self) -> str:
        return current_app.config["DATASET_STORAGE"]

    @property
    def dataset_max_age(self) -> int:
        return current_app.config["DELETE_AGE_HOURS"]

    @property
    def access_key_header(self) -> str:
        return current_app.config["ACCESS_KEY_HEADER"]

    def cleanup(self) -> None:
        check_time = datetime.now(timezone.utc).strftime("%Y-%m-%d %H:%M:%S UTC")
        deleted_files = 0
        print(f"[{check_time}] Starting storage cleanup...")

        os.makedirs(self.storage_location, exist_ok=True)

        for file_path in os.listdir(self.storage_location):
            full_path = os.path.join(self.storage_location, file_path)
            if os.path.isfile(full_path) and (time.time() - os.path.getctime(full_path) >
self.dataset_max_age * 3600):
                os.remove(full_path)
                deleted_files += 1

        print(f"□ Cleanup complete! {deleted_files} file(s) deleted.\n")

    def get_dataset(self, dataset_id: str) -> pd.DataFrame:
```

```
        access_key = request.headers.get(self.access_key_header)
        if not access_key:
            raise BadRequest("Missing access key in headers.")

        expected_filename = f"{dataset_id}__{access_key}"
        full_path = os.path.join(self.storage_location, expected_filename)

        if not os.path.exists(full_path):
            raise NotFound("Dataset not found or invalid access key.")

        return pd.read_pickle(full_path)

    def save_dataset(self, data: pd.DataFrame, dataset_id: str = "") -> tuple[str, str]:
        dataset_id = dataset_id or str(uuid.uuid4())
        access_key = request.headers.get(self.access_key_header, str(uuid.uuid4()))

        os.makedirs(self.storage_location, exist_ok=True)

        filename = f"{dataset_id}__{access_key}"
        full_path = os.path.join(self.storage_location, filename)

        try:
            data.to_pickle(full_path)
        except OSError:
            raise InternalServerError(f"Failed to save your dataset. Try again later or consider using "
                                      f"'{url_for('system.analyze_data')}' endpoint for all-in-one request.")

        return dataset_id, access_key
```

Файл app/system/routes.py

Реалізація додаткового функціоналу рівня застосунку.

```
from flask import request, send_file, jsonify, Response
from flask_pydantic_spec import FileResponse, MultipartFormRequest

from app.controllers import DataFrameLoader, DataFramePreprocessor, DataFrameAnalyzer
from app.extensions import spec
```

```
from app.system import bp
from app.models import FullPipelineParams
from app.errors import ParameterMissing


@bp.route("/")
def index() -> Response:
    ui_set = spec.config._SUPPORT_UI | {spec.config.FILENAME}  # noqa
    base_url = request.host_url.rstrip("/")
    return jsonify({
        "message": "Welcome to the Automated Data Analysis Web Service",
        "documentation": {ui: f"{base_url}/{spec.config.PATH}/{ui}" for ui in ui_set}
    })


@bp.route("/datasets/full_pipeline", methods=["POST"])
@spec.validate(
    body=MultipartFormRequest(model=FullPipelineParams),
    resp=FileResponse(content_type='application/pdf'),
    tags=["Full pipeline"]
)
def analyze_data() -> FileResponse:
    file = request.files.get('file')
    if file is None or file.filename == '':
        raise ParameterMissing("file")

    params: FullPipelineParams = request.context.body  # noqa

    data = DataFrameLoader(file, params).load_data()
    data = DataFramePreprocessor(data).preprocess(params)
    report = DataFrameAnalyzer(data).generate_report(params)
    return send_file(report.to_bytes(), mimetype='application/pdf', as_attachment=False,
download_name='report.pdf')
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБСЕРВІС ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ДАНИХ ТА ГЕНЕРАЦІЇ ЗВІТІВ

Програма та методика тестування

КПІ.ІП-1118.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Іван ЛЯЛЯ

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є вебсервіс для автоматизованого аналізу даних та генерації звітів.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка зручності програмного інтерфейсу;
- перевірка доступності API з різних платформ/клієнтів (Postman, curl, Python скрипти тощо);
- перевірка стабільності API при різних навантаженнях;
- перевірка обробки помилок і валідації вхідних даних;
- перевірка відповідності структури та формату відповіді API специфікації;
- перевірка безпеки взаємодії з API;
- знаходження проблем, помилок і недоліків з метою їх усунення.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним;
- тестування «сірої скриньки» – об’єктом тестування тут є деякі особливості внутрішньої поведінки програми. Перевіряється коректність вихідних даних при заданих вхідних, застосовується для тестування окремих алгоритмів (функцій).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- SonarQube Cloud – для статичного аналізу коду, перевірки відповідності стандартам програмування, оцінювання якості програмного забезпечення та виявлення вразливостей;
- Postman – для надсилання HTTP-запитів до API, перевірки відповідей, обробки сценаріїв взаємодії з клієнтом;
- curl – для перевірки простих API-запитів;
- Python (requests) – для створення скриптів з різними наборами вхідних даних.

Порядок проведення тестування буде наступним:

- статичне тестування коду;
- функціональне тестування (динамічне);
- тестування доступності з різних клієнтів;
- тестування валідації та обробки помилок;
- тестування відповідності формату відповіді документації;
- тестування системи управління сховищем;
- тестування зручності використання;
- тестування зі змінним навантаженням;
- тестування безпеки;
- системне тестування.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБСЕРВІС ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ДАНИХ ТА ГЕНЕРАЦІЇ ЗВІТІВ

Керівництво користувача

КПІ.ІП-1118.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Іван ЛЯЛЯ

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Вебсервіс для автоматизованого аналізу даних та генерації звітів призначений для прискорення й спрощення процесів попередньої обробки даних, виконання їх базового аналізу та формування узагальнених рекомендацій на основі виявлених закономірностей у табличних наборах даних.

Користувач взаємодіє з сервісом через HTTP/HTTPS API. Передбачено можливість інтеграції з різними клієнтами, серед яких Postman, Python-скрипти, curl або інші інструменти, що підтримують запити до REST API.

Репозиторій проєкту містить повний програмний код та список залежностей застосунку в головній (master) гілці, що є необхідним та достатнім для збірки та запуску ПЗ. У гілці demo розміщено програмний код демонстраційного графічного інтерфейсу для зручної взаємодії з API.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для взаємодії з вебсервісом достатньо наявності HTTP-клієнта (наприклад, curl, Postman або будь-яка програмна бібліотека з підтримкою HTTP-запитів).

Мінімальні системні вимоги:

– підключення до мережі Інтернет зі швидкістю від 32 Мбіт/с.

Рекомендовані системні вимоги:

– підключення до мережі Інтернет зі швидкістю від 100 Мбіт/с.

2.2 Завантаження застосунку

Оскільки сервіс надає доступ до функціоналу через HTTP API, немає потреби у завантаженні та встановленні ПЗ на власний пристрій. Для взаємодії з API необхідно виконати HTTP-запит до публічного URL вебзастосунку, де базова адреса – https://llo3itib4ik.pythonanywhere.com/.

2.3 Перевірка коректної роботи

Перевірка функціонування вебсервісу відбувається шляхом виконання GET HTTP-запиту до базової адреси https://llo3itib4ik.pythonanywhere.com/. Якщо код відповіді HTTP 200 та отримано JSON-відповідь, що містить повідомлення «Welcome to the Automated Data Analysis Web Service», то застосунок працює коректно.

3 ВИКОНАННЯ ПРОГРАМИ

Після виконання GET-запиту до базової адреси вебсервісу користувач отримує відповідь у форматі JSON, що містить привітальне повідомлення та посилання на документацію у доступних форматах (рисунок 3.1). Користувач може перейти за одним із наданих посилань, виконавши запит за відповідною URL-адресою (наприклад, через браузер), і отримати автогенеровану документацію до API у зручному вигляді – у форматі Swagger UI (рисунок 3.2), ReDoc або OpenAPI JSON.

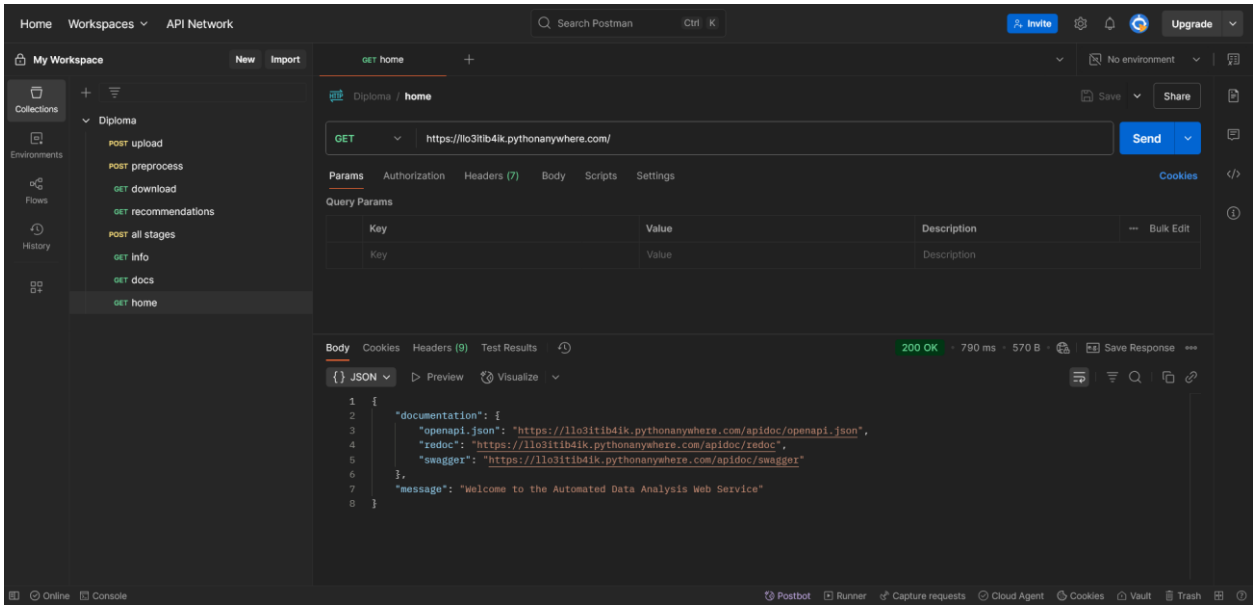


Рисунок 3.1 – Результат запиту до базового URL сервісу у Postman

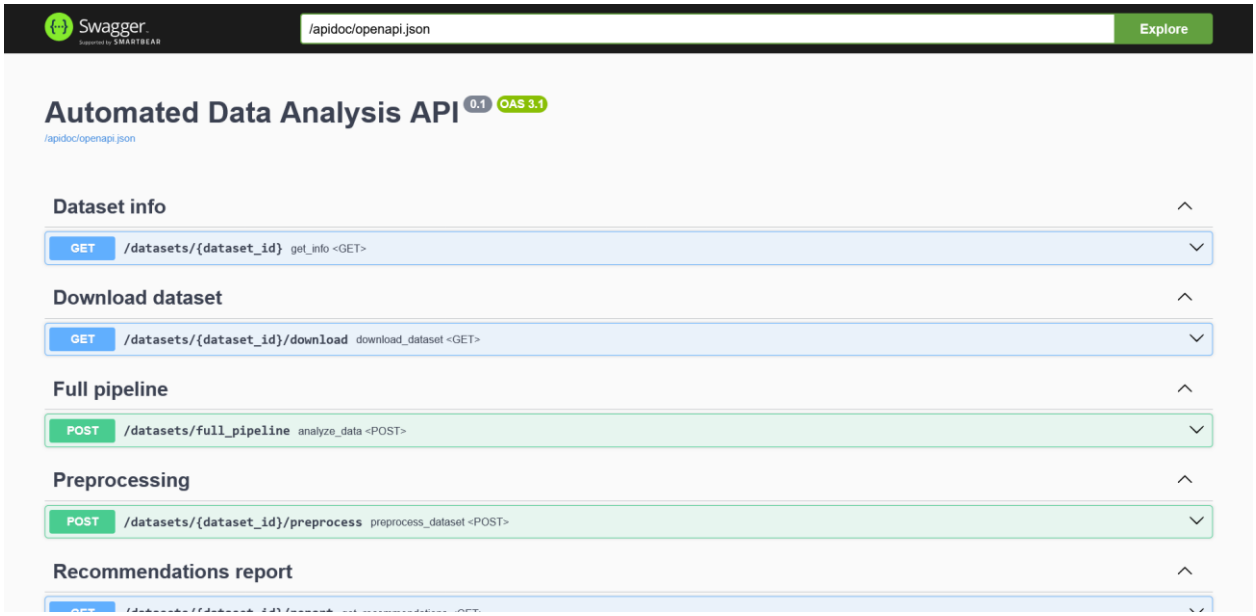


Рисунок 3.2 – Swagger UI документація вебсервісу

Користувач може виконувати запити до ендпоінтів сервісу відповідно до опису в документації, використовуючи будь-який HTTP-клієнт, зокрема безпосередньо через інтерфейс Swagger UI (рисунок 3.2).

Сервіс реалізовано за принципами REST, зокрема – без збереження стану (stateless), тобто інформації про попередні дії, між запитами. Відповідно, вся необхідна інформація повинна передаватися в межах одного запиту.

Для завершення роботи з вебзастосунком не передбачено виконання додаткових дій. Користувач припиняє роботу з HTTP-клієнтом, після чого набір даних, завантажений або створений протягом сесії, зберігатиметься на сервері не менше 24 годин і буде автоматично видалений після закінчення цього терміну.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБСЕРВІС ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ДАНИХ ТА ГЕНЕРАЦІЇ ЗВІТІВ

Графічний матеріал

КПІ.ІП-1118.045440.06.99

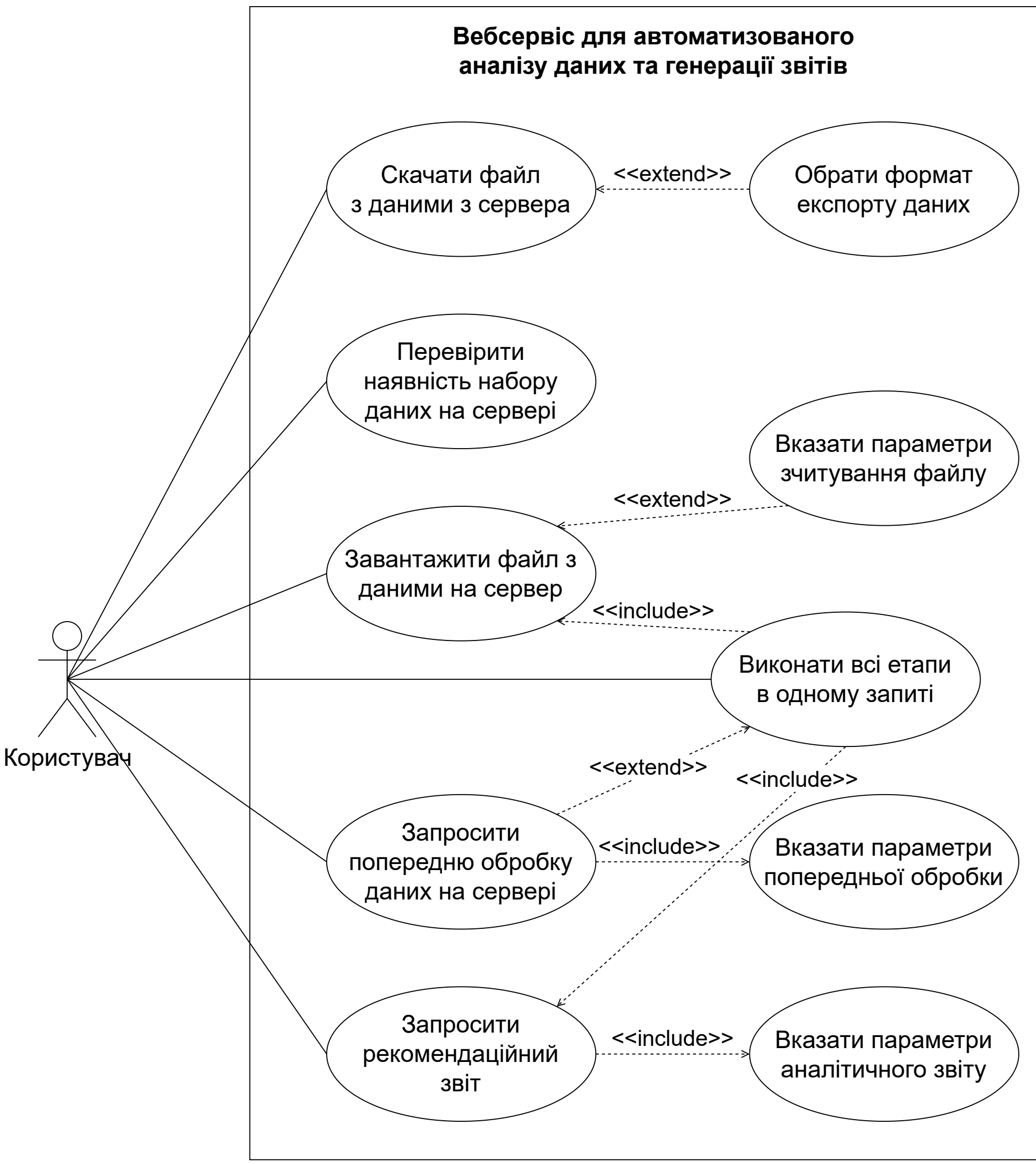
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Тетяна ЛІХОУЗОВА

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Іван ЛЯЛЯ

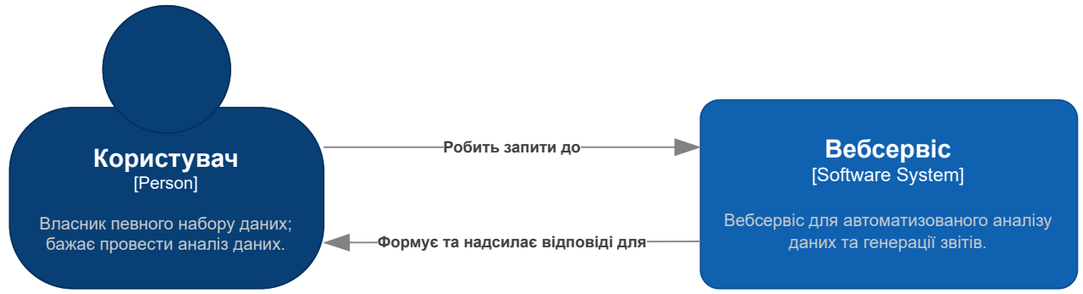
Київ – 2025

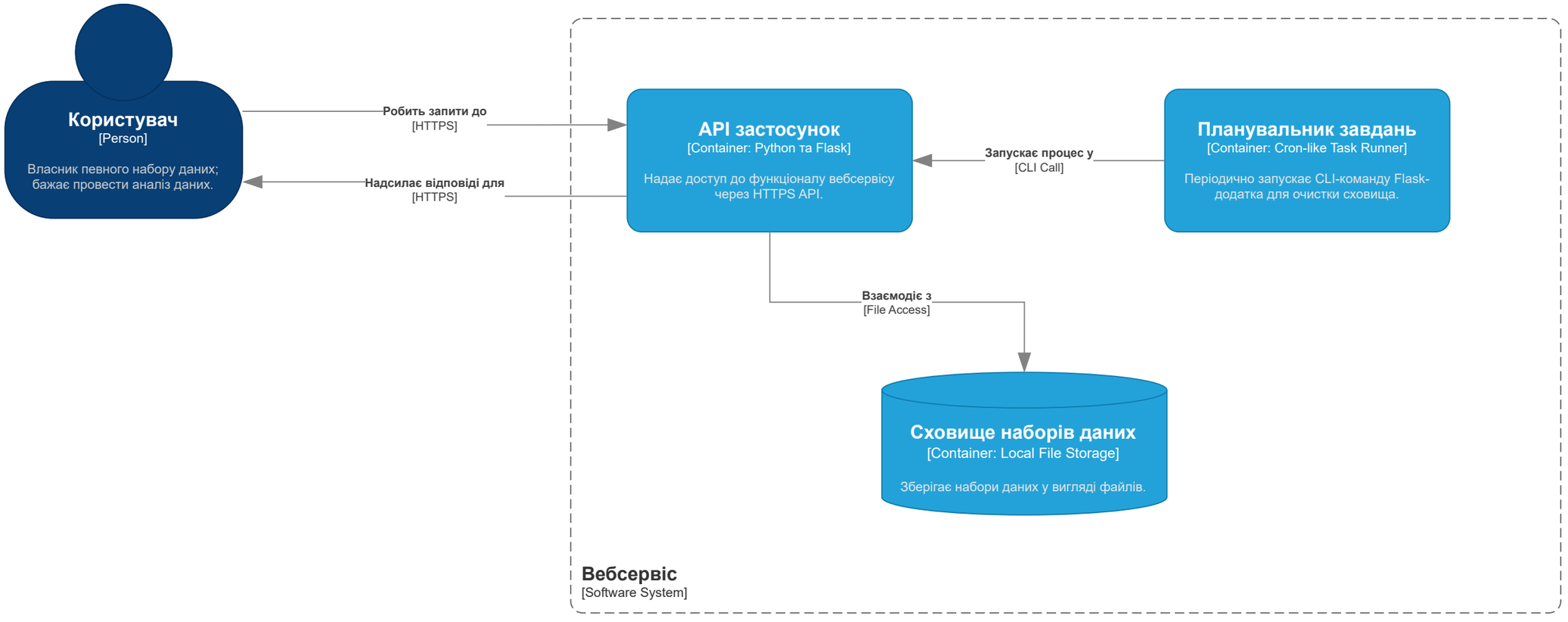
КПІ.ІП-1118.045440.06.99.ССВ



					КПІ.ІП-1118.045440.06.99.ССВ
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна варіантів використання
Розробив		Ляля І.О.			Лит. / Маса / Масштаб
Перевірив		Ліхоузова Т.А.			Аркуш 1 / Аркушів 3
Т. контр.					Вебсервіс для автоматизованого аналізу даних та генерації звітів
Н. контр.		Ліщук К.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.			

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ





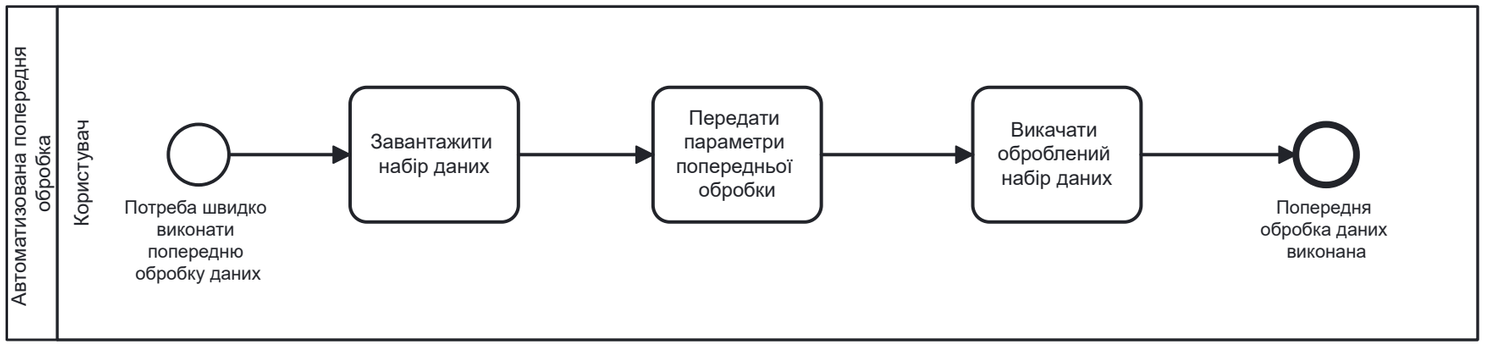
Демонстраційний плакат до дипломного проєкту

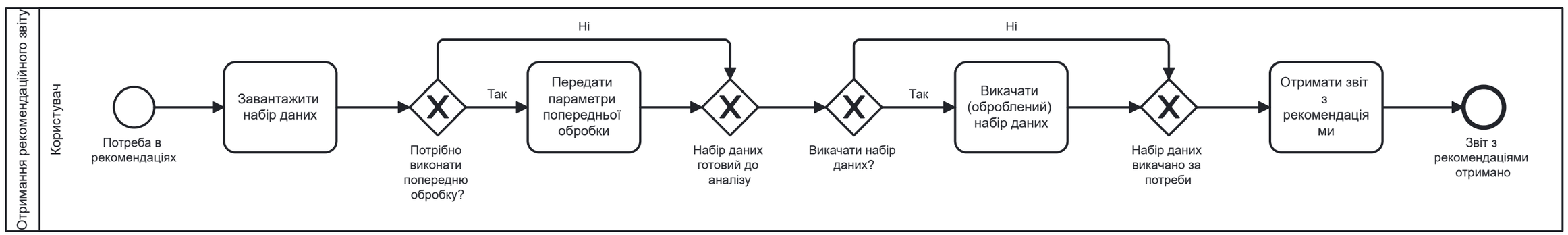
Вебсервіс для автоматизованого аналізу даних та генерації звітів

Виконав студент гр. ІП-11 Ляля І.О.

Керівник Ліхоузова Т.А.

МОДЕЛІ БІЗНЕС-ПРОЦЕСІВ





Демонстраційний плакат до дипломного проєкту

Вебсервіс для автоматизованого аналізу даних та генерації звітів

Виконав студент гр. ІП-11 Ляля І.О.

Керівник Ліхоузова Т.А.