Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Проєкт системи для генерації тестових даних з використання штучного інтелекту

Виконав(ла): студент(ка) 6 курсу, групи СТм-61
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Свінціцький П.В.
(підпис) (прізвище та ініціали)

Керівник Марценко С.В.
(підпис) (прізвище та ініціали)

Нормоконтроль Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"17" листопада 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

студенту Свінціцький Павло Вадимович
(прізвище, ім'я, по батькові)

1. Тема роботи **Проєкт системи для генерації тестових даних з використання штучного інтелекту**

Керівник роботи **к.т.н., доц. Марценко С.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "27" листопада 2025 року № 4/7-1040

2. Термін подання студентом завершеної роботи 22 грудня 2025 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП; 1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ; 1.1 Генерація тестових даних; 1.2 Використання технологій штучного інтелекту для генерації тестових даних; 1.3 Методології генерації тестових даних; 1.4 Підхід до впровадження ШІ для генерації тестових даних та переваги такого підходу; 2 АНАЛІЗ ЗАСОБІВ ГЕНЕРАЦІЇ ТЕСТОВИХ ДАНИХ; 2.1 Кроки впровадження для генерації тестових даних ШІ; 2.2 Галузеві застосування технологій генерації тестових даних за допомогою штучного інтелекту; 2.3 Загальні властивості засобів генерації тестових даних на основі штучного інтелекту; 2.4 Варіанти використання тестових даних у тестуванні веб-сторінок; 3 ОПИС ПРОЄКТУ ПРОПОНОВАНОЇ СИСТЕМИ; 3.1 Огляд програмних інструментів генерації тестових даних на основі технологій ШІ; 3.2 Властивості найпоширеніших генераторів тестових даних; 3.3 Програмний прототип генератора тестових даних на основі ШІ; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ; ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., к.т.н., доц. каф. МТ		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 17 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	17.11.25-18.11.25	Виконано
2.	Підбір наукових джерел за темою роботи	19.11.25-20.11.25	Виконано
3.	Переклад та опрацювання наукових джерел	20.11.25-23.11.25	Виконано
	за темою кваліфікаційної роботи		
4.	Виконання дослідження щодо теми	24.11.25-10.12.25	Виконано
	кваліфікаційної роботи		
5.	Оформлення першого розділу	11.12.25-12.10.25	Виконано
6.	Оформлення другого розділу	12.12.25-13.12.25	Виконано
7.	Оформлення третього розділу	13.12.25-14.12.25	Виконано
8.	Виконання завдання до підрозділу "Охорона праці"	08.12.25-09.11.25	Виконано
9.	Виконання завдання до підрозділу "Безпека в	10.12.25-12.12.25	Виконано
	надзвичайних ситуаціях"		
10.	Оформлення кваліфікаційної роботи	12.12.25-31.12.25	Виконано
11.	Нормоконтроль	14.12.25-15.12.25	Виконано
12.	Перевірка на плагіат	15.12.25	Виконано
13.	Попередній захист кваліфікаційної роботи	18.12.25	Виконано
14.	Захист кваліфікаційної роботи	23.12.2025	

Студент _____ Свінціцький П.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Проєкт системи для генерації тестових даних з використання штучного інтелекту" // Кваліфікаційна робота освітнього рівня "Магістр" // Свінціцький Павло Вадимович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТм-61 // Тернопіль, 2025 // с. – 62, рис. – 16, табл. – 1, джерел – 11.

Ключові слова: забезпечення якості, тестування, тестові дані, штучний інтелект, генерація тестових даних

Великі мовні моделі (LLM) трансформують процес генерації тестових даних, використовуючи архітектури глибокого навчання та алгоритми обробки природної мови для автоматизованого створення складних (комплексних) наборів даних. Це забезпечує зменшення ручних витрат праці при одночасному підвищенні покриття тестування та якості валідації.

Еволюція методів генерації тестових даних демонструє три основні етапи. Ранні підходи базувалися на ручному створенні тестових випадків, що характеризувалося високою трудомісткістю та обмеженим масштабуванням. У 1990-х роках автоматизовані системи на основі правил запровадили шаблонне генерування, проте без можливості адаптації до складної бізнес-логіки.

Якісний прорив відбувся з інтеграцією машинного навчання у 2010-х роках. Впровадження трансформерних архітектур (2014) та подальший розвиток LLM забезпечили фундаментальну зміну парадигми генерації тестових даних.

ANNOTATION

“Project of a System for Generating Test Data Using Artificial Intelligence” // Master’s degree qualification paper // Svintsitskyi Pavlo // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СТм-61 // Ternopil, 2025 // p. – 62, fig. – 16, tables – 1, references – 11.

Key words: quality assurance, testing, test data, artificial intelligence, test data generation

Large Language Models (LLMs) are transforming the test data generation process by leveraging deep learning architectures and natural language processing algorithms for automated creation of comprehensive datasets. This ensures reduced manual labor costs while simultaneously improving test coverage and validation quality.

The evolution of test data generation methods demonstrates three primary stages. Early approaches relied on manual test case creation, characterized by high labor intensity and limited scalability. In the 1990s, rule-based automated systems introduced template-driven generation, though without the capability to adapt to complex business logic.

A qualitative breakthrough occurred with the integration of machine learning in the 2010s. The introduction of transformer architectures (2014) and subsequent LLM development provided a fundamental paradigm shift in test data generation.

ЗМІСТ

ВСТУП 6
1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ 8
1.1 Генерація тестових даних 8
1.2 Використання технологій штучного інтелекту для генерації тестових даних 10
1.3 Методології генерації тестових даних 11
1.4 Підхід до впровадження ШІ для генерації тестових даних та переваги такого підходу 12
2 АНАЛІЗ ЗАСОБІВ ГЕНЕРАЦІЇ ТЕСТОВИХ ДАНИХ 16
2.1 Кроки впровадження для генерації тестових даних ШІ 16
2.2 Галузеві застосування технологій генерації тестових даних за допомогою штучного інтелекту 17
2.3 Загальні властивості засобів генерації тестових даних на основі штучного інтелекту 20
2.4 Варіанти використання тестових даних у тестуванні веб-сторінок 23
3 ОПИС ПРОЄКТУ ПРОПОНОВАНОЇ СИСТЕМИ 25
3.1 Огляд програмних інструментів генерації тестових даних на основі технологій ШІ 25
3.2 Властивості найпоширеніших генераторів тестових даних 27
3.3 Програмний прототип генератора тестових даних на основі ШІ 43
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 48
4.1 Питання щодо охорони праці 48
4.2 Підвищення стійкості роботи об'єктів господарської діяльності у воєнний час 51
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58
ДОДАТКИ 61

ВСТУП

Актуальність задачі.

Тестові дані – це основа тестування програмного забезпечення. Вони представляють собою вхідні дані та умови, які використовуються для перевірки функціональності, продуктивності та безпеки програмного застосунку. Без відповідних та різноманітних тестових даних неможливо ретельно оцінити, як застосунок поводитиметься в руках користувачів.

Безпосередньо про задачу генерації тестових даних наукових публікацій в ТНТУ поки що не було знайдено. Проте варто відмітити роботи, що стосуються питань забезпечення якості програмних продуктів на ранніх етапах проєктування, наприклад, [1], [2]. Також є роботи, присвячені питанням побудови мовних моделей, що застосовуються у цій кваліфікаційній роботі для опрацювання запитів, наприклад, [3 – 6].

Традиційно, генерація тестових даних була ручним процесом, що вимагало від тестувальників створення наборів даних, вхідних значень та тестових сценаріїв вручну. Однак, оскільки програмні застосунки стали складнішими, а потреба в швидкому та безперервному тестуванні зросла, ручна генерація тестових даних стала вузьким місцем.

Саме тут на допомогу приходить штучний інтелект, пропонуючи нову парадигму для генерації тестових даних. Алгоритми штучного інтелекту можуть аналізувати вимоги додатків, історичні дані та моделі використання, щоб автоматично генерувати тестові дані, які є не лише комплексними, але й динамічними та адаптивними.

Мета роботи.

Метою кваліфікаційної роботи є виконання огляду літературних джерел на тему автоматизації тестування та генерації тестових даних з використанням штучного інтелекту.

Для досягнення мети потрібно дослідити та дати відповіді на питання дослідження:

1. Здійснити огляд методів та засобів штучного інтелекту для генерації тестових даних.

2. Здійснити аналіз наявних програмних інструментів генерації тестових даних та автоматизації тестування.

3. Розробити прототип системи для генерації тестових даних.

Об’єкт дослідження: процеси генерування тестових даних для тестування програмного забезпечення з використанням технологій штучного інтелекту.

Предмет дослідження: методи і засоби генерації тестових даних з використанням технологій штучного інтелекту.

Наукова новизна отриманих результатів.

Проведено огляд літературних джерел на тематику генерації тестових даних, автоматизації тестування та використання штучного інтелекту в цих процесах. Запропоновано прототип програмного продукту для генерації тестових даних.

Практичне значення отриманих результатів.

В кваліфікаційній роботі виконано дослідження використання технологій штучного інтелекту для генерації тестових даних. Дане дослідження може мати практичну цінність для створення програмних продуктів та компонентів в процесах забезпечення якості при розробці програмного забезпечення.

Апробація результатів та особистий внесок здобувача.

Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету імені Івана Пулюя. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції "Актуальні задачі сучасних технологій – 2025", яка проводилась у ТНТУ.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Генерація тестових даних

У сучасному швидкозмінному середовищі розробки програмного забезпечення великі моделі штучного інтелекту фундаментально змінюють наш підхід до генерації тестових даних. Ці складні системи використовують глибоке навчання та обробку природної мови для автоматичного створення реалістичних, комплексних наборів тестових даних, які значно зменшують ручні зусилля, одночасно покращуючи якість тестування та охоплення.

За останні кілька десятиліть процес генерації тестових даних зазнав разючих змін. На ранніх етапах тестування програмного забезпечення розробники та тестувальники вручну створювали тестові дані – процес, який був не лише трудомістким, але й схильним до людських помилок та за своєю суттю обмеженим за обсягом.

Протягом 1990-х років з'явилися автоматизовані інструменти на основі правил, які вперше запропонували можливість систематичного генерування тестових даних. Ці інструменти спиралися на попередньо визначені шаблони та патерни, забезпечуючи незначне покращення ефективності, але не маючи інтелекту для розуміння складної бізнес-логіки або створення справді різноманітних сценаріїв.

Прорив відбувся у 2010-х роках із впровадженням методів машинного навчання в робочі процеси тестування. Однак лише з появою моделей-трансформерів у 2014 році та подальшими розробками моделей великих мов програмування генерація тестових даних пережила свій найбільший стрибок уперед (див. рис. 1.1).

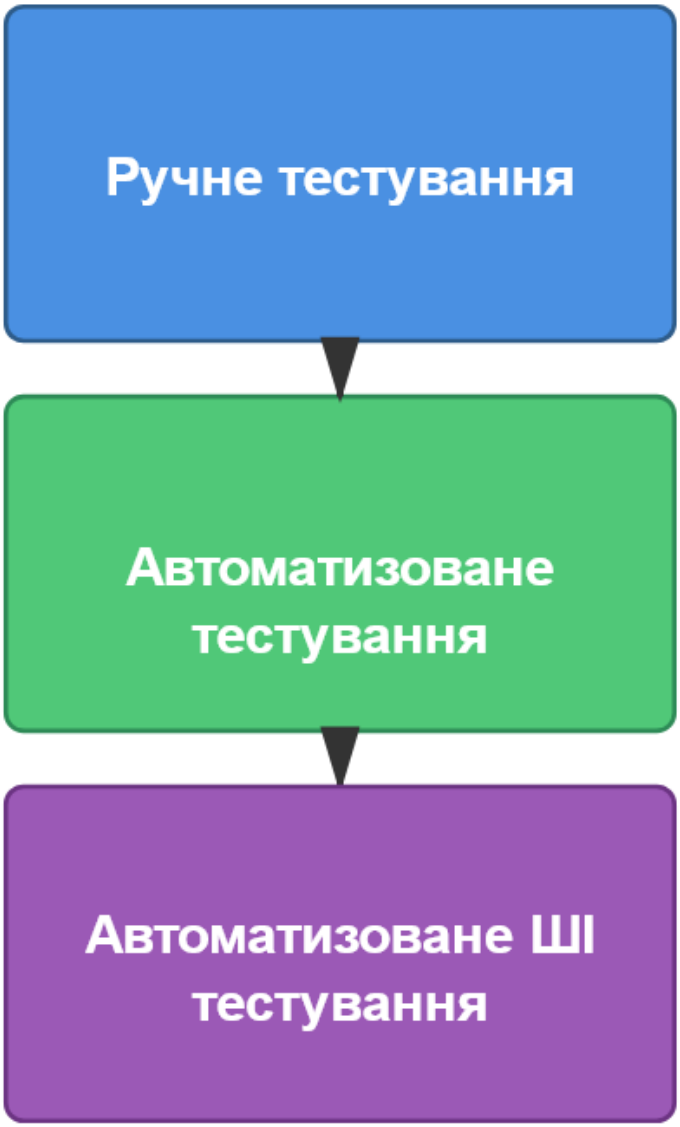


Рисунок 1.1 – Еволюція від ручної до генерації тестових даних за допомогою штучного інтелекту

Ключові етапи розвитку генерації тестових даних:

- 1980-ті-1990-ті. Домінувало ручне створення тестових даних, що вимагало значних людських ресурсів.
- 2000-ті. З'явилися базові генератори на основі правил, що пропонували обмежену автоматизацію.
- 2010-ті. Методи машинного навчання почали інтегруватися в робочі процеси тестування
- 2014. Впровадження архітектури Transformer заклало основу для передових моделей штучного інтелекту.

- 2018 – дотепер. GPT та подібні великі моделі революціонізували автоматизовану генерацію тестових даних.

Тестові дані – це інформація, яка використовується під час тестування програмного забезпечення для перевірки функціональності, продуктивності та надійності системи. Ці дані повинні всебічно охоплювати нормальні операції, граничні випадки та сценарії збоїв, щоб забезпечити надійну валідацію системи.

Синтетичні дані являють собою штучно згенеровану інформацію, яка імітує характеристики даних реального світу, не маючи фактичної конфіденційної інформації. Такий підхід дозволяє проводити тестування, зберігаючи дотримання конфіденційності та нормативних вимог.

Великі мовні моделі (LLM) – це складні системи штучного інтелекту з мільярдами параметрів, здатні розуміти та генерувати текст, подібний до людського. У контексті тестування ці моделі аналізують структури даних та бізнес-вимоги для створення контекстуально відповідних тестових сценаріїв.

Генерація з урахуванням схеми описує процес, за допомогою якого моделі штучного інтелекту аналізують схеми баз даних, специфікації API та структури даних для створення відповідних тестових даних. Це гарантує, що згенеровані дані дотримуються обмежень, зв'язків та бізнес-правил, властивих системному дизайну.

1.2 Використання технологій штучного інтелекту для генерації тестових даних

Сучасна генерація тестових даних на основі штучного інтелекту працює через кілька складних технічних рівнів, кожен з яких сприяє здатності всієї системи створювати високоякісні та релевантні тестові дані.

Механізм обробки природної мови: Розширені можливості NLP дозволяють моделям штучного інтелекту аналізувати документацію, розуміти бізнес-вимоги та інтерпретувати визначення схем. Цей компонент перетворює

зрозумілі для людини специфікації на параметри генерації, що підлягають машинному застосуванню.

- Система розпізнавання образів. Алгоритми машинного навчання визначають зв'язки між елементами даних, розуміють потоки бізнес-логіки та розпізнають шаблони обмежень. Ця система гарантує, що згенеровані дані зберігають логічну узгодженість та відповідність бізнес-правилам.
- Модуль відповідності обмеженням. Складні алгоритми гарантують, що всі згенеровані дані відповідають визначеним обмеженням, включаючи вимоги до типу даних, діапазони значень та реляційну цілісність. Цей компонент запобігає генерації недійсних даних, одночасно максимізуючи охоплення тестових сценаріїв.
- Аналізатор граничних умов. Спеціалізовані алгоритми визначають та генерують тестові випадки для граничних умов, екстремальних значень та незвичайних комбінацій вхідних даних. Ця можливість забезпечує повне охоплення тестуванням, включаючи сценарії, які тестувальники-люди можуть пропустити.

1.3 Методології генерації тестових даних

- Генеративно-змагальні мережі (GAN). Ці системи використовують конкуруючі нейронні мережі – генератори та дискримінатори – для створення дедалі реалістичніших синтетичних даних. Процес змагального навчання забезпечує високоякісний результат, який максимально нагадує розподіл даних у реальному світі.
- Варіаційні автоенкодери (VAE). Архітектури VAE вивчають стиснуті представлення шаблонів даних, що дозволяє генерувати нові зразки, які зберігають статистичні властивості вихідних наборів даних. Цей підхід особливо добре підходить для збереження характеристик розподілу даних.
- Моделі на основі трансформаторів. Використовуючи механізми уваги, архітектури трансформаторів чудово розуміють складні взаємозв'язки в

структурах даних та генерують контекстуально відповідні тестові сценарії. Ці моделі демонструють чудову продуктивність в обробці послідовних та структурованих даних (див. рис. 2.2).

Рисунок 1.2 –Схема робочого процесу генерації тестових даних ШІ

Впровадження в реальному світі: тематичне дослідження платформи електронної комерції

Нещодавно велика платформа електронної комерції впровадила генерацію тестових даних на основі штучного інтелекту для вирішення проблем тестування алгоритму рекомендацій та продуктивності системи за різних умов навантаження.

1.4 Підхід до впровадження ШІ для генерації тестових даних та переваги такого підходу

1. Фаза аналізу схеми. Система штучного інтелекту спочатку проаналізувала складну схему бази даних платформи, включаючи таблиці користувачів, каталоги продуктів, історію замовлень та дані відстеження поведінки. Цей аналіз виявив 47 різних типів сутностей з 312 обмеженнями зв'язків.

2. Інтеграція бізнес-логіки. Методи обробки природної мови проаналізували документи з бізнес-вимогами, витягнувши 156 бізнес-правил, що регулюють взаємодію з користувачами, процеси закупівель та управління запасами. Ці правила вплинули на створення реалістичних сценаріїв взаємодії користувача.

3. Виробництво синтетичних даних. Система згенерувала 500 000 профілів користувачів із відповідними моделями поведінки, 2,3 мільйона записів взаємодії з продуктами та 850 000 історій транзакцій, що охоплюють 18 місяців змодельованої активності.

Впровадження призвело до суттєвих покращень за кількома показниками.

- Скорочення часу: час підготовки тестових даних скоротився з 72 годин до 45 хвилин – покращення на 96%.
- Покращення покриття: покриття тестових сценаріїв збільшено з 62% до 94%, виявлено 23 раніше невиявлених вузьких місця системи.
- Покращення якості: згенеровані дані досягли 97% узгодженості зі шаблонами виробничих даних, зберігаючи при цьому повну відповідність конфіденційності.

Ключовими перевагами генерації тестових даних на основі штучного інтелекту можна вважати наступні пункти.

Підвищена ефективність та швидкість. Системи штучного інтелекту можуть генерувати величезні обсяги тестових даних за лічені хвилини, а не за дні чи тижні, необхідні для ручного створення. Таке прискорення дозволяє здійснювати частіші цикли тестування та підтримує гнучкі методології розробки, де швидка ітерація є важливою.

Дослідження показують, що підходи на основі штучного інтелекту скорочують час створення тестових даних на 60–80% порівняно з традиційними методами. Тепер команди можуть виділити свої цінні людські ресурси на стратегію тестування та аналіз вищого рівня, а не на повторювані завдання створення даних.

Моделі штучного інтелекту чудово справляються зі створенням різноманітних тестових сценаріїв, які люди-тестувальники можуть не враховувати. Аналізуючи величезну кількість шаблонів реальних даних, ці системи можуть виявляти граничні випадки, граничні умови та незвичайні комбінації вхідних даних, які можуть виявити критичні вразливості системи.

Покращення покриття включають:

- Тестування граничних значень: автоматична генерація мінімальних/максимальних значень, нульових умов та сценаріїв граничного тестування.
- Моделювання граничних випадків: створення незвичайних, але допустимих комбінацій вхідних даних, які перевіряють стійкість системи до стресових навантажень.
- Об'ємне тестування: генерація великомасштабних наборів даних для тестування продуктивності та масштабованості.

Однією з найважливіших переваг синтетичних даних, згенерованих штучним інтелектом, є їхній невід'ємний захист конфіденційності. Оскільки дані створені штучно, вони не містять фактичної особистої інформації, що усуває ризики, пов'язані з витоками даних або порушеннями нормативних актів.

Сучасні методи генерації синтетичних даних включають методи диференціальної конфіденційності, що гарантує, що навіть статистичні шаблони не можуть бути зворотно спроектовані для ідентифікації окремих записів. Такий підхід дозволяє тестувати з реалістичними даними, зберігаючи при цьому повну відповідність GDPR, HIPAA та іншим нормативним вимогам.

Проблеми якості та реалістичності даних. Хоча дані, згенеровані штучним інтелектом, пропонують численні переваги, забезпечення їх точного відображення складності реального світу залишається складним завданням. Згенеровані дані іноді можуть не мати тонких невідповідностей та аномалій, присутніх у реальних виробничих середовищах.

Стратегії пом'якшення наслідків:

- Статистична валідація. Впровадження комплексних статистичних тестів на подібність для перевірки відповідності згенерованих даних виробничим моделям.
- Перевірка бізнес-правил. Автоматизовані системи перевірки гарантують, що всі згенеровані дані відповідають складним бізнес-логічним обмеженням.
- Ітеративне вдосконалення. Процеси постійного вдосконалення, які коригують параметри генерації на основі результатів тестування.

Незважаючи на притаманні синтетичним даним переваги конфіденційності, організації все одно повинні впроваджувати надійні заходи безпеки для захисту моделей генерації та запобігання потенційному зловживанню.

Вимоги безпеки при впровадженні технологій ШІ для генерації тестових даних:

- Захист моделі: захист самих моделей штучного інтелекту від несанкціонованого доступу або маніпуляцій.
- Контроль доступу: впровадження дозволів на основі ролей для можливостей генерації даних.
- Аудиторські журнали: ведення вичерпних журналів усіх дій зі створення даних для цілей дотримання вимог.

2 АНАЛІЗ ЗАСОБІВ ГЕНЕРАЦІЇ ТЕСТОВИХ ДАНИХ

2.1 Кроки впровадження для генерації тестових даних ШІ

Фаза 1: Оцінка та планування.

1. Аналіз вимог. Слід почати з ретельного документування поточних потреб у тестових даних, включаючи типи даних, обсяги та вимоги до складності. Варто визначити конкретні проблемні точки в існуючих ручних процесах та встановити показники успіху для впровадження штучного інтелекту.

2. Огляд технічної інфраструктури. Оцінка існуючої інфраструктури розробки та тестування для визначення вимог до інтеграції. Забезпечення достатньої кількості обчислювальних ресурсів для роботи моделі штучного інтелекту та процесів генерації даних.

3. Підготовка команди. Навчання команд розробників та тестувальників методологіям на основі штучного інтелекту. Чітко визначити ролі та обов'язки щодо управління новими автоматизованими системами.

Фаза 2: Пілотне впровадження.

1. Визначення схеми. Створення комплексних визначень структури даних, включаючи типи полів, обмеження та зв'язки. Ця фундаментальна робота дозволяє системі штучного інтелекту розуміти ваші вимоги до даних та генерувати відповідні тестові сценарії.

2. Специфікація бізнес-правил. Документування бізнес-логіки, правил перевірки та вимог до потоку даних у форматах, які можуть обробляти системи штучного інтелекту. Цей крок гарантує, що згенеровані дані відповідають специфічним операційним обмеженням організації.

3. Початкове тестування. Слід почати з простих, добре зрозумілих структур даних, перш ніж переходити до складних сценаріїв. Такий підхід дозволяє командам зміцнити впевненість у системі, водночас виявляючи потенційні проблеми на ранніх етапах процесу впровадження.

Фаза 3: Розгортання у виробничому середовищі.

1. Впровадження перевірки якості. Впровадження автоматизованих процесів перевірки якості згенерованих даних за заздалегідь визначеними критеріями. Включає як технічну перевірку (типи даних, обмеження), так і бізнес-перевірку (логічна узгодженість, реалістичні закономірності).

2. Інтеграція з робочими процесами тестування. Підключення системи генерації ШІ до існуючих конвеєрів тестування за допомогою API та інструментів автоматизації. Забезпечення безперешкодного інтегрування згенерованих даних у поточні процеси тестування без порушення встановлених робочих процесів.

3. Моніторинг та оптимізація. Впровадження комплексних систем моніторингу, які відстежують продуктивність генерації, показники якості даних та використання системи. Використання цієї інформації для постійної оптимізації параметрів генерації та покращення якості продукції.

2.2 Галузеві застосування технологій генерації тестових даних за допомогою штучного інтелекту

Системи охорони здоров'я.

Організації охорони здоров'я успішно впровадили генерацію тестових даних на основі штучного інтелекту, щоб задовольнити суворі вимоги конфіденційності, зберігаючи при цьому реалістичні шаблони даних пацієнтів для тестування систем.

Великий постачальник медичних послуг скоротив час підготовки даних тестування на 85%, досягнувши при цьому 100% відповідності HIPAA завдяки генерації синтетичних записів пацієнтів. Система створила різноманітні групи пацієнтів з реалістичними історіями хвороби, схемами лікування та даними про результати для комплексного тестування системи електронних медичних карток.

Фінансові послуги.

Банківські установи використовують синтетичні дані, згенеровані штучним інтелектом, для тестування систем виявлення шахрайства, можливостей обробки транзакцій та функцій регуляторної звітності.

Один багатонаціональний банк впровадив генерацію тестових даних на основі штучного інтелекту для своїх систем оцінки кредитних ризиків, створивши 10 мільйонів синтетичних профілів клієнтів з реалістичною фінансовою поведінкою та кредитною історією. Такий підхід дозволив комплексне тестування моделі, зберігаючи при цьому повний захист конфіденційності клієнтів.

Уряд та оборона.

Урядові установи використовують дані, згенеровані штучним інтелектом, для тестування систем обслуговування громадян, програм безпеки та платформ аналізу даних, забезпечуючи при цьому захист конфіденційної інформації.

Наступне покоління систем генерації тестових даних на основі ШІ включатиме більш глибоке розуміння бізнес-контексту та вимог до предметної області. Ці системи вийдуть за рамки простої генерації даних і перейдуть до створення інтелектуальних сценаріїв, що враховують складні бізнес-робочі процеси та шляхи користувачів. Нові можливості включають:

- Багатомодальна генерація. Одночасне створення тестових даних для тексту, зображень, аудіо та структурованих даних.
- Часова обізнаність. Генерування даних часових рядів з реалістичними часовими закономірностями та сезонними коливаннями.
- Міжсистемна інтеграція. Створення тестових даних, які забезпечують узгодженість між кількома взаємопов'язаними системами.

Генерація тестових даних для штучного інтелекту розвивається в напрямку спеціалізованих рішень, адаптованих для конкретних галузей та випадків використання. Ці предметно-орієнтовані підходи включають галузеві знання, нормативні вимоги та спеціалізовані шаблони даних.

Майбутні впровадження передбачатимуть глибшу інтеграцію з життєвими циклами розробки програмного забезпечення, автоматично генеруючи тестові дані як частину процесів безперервної інтеграції та розгортання.

Покращення робочого процесу може відбуватись в наступних напрямках.

- Генерація в режимі реального часу. Створення тестових даних на вимогу, що ініціюється коммітами коду або подіями розгортання.
- Інтелектуальне масштабування. Автоматичне налаштування обсягу та складності даних на основі вимог тестування.
- Цикли зворотного зв'язку. Системи, які навчаються на результатах тестування для покращення якості генерації даних у майбутньому.

Великі моделі штучного інтелекту фундаментально перетворили генерацію тестових даних з ручного, трудомісткого процесу на інтелектуальну, автоматизовану функцію, яка підвищує як ефективність, так і якість тестування. Завдяки складним методам обробки природної мови, розпізнавання образів та генерації синтетичних даних, ці системи забезпечують комплексне тестове покриття, зберігаючи при цьому дотримання конфіденційності та знижуючи експлуатаційні витрати.

Докази реальних впроваджень демонструють суттєві переваги – команди розробників досягають економії часу на 60–80% на підготовку тестових даних, покращення охоплення тестами на 40% та майже ідеального дотримання конфіденційності завдяки підходам на основі синтетичних даних. Ці покращення дозволяють організаціям розподіляти людські ресурси на більш цінні види тестування, дотримуючись при цьому суворих стандартів якості.

Організації, які досягають найбільшого успіху в генерації тестових даних на основі штучного інтелекту, зосереджуються на ретельному плануванні, поступовому впровадженні та постійній оптимізації. Вони інвестують у навчання команди, встановлюють чіткі показники якості та підтримують міцну інтеграцію з існуючими робочими процесами розробки.

З розвитком технологій штучного інтелекту ми можемо очікувати ще більш складних можливостей, включаючи мультимодальне генерування даних,

галузеві рішення та глибшу інтеграцію з життєвими циклами розробки. Організації, які почнуть впроваджувати ці технології зараз, будуть найкраще підготовлені до використання майбутніх удосконалень та збереження конкурентних переваг у якості програмного забезпечення та швидкості його розробки.

Трансформація генерації тестових даних є лише одним із прикладів того, як штучний інтелект революціонізує практики розробки програмного забезпечення. Завдяки продуманому та стратегічному застосуванню цих технологій команди розробників можуть досягти нових рівнів ефективності, якості та інновацій, зберігаючи при цьому суворі стандарти, необхідні для надійних програмних систем.

Однак успішне впровадження вимагає більше, ніж просто впровадження технологій. Воно вимагає відданості розумінню бізнес-вимог, підтримці стандартів якості даних та збереженню людського досвіду, який керує ефективними стратегіями тестування. Коли можливості штучного інтелекту поєднуються з людським розумінням та стратегічним мисленням, результатом є потужний підхід до генерації тестових даних, який приносить користь як командам розробників, так і кінцевим користувачам завдяки більш надійним, ретельно протестованим програмним системам.

2.3 Загальні властивості засобів генерації тестових даних на основі штучного інтелекту

Впровадження генерації тестових даних на основі штучного інтелекту включає кілька ключових кроків.

1. Синтез даних.

Алгоритми штучного інтелекту можуть синтезувати дані, що імітують реальні сценарії. Наприклад, якщо ви тестуєте веб-сайт електронної комерції, штучний інтелект може генерувати профілі клієнтів з різними атрибутами,

такими як імена, адреси та історія покупок. Ці синтезовані дані можна використовувати для моделювання різних взаємодій користувачів.

2. Маскування та анонімізація даних.

Конфіденційність даних є головним пріоритетом у тестуванні програмного забезпечення, особливо під час роботи з конфіденційною інформацією. Штучний інтелект може допомогти маскувати та анонімізувати дані, замінюючи конфіденційну інформацію реалістичними, але вигаданими даними. Наприклад, ШІ може замінювати справжні імена вигаданими, гарантуючи, що під час тестування не буде розкрито жодних фактичних даних користувачів.

3. Доповнення даних.

У деяких випадках тестування може вимагати більшого набору даних, ніж той, що є в наявності. Штучний інтелект може доповнити існуючі набори даних, генеруючи додаткові точки даних. Наприклад, якщо у вас є набір відгуків клієнтів, ШІ може генерувати додаткові відгуки з різними настроями, щоб перевірити надійність алгоритмів аналізу настроїв.

Розглянемо кілька реальних прикладів того, як штучний інтелект використовується для генерації тестових даних.

1. Тестування програмного забезпечення для охорони здоров'я.

Нехай є потреба протестувати програмне забезпечення для системи охорони здоров'я, яка керує записами пацієнтів. Забезпечення конфіденційності та безпеки даних є надзвичайно важливим. Штучний інтелект може створювати синтетичні записи пацієнтів, які дуже схожі на реальні дані пацієнтів, включаючи історії хвороби, діагнози та плани лікування. Це дозволяє вам ретельно протестувати обробку програмним забезпеченням конфіденційної медичної інформації без шкоди для конфіденційності пацієнтів.

2. Тестування додатків для електронної комерції.

Для додатків електронної комерції тестування часто включає такі сценарії, як обробка замовлень, платіжні транзакції та управління запасами. Штучний інтелект може генерувати синтетичні дані для продуктів, клієнтів та транзакцій, що дозволяє вам імітувати різні сценарії покупок. Ці динамічні тестові дані

гарантують, що ваш додаток може обробляти широкий спектр взаємодій користувачів, від перегляду продуктів до здійснення покупок.

Впровадження генерації тестових даних на основі штучного інтелекту вимагає відповідних технологій та інструментів. Ось деякі популярні варіанти.

1. Бібліотеки генерації синтетичних даних.

Faker. Бібліотека Python, яка генерує фальшиві дані, такі як імена, адреси та дати.

SynthDet. Інструмент для створення синтетичних даних для моделей виявлення об'єктів.

Mockaroo. онлайн-платформа для створення реалістичних тестових даних для різних випадків використання.

2. Інструменти маскування та анонімізації даних.

Google DLP. API запобігання втраті даних (DLP) від Google допомагає маскувати та анонімізувати конфіденційні дані.

Apache Nifi. Інструмент інтеграції даних з відкритим кодом, що включає процесори для маскування даних.

3. Фреймворки для доповнення даних.

Керас ImageDataGenerator: Бібліотека доповнення даних для наборів зображень, що використовуються в машинному навчанні.

NLPAu. Бібліотека Python для доповнення текстових даних для завдань обробки природної мови.

Генерація тестових даних на основі штучного інтелекту – це революційний процес у тестуванні програмного забезпечення. Вона дозволяє тестувальникам створювати різноманітні, динамічні та конфіденційні набори даних, що забезпечують ретельне тестування програмних застосунків. Впроваджуючи генерацію тестових даних на основі штучного інтелекту, організації можуть пришвидшити свої процеси тестування, покращити охоплення тестами та надавати програмні продукти вищої якості.

2.4 Варіанти використання тестових даних у тестуванні веб-сторінок

Тестові дані відіграють вирішальну роль у забезпеченні точності, функціональності та продуктивності веб-сторінок. Ось деякі ключові випадки використання, коли тестові дані є важливими для тестуваня.

Перевірка та надсилання форми.

- Тестові дані потрібні для перевірки полів введення веб-форм, що гарантує дотримання правильних типів даних, форматів та обмежень.
- Це дозволяє тестувальникам оцінювати поведінку веб-форм під час введення різних типів даних, таких як дійсні, недійсні або граничні значення.
- Тестові дані допомагають оцінити точність процесів надсилання форм, включаючи обробку помилок, повідомлення про успіх та збереження даних.

Автентифікація та авторизація користувача.

- Тестові дані необхідні для імітації різних облікових даних та ролей користувачів під час процесів входу та автентифікації.
- Це дозволяє тестувальникам оцінювати поведінку веб-сторінки на основі різних сценаріїв користувача, таких як дійсні облікові дані, неправильні паролі або заблоковані облікові записи.
- Тестові дані допомагають перевірити механізми авторизації, оцінюючи доступність різних веб-сторінок та функцій на основі ролей та дозволів користувачів.

Відображення та рендеринг даних.

- Тестові дані мають вирішальне значення для оцінки точності відображення та візуалізації даних на веб-сторінках.
- Це дозволяє тестувальникам перевіряти правильність представлення різних типів даних, таких як текст, числа, дати, зображення або мультимедійний контент.
- Це допомагає виявити проблеми, пов'язані з усіканням даних, форматуванням або вирівнюванням, які можуть впливати на візуальне представлення веб-контенту.

Data driven test.

- Тестові дані використовуються для проведення тестування на основі даних, де різні набори вхідних даних застосовуються до веб-сторінки для оцінки її поведінки та реакції.
- Це дозволяє тестувальникам перевіряти функціональність та продуктивність веб-сторінки за різних сценаріїв даних, забезпечуючи її надійність та масштабованість.
- Це допомагає виявити потенційні проблеми, пов'язані з обробкою, перетворенням або зміною даних, які можуть вплинути на точність або продуктивність веб-сторінки.

Локалізація та інтернаціоналізація.

- Тестові дані є важливими для оцінки аспектів локалізації та інтернаціоналізації веб-сторінок.
- Це допомагає тестувальникам оцінити поведінку веб-сторінки, коли застосовуються різні мови, формати дати, часові пояси або культурні уподобання.
- Це допомагає виявити проблеми, пов'язані з кодуванням тексту, перекладом або адаптацією контенту, які можуть вплинути на зручність використання та доступність веб-сторінки в різних регіонах.

Використовуючи відповідні та різноманітні тестові дані в цих випадках використання, тестувальники можуть забезпечити надійність, функціональність та зручність використання веб-сторінок. Це призводить до покращення взаємодії з користувачем та підвищення якості застосунків.

3 ОПИС ПРОЄКТУ ПРОПОНОВАНОЇ СИСТЕМИ

3.1 Огляд програмних інструментів генерації тестових даних на основі технологій ШІ

У сучасному середовищі розробки програмного забезпечення важливість надійних методологій тестування неможливо переоцінити. Одним із ключових аспектів тестування програмного забезпечення є створення надійних та різноманітних тестових даних. Якість тестових даних безпосередньо впливає на ефективність тестування. Це допомагає виявляти потенційні проблеми та забезпечує загальну стабільність і продуктивність програмного забезпечення.

Щоб допомогти тестувальникам програмного забезпечення в цій справі, з'явилося кілька чудових інструментів для генерації тестових даних. Отже, тестувальникам програмного забезпечення важливо бути обізнаними з найновішими інструментами генерації тестових даних. Це може спростити їхню роботу та максимізувати ефективність їхніх зусиль з тестування. Ці інструменти пропонують різні функції та можливості. Це дозволяє тестувальникам генерувати різноманітні тестові дані, що охоплюють широкий спектр сценаріїв.

Крім того, використання цих інструментів дозволяє тестувальникам заощаджувати дорогоцінний час і зусилля. В іншому випадку вони були б витрачені на ручне генерування даних. Це дозволяє їм зосередитися на інших критичних аспектах процесу тестування. Завдяки можливостям автоматизації, які пропонують ці інструменти, тестувальники можуть забезпечити послідовне та надійне генерування тестових даних. Це допоможе їм мінімізувати людські помилки та підвищити загальну ефективність.

З огляду на ці міркування, у цій роботі розглянемо деякі з найпопулярніших та найефективніших інструментів генерації тестових даних, доступних сьогодні. Досліджуючи їхні функції, переваги та сфери застосування, тестувальники програмного забезпечення можуть приймати обґрунтовані

рішення та вибирати найкращий інструмент для задоволення своїх потреб у тестуванні.

Генератор тестових даних – це інструмент, призначений для автоматизації процесу створення тестових даних для цілей тестування програмного забезпечення. Він пропонує низку функцій, які дозволяють тестувальникам ефективно генерувати різноманітні та реалістичні набори даних для моделювання різних сценаріїв та умов.

Зокрема, генератори тестових даних дозволяють тестувальникам визначати певні параметри та критерії для генерації даних. Це включає типи даних, діапазони, формати та зв'язки між елементами даних. Ці інструменти забезпечують створення тестових даних, які охоплюють широкий спектр можливостей та адекватно відображають реальні сценарії.

Особливості та переваги генераторів тестових даних.

- Автоматизація. Генератори тестових даних автоматизують процес створення тестових даних, заощаджуючи час і зусилля тестувальників.
- Генерація різноманітних даних. Ці інструменти пропонують різні варіанти для створення різноманітних наборів даних, що охоплюють широкий спектр сценаріїв та умов.
- Налаштування. Вони дозволяють тестувальникам визначати певні параметри та критерії для створення даних, такі як типи даних, діапазони, формати та зв'язки між елементами даних.
- Рандомізація. Вони надають опції рандомізації, що дозволяє генерувати великі обсяги даних з унікальними характеристиками.
- Конфіденційність та безпека даних. Ці інструменти пропонують функції для анонімізації конфіденційної інформації, що гарантує конфіденційність особистих або чутливих даних під час тестування.
- Ефективність навантажувального тестування. Генератори тестових даних особливо корисні для навантажувального тестування та стрес-тестування. Тут необхідно оцінити поведінку системи за великих обсягів даних.

- Реалістичне представлення даних. Вони створюють реалістичні набори даних, які максимально нагадують реальні сценарії. Це підвищує точність та ефективність тестування програмного забезпечення.
- Масштабованість. Ці інструменти можуть генерувати тестові дані у великому масштабі, задовольняючи потреби тестування великомасштабних систем і додатків.
- Повторне використання даних. Генератори тестових даних дозволяють створювати набори тестових даних повторного використання, що дозволяє тестувальникам ефективно проводити повторне або регресійне тестування.

3.2 Властивості найпоширеніших генераторів тестових даних

Проаналізуємо деякі з найкращих інструментів для генерації тестових даних, доступних сьогодні, та розкриємо їхні функції та переваги. Використовуючи ці інструменти, тестувальники можуть підвищити ефективність, точність та надійність своїх зусиль з тестування. Зрештою, це призводить до створення надійних та високопродуктивних програмних продуктів. Ці інструменти можна поєднувати з інструментами керування тестовими даними для створення потужного центру тестування.

Testsigma.

Testsigma – це комплексна платформа автоматизації тестування, яка включає потужну функцію генерації тестових даних (див. рис. 3.1). Вона дозволяє тестувальникам створювати високоякісні та різноманітні тестові дані, що охоплюють різні сценарії, забезпечуючи ретельне тестування програмного забезпечення.

Testsigma надає інтуїтивно зрозумілий інтерфейс, де тестувальники можуть визначати вимоги до тестових даних та генерувати дані відповідно. Він пропонує низку варіантів генерації даних, включаючи випадкові значення, послідовні дані та дані із зовнішніх джерел. Тестери можуть легко налаштувати

згенеровані дані відповідно до конкретних потреб тестування, забезпечуючи точне та надійне покриття тестами.

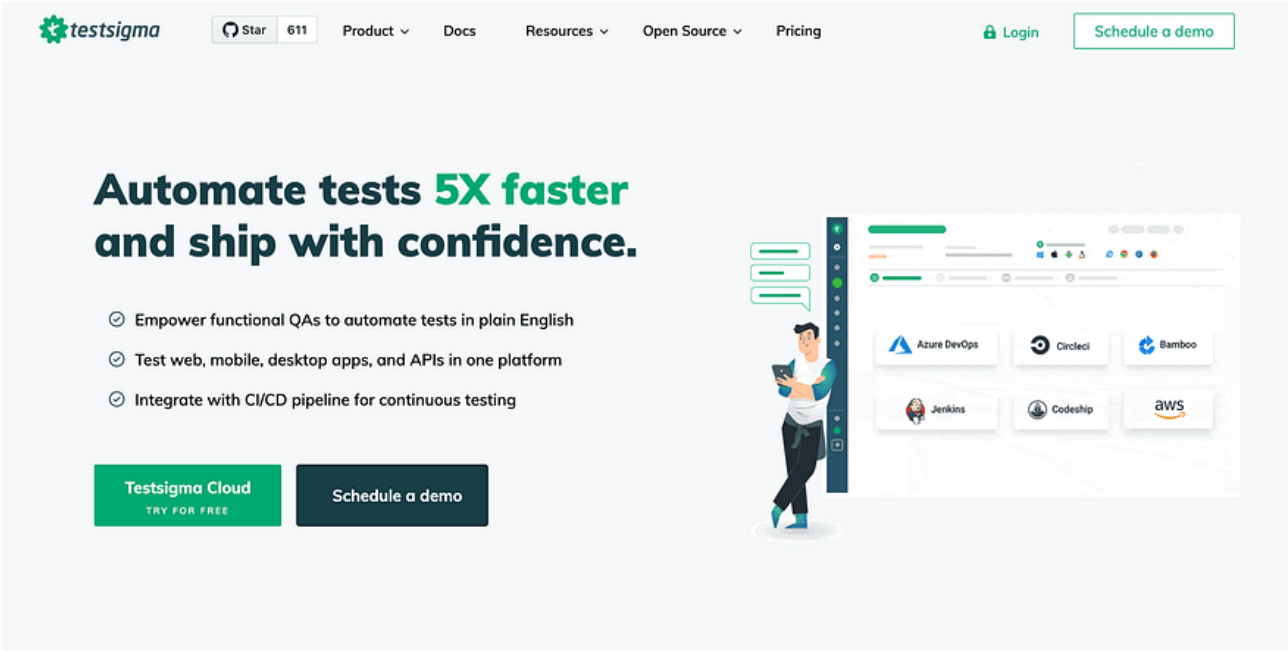


Рисунок 3.1 – Інтерфейс користувача Testsigma

Він має вбудовані функції для параметризованого, тобто керованого даними тестування. Він надає вбудований репозиторій для зберігання, керування та використання тестових даних. Їх можна генерувати або імпортувати за допомогою функцій імпорту даних, які підтримують CSV-файли та бази даних. Але найбільший прискорення випуску забезпечується підтримкою Testsigma паралельного виконання тестів на основі даних, що може допомогти заощадити час і підвищити ефективність тестування.

Особливості Testsigma.

- Пропонує зручний інтерфейс для визначення вимог до тестових даних.
- Надає різні варіанти генерації даних, включаючи випадкові значення, послідовні дані та дані із зовнішніх джерел.
- Дозволяє налаштовувати згенеровані дані відповідно до конкретних сценаріїв тестування.
- Підтримує маскування та анонімізацію даних для захисту конфіденційної інформації під час тестування.

- Бездоганно інтегрується з платформою автоматизації тестування Testsigma, підвищуючи загальну ефективність тестування.

Він доступний як у хмарі з відкритим кодом, так і в преміум-версії.

Mostly AI.

Mostly AI – це інноваційний інструмент для генерації тестових даних, який використовує алгоритми штучного інтелекту та машинного навчання (див. рис. 3.2). Він застосовує їх для створення реалістичних та конфіденційних синтетичних даних. Використовуючи можливості ШІ, тестувальники можуть створювати різноманітні та репрезентативні набори тестових даних. Ці набори даних точно імітують реальні дані, захищаючи при цьому конфіденційну інформацію.

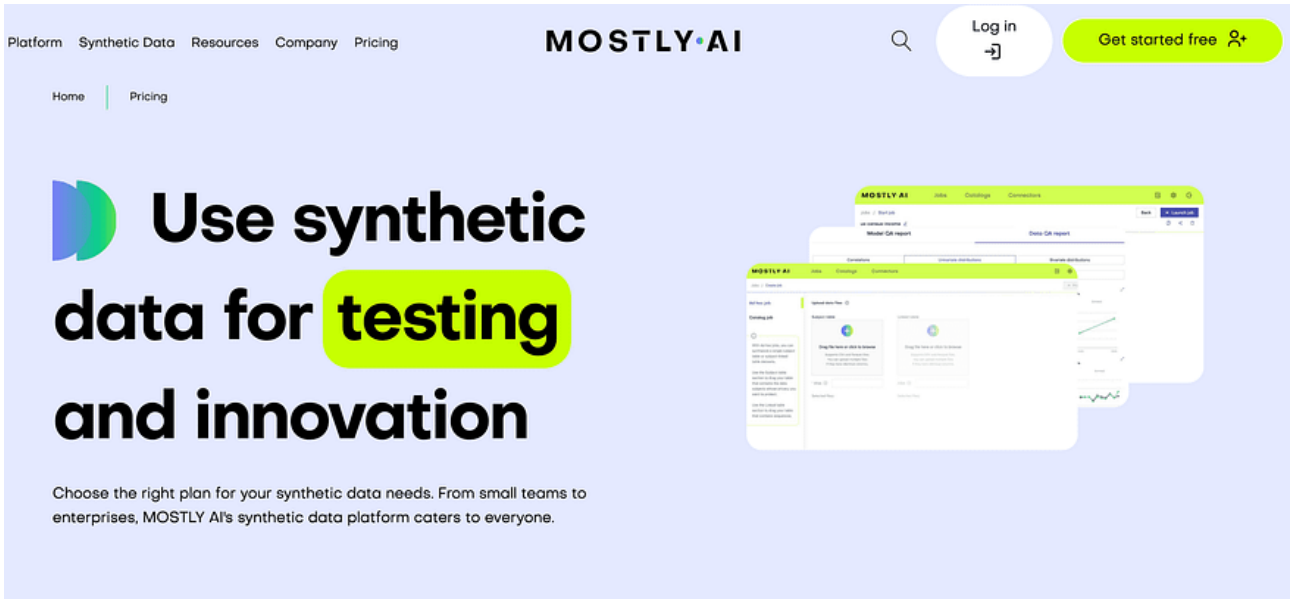


Рисунок 3.2 – Інтерфейс користувача

Він використовує найсучасніші генеративні моделі для створення синтетичних даних, які зберігають статистичні властивості та зв'язки вихідного набору даних.

Особливості програми Mostly AI.

- Генерація синтетичних даних. ШІ здебільшого використовує генеративні моделі для створення реалістичних та репрезентативних

синтетичних даних. Він зберігає статистичні властивості та зв'язки вихідного набору даних.

- Налаштування. Тестери можуть визначати певні атрибути даних, включаючи типи даних, розподіли та кореляції. Це дозволяє їм генерувати адаптовані тестові дані, що відповідають їхнім конкретним вимогам до тестування.
- Збереження конфіденційності. ШІ здебільшого впроваджує методи збереження конфіденційності для анонімізації або приховування конфіденційної інформації в згенерованих тестових даних. Це забезпечує дотримання правил конфіденційності та безпеки даних.
- Масштабованість. Інструмент може ефективно генерувати великі обсяги синтетичних даних. Це робить його придатним для тестових сценаріїв, які потребують значної кількості різноманітних та реалістичних даних.
- Інтеграція. ШІ здебільшого безперешкодно інтегрується з існуючими системами та робочими процесами тестування, що сприяє легкому впровадженню та включенню в процес тестування.

Він має безкоштовну пробну версію, яка дозволяє генерувати до 100 тисяч рядків на день. Він також пропонує командний план за 3,00 долари США та корпоративний план за 5,00 доларів США за кредит.

Datprof.

Datprof – це комплексний інструмент для генерації тестових даних, розроблений для спрощення та оптимізації процесу створення високоякісних та репрезентативних тестових даних (див. рис. 3.3). Він має зручний інтерфейс та розширені функції. Його використання дозволяє тестувальникам генерувати різноманітні набори даних, які точно відображають реальні сценарії, підвищуючи ефективність тестування програмного забезпечення.

Він пропонує різні методи генерації даних, включаючи генерацію на основі правил, шаблонів та випадкову генерацію. Тестери можуть визначати правила та обмеження, щоб гарантувати, що згенеровані дані відповідають певним вимогам та дотримуються заздалегідь визначених шаблонів. Інструмент підтримує

широкий спектр типів даних і може генерувати дані масово. Це дозволяє ефективно тестувати складні системи та програми.

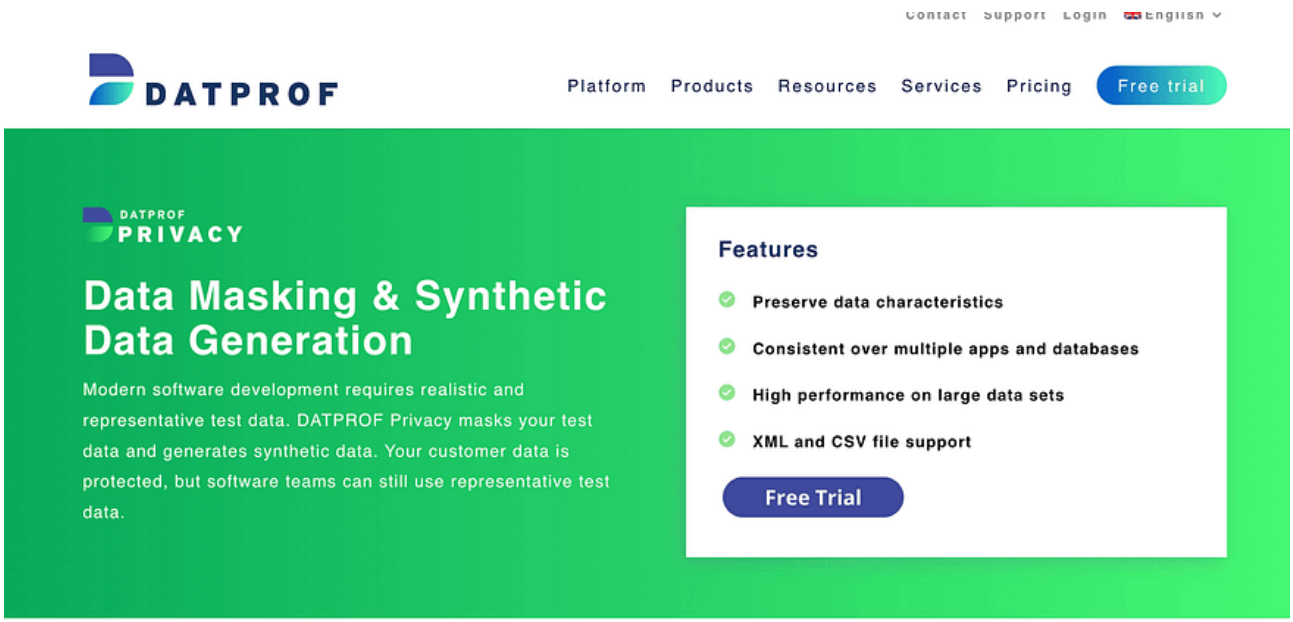


Рисунок 3.3 – Інтерфейс користувача Datprof

Особливості Datprof:

- Генерація на основі правил. Datprof дозволяє тестувальникам визначати правила та обмеження для генерації даних. Це забезпечує відповідність конкретним вимогам та бізнес-логіці.
- Генерація на основі шаблонів. Інструмент підтримує створення даних на основі попередньо визначених шаблонів. Він дозволяє моделювати реалістичні сценарії та варіації даних.
- Випадкова генерація. Datprof пропонує можливості генерації випадкових даних, що дозволяє тестувальникам швидко створювати різноманітні набори даних.
- Генерація масових даних. Інструмент може ефективно генерувати великі обсяги даних, що робить його придатним для тестових сценаріїв, які потребують широкого охоплення даних.

- Маскування даних. Datprof надає функції маскування даних для захисту конфіденційної інформації під час тестування, забезпечуючи конфіденційність та безпеку даних.
- Перевірка даних. Інструмент містить функції перевірки даних для виявлення та позначення будь-яких невідповідностей або помилок у згенерованих тестових даних.

Він має безкоштовну пробну версію з обмеженими можливостями. Вона дозволяє отримати індивідуальну модель ціноутворення залежно від того, які саме функції вам потрібні. Ці функції включають маскування даних, генерацію синтетичних даних, маскування плоских файлів (наприклад, XML, CSV), підгрупування даних, автоматизацію тестових даних, портал тестових даних самообслуговування та виявлення даних.

Data generator EMS.

Data generator EMS – це потужний та універсальний інструмент (див. рис. 3.4). Він розроблений для спрощення процесу генерації тестових даних для тестування баз даних.

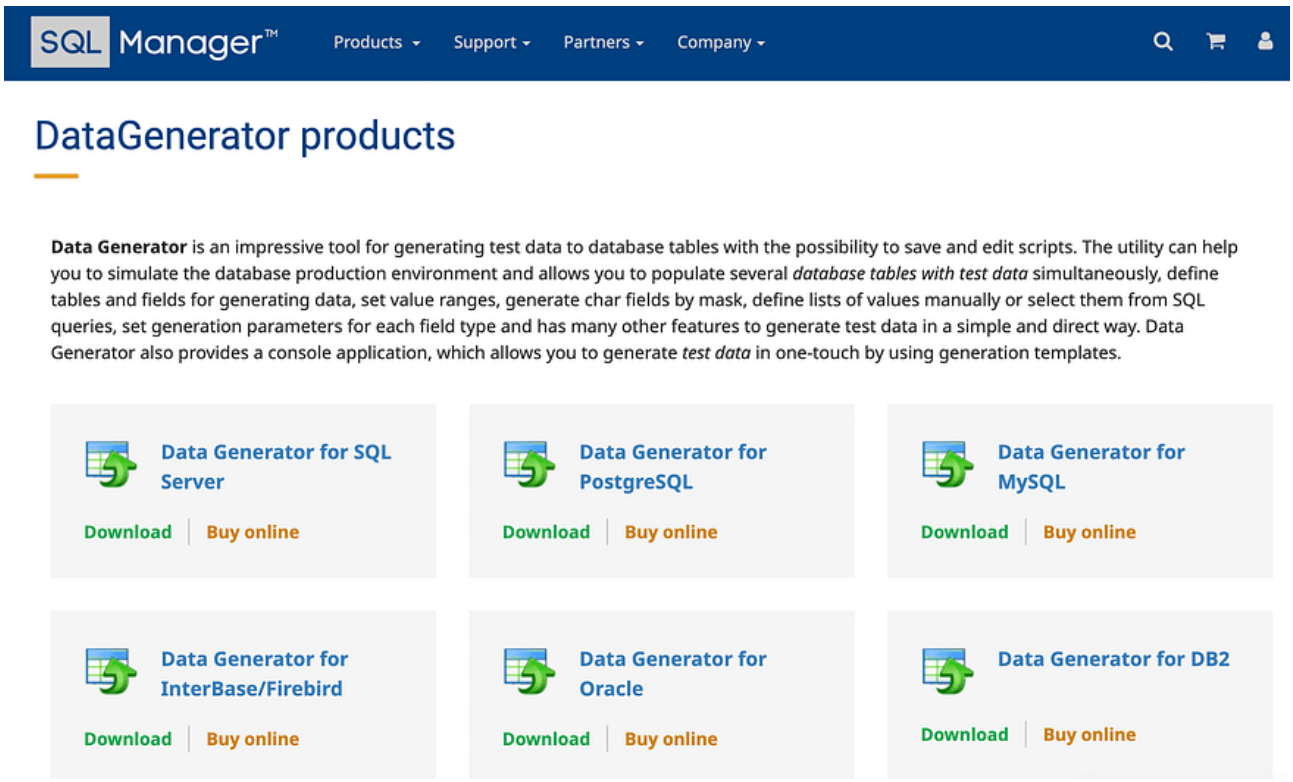


Рисунок 3.4 – Інтерфейс користувача Data generator EMS

Він пропонує повний набір функцій та можливостей. Це дозволяє швидко генерувати великі обсяги реалістичних та настроюваних тестових даних. Це робить його безцінним активом для тестувальників програмного забезпечення та адміністраторів баз даних.

Data generator EMS підтримує різні платформи баз даних і дозволяє тестувальникам визначати власні правила та шаблони генерації даних. Він дозволяє генерувати тестові дані для таблиць, представлень та запитів. Він також має опції для визначення типів даних, діапазонів, зв'язків тощо. Інструмент також надає розширені опції для рандомізації даних, маскування даних та генерації даних на основі SQL-скриптів.

Функції Data generator EMS.

- Підтримка кількох платформ. EMS Data Generator підтримує популярні платформи баз даних, такі як MySQL, PostgreSQL, Oracle, SQL Server та інші.
- Налаштовуване створення даних. Тестери можуть визначати власні правила створення даних, шаблони та зв'язки для створення індивідуальних тестових даних.
- Рандомізація даних. Інструмент пропонує опції рандомізації значень даних, забезпечуючи різноманітні та реалістичні набори даних.
- Маскування даних. EMS Data Generator надає функції маскування даних для захисту конфіденційної інформації під час тестування.
- Генерація SQL-скриптів. Тестери можуть генерувати тестові дані на основі SQL-скриптів, що забезпечує точний контроль та гнучкість.
- Продуктивність та масштабованість. Інструмент може ефективно генерувати великі обсяги тестових даних. Це робить його придатним для тестування складних систем баз даних.

EMS Data Generator дотримується моделі ліцензування, яка базується на конкретному виданні та кількості користувачів. Щоб отримати детальну інформацію про ціни, ви можете звернутися до відділу продажів EMS Data Generator для отримання персоналізованої цінової пропозиції.

SQL RedGate.

Генератор даних Redgate SQL – це потужний інструмент, розроблений для спрощення та автоматизації процесу створення реалістичних та змістовних тестових даних для баз даних SQL (див. рис. 3.5). Він пропонує повний набір функцій та можливостей. Це дозволяє створювати різноманітні та настроювані набори тестових даних. Це дає змогу тестувальникам програмного забезпечення та фахівцям з баз даних підвищити ефективність та точність своїх зусиль з тестування.

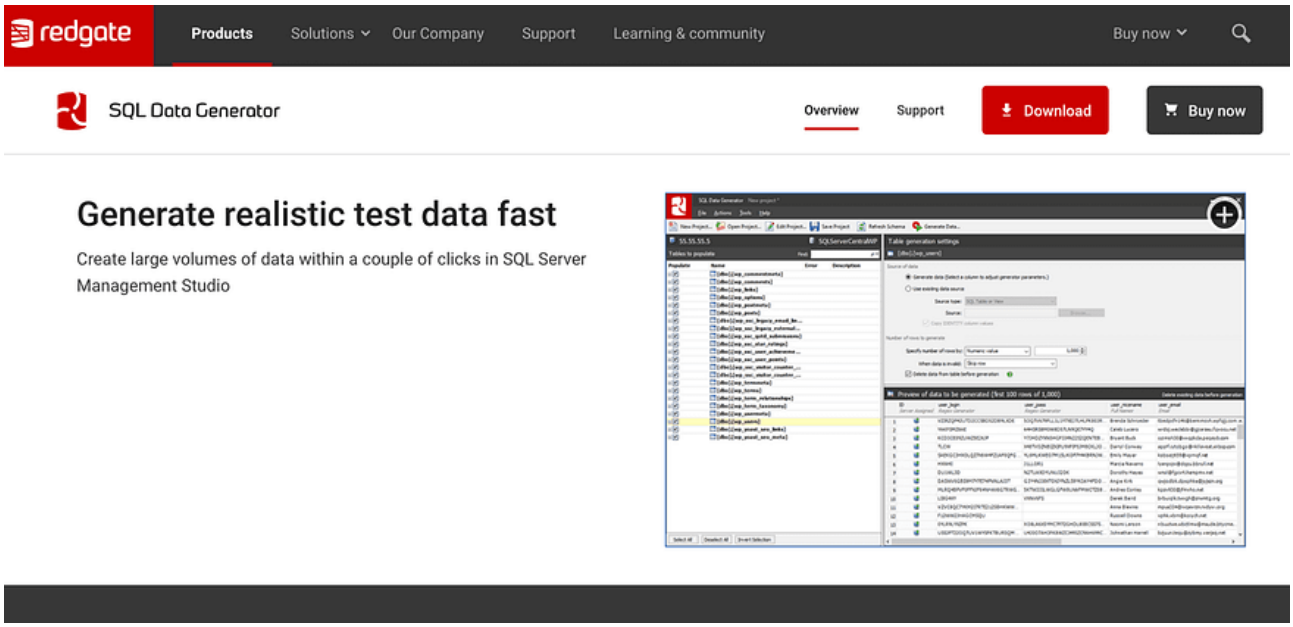


Рисунок 3.5 – Інтерфейс користувача RedGate SQL

Генератор даних Redgate SQL дозволяє користувачам визначати структуру та зв'язки своїх баз даних і відповідно генерувати тестові дані. Він пропонує широкий спектр опцій генерації даних, включаючи випадкові значення, дані на основі словника, користувацькі скрипти тощо. Інструмент також забезпечує гнучкість у визначенні обмежень даних. Він пропонує розширені можливості маскування даних для захисту конфіденційної інформації під час тестування.

Функції генератора даних Redgate SQL.

- Генерація з урахуванням бази даних: аналізує структуру бази даних та інтелектуально генерує дані, що відповідають зв'язкам, обмеженням та типам даних у таблицях.

- Генерація різноманітних даних. Redgate SQL Data Generator пропонує кілька варіантів генерації даних. Це включає випадкові дані, дані на основі словника, користувацькі скрипти та послідовні дані.
- Налаштування та обмеження. Користувачі можуть визначати конкретні обмеження та правила для створення даних.
- Маскування даних. Інструмент надає функції маскування даних для анонімізації або приховування конфіденційної інформації, захищаючи конфіденційність даних під час тестування.
- Продуктивність та масштабованість. Redgate SQL Data Generator може ефективно генерувати великі обсяги тестових даних. Це робить його високомасштабованим та зручним для складних баз даних.
- Інтеграція. Інструмент легко інтегрується з популярними системами керування базами даних. Його можна легко включити в існуючі робочі процеси розробки та тестування баз даних.

Redgate SQL Data Generator дотримується моделі ліцензування, яка може сягати 264 доларів за ліцензію на рік.

DTM Data Generator.

DTM Data Generator – це багатофункціональний інструмент, розроблений для полегшення створення реалістичних та різноманітних тестових даних для баз даних, електронних таблиць та інших джерел даних (див. рис. 3.6). Він має повний набір функцій. DTM Data Generator дозволяє тестувальникам програмного забезпечення та фахівцям з баз даних ефективно створювати високоякісні тестові дані, які точно імітують реальні сценарії.

Він пропонує зручний інтерфейс, де користувачі можуть визначати правила генерації даних та налаштовувати різні параметри. Інструмент підтримує різні методи генерації даних, включаючи випадкову генерацію, послідовну генерацію та генерацію на основі діапазону значень. Він також надає опції для налаштування даних, такі як форматування, маскування даних та зв'язки зовнішніх ключів.

Функції DTM Data Generator.

- Підтримка кількох джерел: Підтримує різноманітні джерела даних, включаючи бази даних (SQL Server, Oracle, MySQL тощо), електронні таблиці (Excel, CSV) тощо.

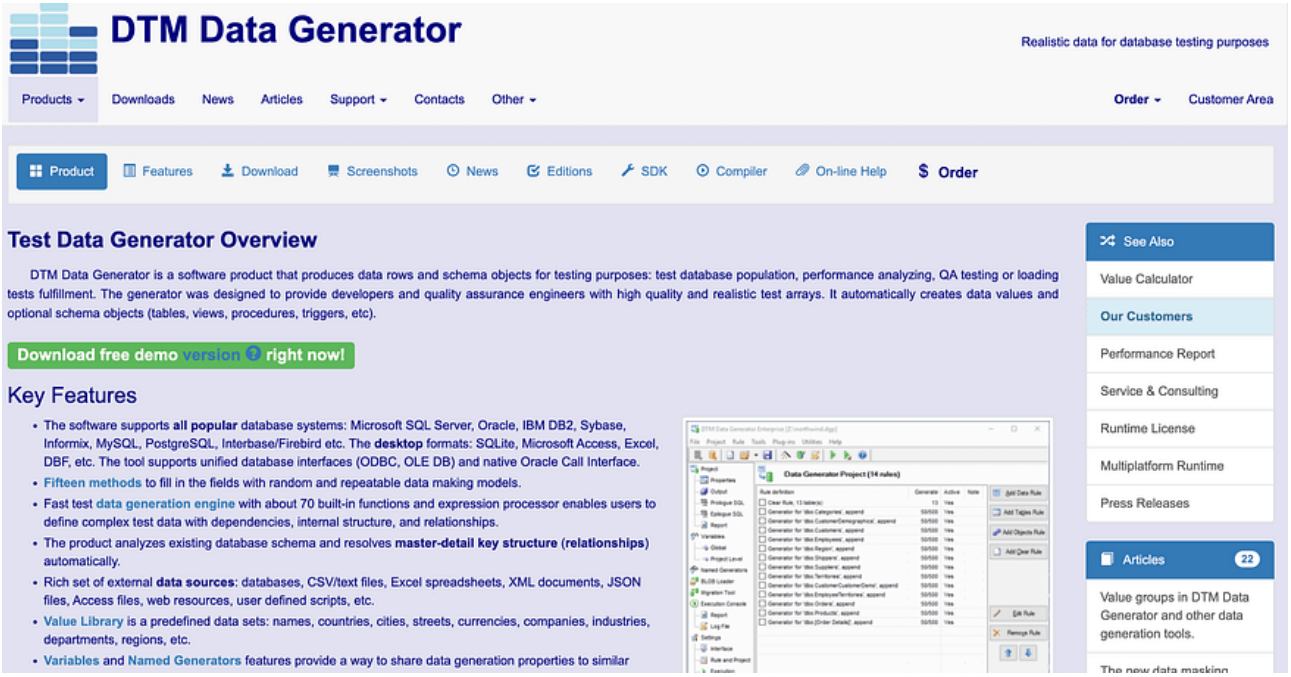


Рисунок 3.6 – Інтерфейс користувача DTM Data Generator

- Гнучкі методи генерації даних: Інструмент пропонує кілька методів генерації даних. Це дозволяє користувачам генерувати дані випадковим чином, послідовно або на основі попередньо визначених діапазонів значень.
- Налаштування даних: дозволяє користувачам налаштовувати згенеровані дані, вказуючи параметри форматування. Вони також можуть застосовувати методи маскування даних та встановлювати зв'язки між таблицями за допомогою зовнішніх ключів.
- Генерація масових даних: Інструмент ефективно генерує великі обсяги тестових даних, що робить його придатним для сценаріїв тестування, які потребують широкого охоплення даних.
- Перевірка та узгодженість даних: Генератор даних DTM також включає функції перевірки даних. Це гарантує, що згенеровані дані відповідають заздалегідь визначеним правилам та підтримують узгодженість з моделлю даних.

- Експорт та інтеграція: Інструмент надає можливості експорту згенерованих даних у різні формати та легко інтегрується з популярними системами управління базами даних.

DTM Data Generator пропонує різні версії з різними функціями та можливостями. Щоб отримати детальну інформацію про ціни, ви можете відвідати веб-сайт генератора даних DTM або зв'язатися з його відділом продажів.

Mockaroo.

Mockaroo – це універсальний та зручний інструмент для генерації тестових даних, який дозволяє тестувальникам та розробникам програмного забезпечення створювати реалістичні та настроювані макетні дані (див. рис. 3.7). Завдяки інтуїтивно зрозумілому інтерфейсу та широкому набору функцій, Mockaroo спрощує процес генерації тестових даних, дозволяючи користувачам визначати структури даних, вказувати типи полів та швидко генерувати великі обсяги різноманітних тестових даних.

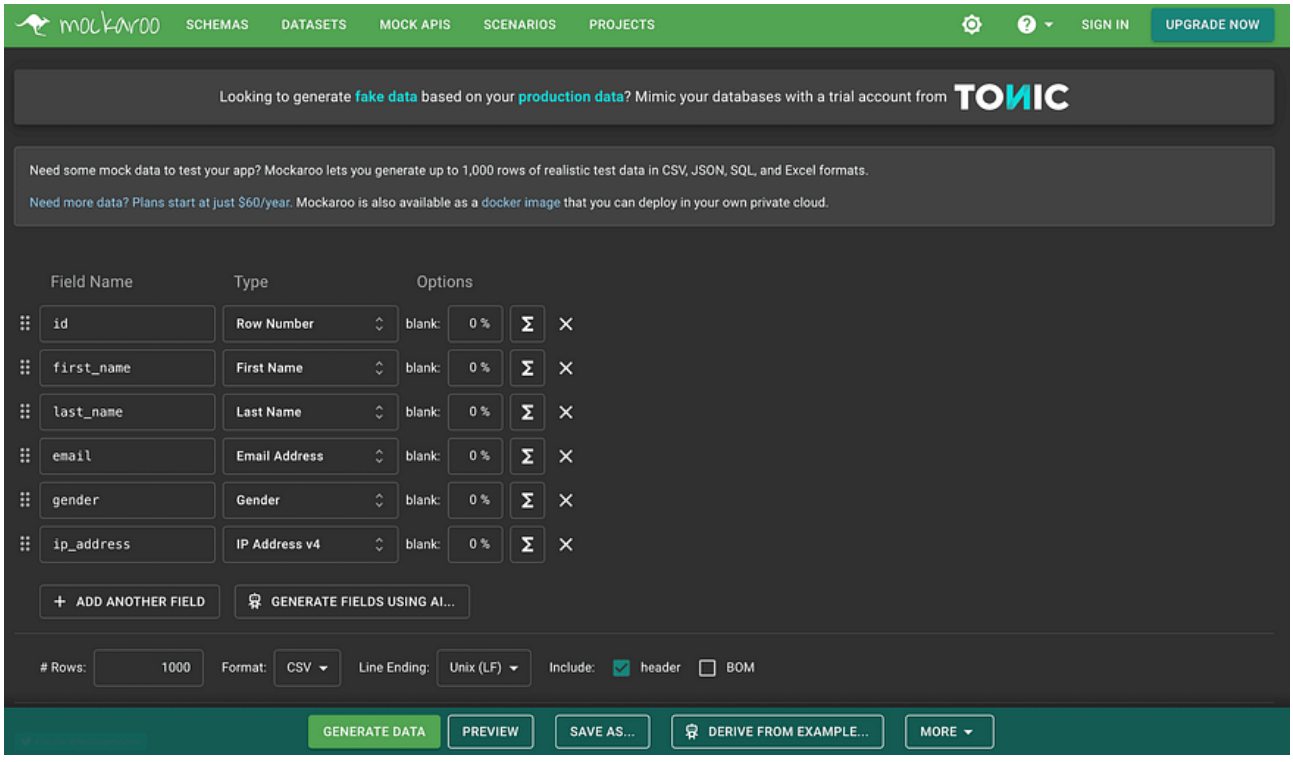


Рисунок 3.7 – Інтерфейс користувача Mockaroo

Він пропонує візуальний редактор схем, де користувачі можуть визначати структуру та характеристики своїх тестових даних. Він підтримує широкий спектр типів даних, включаючи рядки, числа, дати та власні формати. Користувачі можуть налаштовувати різні параметри, такі як обмеження, шаблони та розподіли, для створення даних, які максимально нагадують реальні сценарії. Інструмент також надає опції експорту згенерованих даних у різні формати, включаючи CSV, JSON, SQL та Excel.

Особливості Mockaroo.

- Редактор візуальних схем: Mockaroo надає зручний інтерфейс для визначення структур даних та налаштування типів полів, обмежень та зв'язків.
- Генерація різноманітних даних: Інструмент підтримує широкий спектр типів даних і надає опції для генерації реалістичних даних, включаючи імена, адреси, номери телефонів тощо.
- Налаштовуване створення даних: Користувачі можуть вказувати обмеження, шаблони та розподіли для створення тестових даних, що відповідають конкретним вимогам тестування.
- Генерація масових даних: Mockaroo дозволяє користувачам швидко генерувати великі обсяги тестових даних, що робить його придатним для сценаріїв тестування, які потребують великих наборів даних.
- Експорт даних: Інструмент надає опції експорту згенерованих даних у різних форматах, що дозволяє безперешкодно інтегруватися з різними фреймворками та інструментами тестування.
- Перевірка даних: Mockaroo пропонує функції перевірки даних, щоб гарантувати, що згенеровані дані відповідають заздалегідь визначеним правилам та обмеженням.

Він має безкоштовний рівень, обмежений 1000 рядками на файл, та срібний рівень вартістю 60 доларів на рік зі 100 тисячами рядків на файл. Він також пропонує золотий та корпоративний рівні за 500 доларів та 7500 доларів на рік відповідно.

GenerateData.

GenerateData.com пропонує простий та зручний підхід до створення тестових даних. Користувачі можуть визначити кількість рядків і стовпців для свого набору даних, а також вказати типи даних і характеристики кожного поля. Інструмент підтримує широкий спектр типів даних, включаючи текст, числа, дати тощо. Користувачі також можуть застосовувати власні формули, генерувати випадкові дані та включати умовну логіку для створення складніших наборів даних.

Можливості GenerateData.

- Простий у використанні інтерфейс: GenerateData пропонує простий та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко визначати структури даних та генерувати тестові дані без необхідності складних конфігурацій.
- Підтримка типів даних: Інструмент підтримує різні типи даних, включаючи текст, числа, дати, адреси електронної пошти тощо, забезпечуючи гнучкість у створенні різноманітних наборів даних.
- Параметри налаштування: Користувачі можуть застосовувати власні формули, вказувати формати полів і визначати обмеження даних для створення тестових даних, які відповідають певним вимогам тестування.
- Генерація масових даних: GenerateData дозволяє генерувати великі обсяги тестових даних, що робить його придатним для сценаріїв тестування, які потребують великих наборів даних.
- Рандомізація та варіація: Інструмент дозволяє рандомізувати значення даних, забезпечуючи варіативність та реалістичність згенерованих наборів даних.
- Експорт та інтеграція: GenerateData надає можливості експортувати згенеровані дані в різних форматах, що сприяє безперешкодній інтеграції з різними фреймворками та інструментами тестування.

GenerateData пропонує як безкоштовні, так і платні плани підписки. Безкоштовний план надає базові функції, тоді як платні плани пропонують

додаткові можливості, такі як збільшений розмір набору даних, підтримка пріоритетів та доступ до розширених функцій.

Upscene – розширений генератор даних.

Upscene Database Workbench – це комплексний набір інструментів, призначений для розробки, тестування та адміністрування баз даних (див. рис. 2.8). Серед багатьох функцій Database Workbench включає потужний компонент генерації тестових даних, який допомагає тестувальникам програмного забезпечення створювати реалістичні та настроювані набори даних для тестування баз даних.

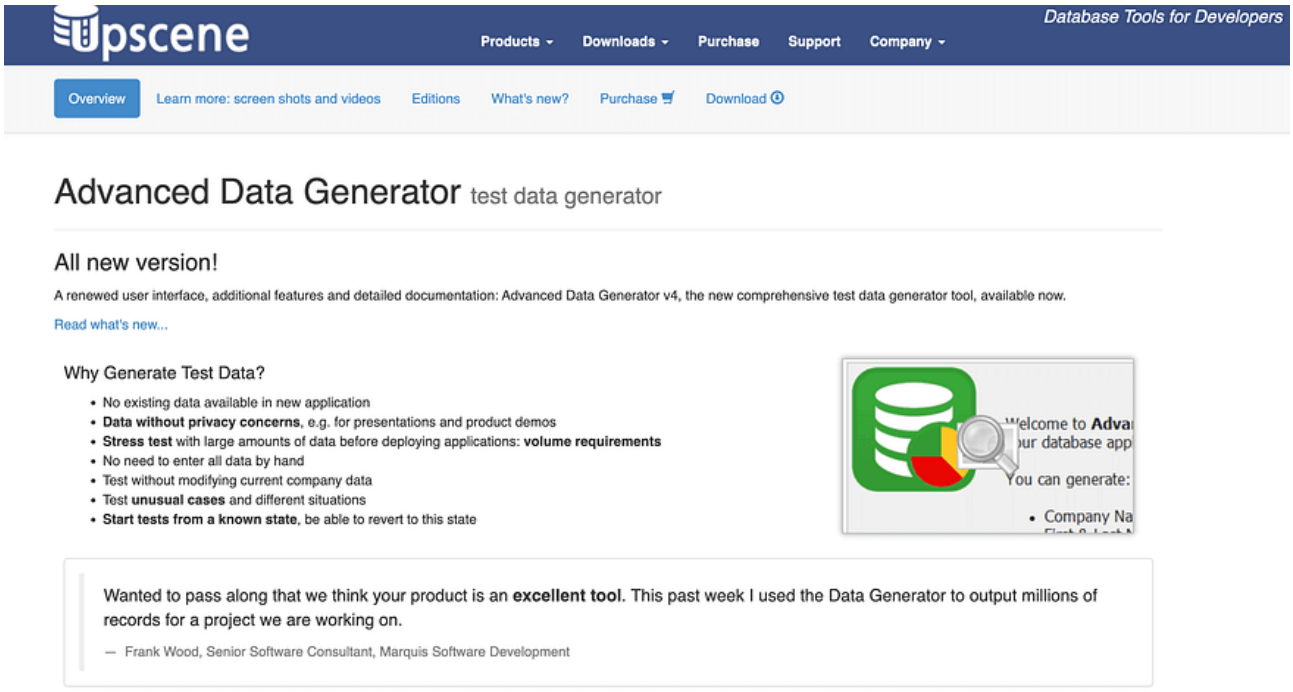


Рисунок 3.8 – Інтерфейс користувача Upscene Database Workbench

Upscene Advanced Data Generator надає зручний інтерфейс для визначення правил генерації тестових даних та налаштування різних параметрів. Інструмент підтримує різні методи генерації даних, включаючи випадкову генерацію, послідовну генерацію та генерацію на основі діапазону значень. Він пропонує широкі можливості для налаштування генерації даних, такі як типи даних, обмеження та зв'язки.

Особливості розширеного генератора даних Upscene.

- Генерація тестових даних: Upscene пропонує спеціальний компонент генерації тестових даних для створення різноманітних та реалістичних наборів даних для тестування баз даних.
- Налаштовуване створення даних. Інструмент дозволяє користувачам визначати правила створення даних, вказувати типи даних та застосовувати обмеження для створення тестових даних, які відповідають певним вимогам тестування.
- Генерація зв'язків. Дозволяє створювати зв'язки між таблицями, забезпечуючи цілісність посилань у згенерованих тестових даних.
- Перевірка даних. Інструмент містить функції перевірки даних для виявлення та позначення будь-яких невідповідностей або помилок у згенерованих тестових даних.
- Генерація масових даних. Він може ефективно генерувати великі обсяги тестових даних, що робить його придатним для сценаріїв тестування, які потребують широкого охоплення даних.
- Підтримка баз даних. Інструмент підтримує різні системи керування базами даних, включаючи Oracle, Microsoft SQL Server, MySQL та інші, що дозволяє безперешкодно інтегруватися в існуючі середовища баз даних.

Upscene Database Workbench працює за комерційною моделлю ліцензування. Його ціна може сягати 210 доларів на рік залежно від необхідних вам налаштованих функцій.

Solix EDMS (Enterprise Data Management Suite).

Solix EDMS (Enterprise Data Management Suite) – це комплексне рішення, розроблене для управління корпоративними даними протягом усього їхнього життєвого циклу, включаючи архівування даних, управління даними та безпеку даних (див. рис. 3.9). У своєму наборі функцій Solix EDMS пропонує надійні можливості для генерації тестових даних, надаючи організаціям засоби для ефективного створення реалістичних та безпечних наборів тестових даних для цілей тестування та розробки програмного забезпечення.

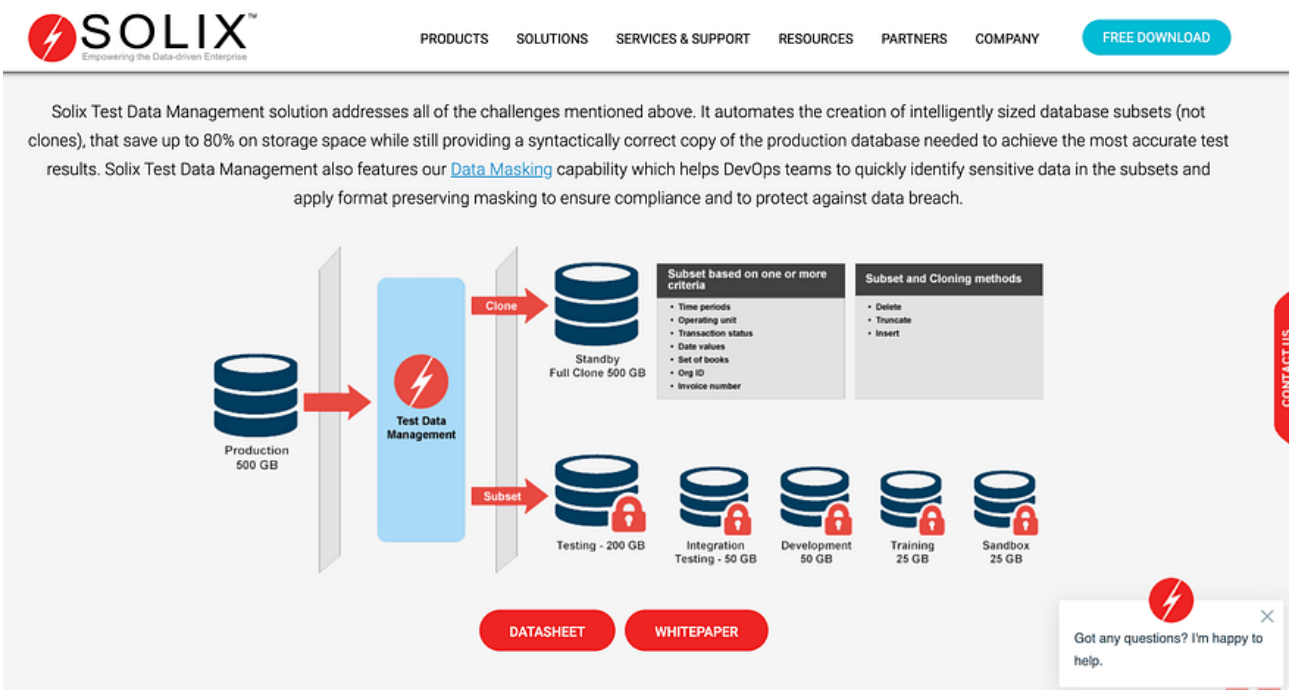


Рисунок 3.9 – Інтерфейс користувача Solix EDMS

Solix EDMS спрощує процес генерації тестових даних завдяки своєму інтуїтивно зрозумілому інтерфейсу та потужним функціям. Інструмент дозволяє користувачам визначати правила генерації даних, вказувати типи даних та обмеження, а також генерувати тестові дані, які максимально нагадують робочі середовища. Використовуючи Solix EDMS, тестувальники можуть створювати репрезентативні набори даних, що охоплюють широкий спектр сценаріїв, сприяючи комплексному та ефективному тестуванню програмного забезпечення.

Особливості системи управління системою Solix EDMS.

- Налаштовуване створення даних. Solix EDMS надає користувачам можливість визначати власні правила створення даних, включаючи певні типи даних, формати та обмеження, гарантуючи відповідність згенерованих тестових даних вимогам програми.
- Маскування даних. Інструмент пропонує розширені можливості маскування даних для приховування конфіденційної інформації, забезпечуючи конфіденційність та безпеку даних під час тестування.

- Підмножина та вибірка. Solix EDMS дозволяє користувачам витягувати підмножини даних з виробничих середовищ, що дозволяє створювати цільові набори тестових даних, що зосереджені на певних сценаріях або модулях.
- Профілювання даних. Solix EDMS надає функції профілювання даних для аналізу та розуміння характеристик даних, гарантуючи, що згенеровані тестові дані точно відображають розподіл та закономірності вихідних даних.
- Збереження зв'язків між даними. Інструмент підтримує зв'язки між даними під час генерації тестових даних, забезпечуючи цілісність посилань та дозволяючи точне тестування реляційних систем баз даних.
- Оновлення та версіонування даних. Solix EDMS дозволяє оновлювати дані та версіонувати їх, що дозволяє тестувальникам легко оновлювати та керувати наборами даних тестування в міру розвитку програми.

Solix EDMS пропонує гнучкі варіанти розгортання та моделі ліцензування, адаптовані до конкретних потреб організацій. Для отримання детальної інформації про ціни та додаткові функції ви можете відвідати веб-сайт Solix EDMS або зв’язатися з їхньою командою продажів.

3.3 Програмний прототип генератора тестових даних на основі ШІ

Запропонуємо прототип інтелектуальної багатоагентної системи для генерації реалістичних тестових даних JSON за допомогою механізму робочого процесу LangGraph4j і кількох моделей ШІ.

Тестування сучасних програм вимагає реалістичних, різноманітних тестових даних, які суворо відповідають схемам JSON. Створення даних вручну займає багато часу та часто призводить до створення масивних, нереалістичних наборів даних. Цей агент вирішує ці проблеми.

Властивості пропонованого генератора.

Інтелектуальна генерація – генератор використовує штучний інтелект для створення реалістичних тестових даних з урахуванням локалі.

Відповідність схеми – сувора перевірка на відповідність схемі JSON.

Самовідновлення – автоматично виявляє та виправляє помилки перевірки.

Політика проти заповнювачів – активно уникає підроблених даних, таких як "John Doe", "123-456", "test@example.com".

Locale-Aware – генерує реалістичні імена, адреси, номери телефонів, податкові ідентифікаційні номери

Мультимодельний – направляє різні завдання до оптимальних моделей AI (генерація проти перевірки).

Система використовує LangGraph4j для оркестрування кількох спеціалізованих агентів.

Schema Validator: перевіряє сумісність схеми JSON.

Generation Planner: створює стратегії генерації даних.

JSON Generator: створює вихідні дані, сумісні зі схемою.

Normalizer: очищає та стандартизує вихід.

Validator: сувора перевірка відповідності схеми.

Decision Router: інтелектуальне визначення наступного кроку.

Fix Planner: аналізує помилки перевірки.

Error Fixer: застосовує цілеспрямовані виправлення.

Таблиця 3.1 містить опис різних моделей ШІ, що використовуються в роботі генератора тестових даних.

Таблиця 3.1 – Моделі ШІ для розроблюваного генератора тестових даних

Тип завдання	**Контекст локалізації**	**Контекст інтернаціоналізації**	**Обгрунтування**
JSON Generation	GigaChat	Claude-3.5-Sonnet	Підтримка кириличних мов
Validation & Analysis	DeepSeek-R1 (free)	Claude-3.5-Sonnet	Вибір на користь безкоштовного інструменту
Planning & Strategy	DeepSeek-R1 (free)	Claude-3.5-Sonnet	Можливість багатоетапного планування
Decision Making	DeepSeek-R1 (free)	GPT-4	Логіка бінарних рішень

Маршрутизацію моделей ШІ показано на рисунку 3.10.

```yaml
ai:
  model-routing:
    nodes:
      # CIS-focused generation with GigaChat-2-Pro
      GenerateJsonNode: "GigaChat-2-Pro"
      FixValidationErrorsInJsonTool: "GigaChat-2-Pro"

      # Cost-effective reasoning with free DeepSeek-R1
      ThinkHowToGenerateTool: "deepseek/deepseek-r1"
      ReasonAndRouteNode: "deepseek/deepseek-r1"

      # Alternative options
      # ThinkHowToGenerateTool: "anthropic/claude-4.1-sonnet"
      # ReasonAndRouteNode: "anthropic/claude-4.1-sonnet"
```

Рисунок 3.1 – Маршрутизація моделей ШІ

Встановлення значень змінних оточення та налаштування моделей ШІ показано на рисунках 3.11 – 3.13.

```
# Application
SERVER_PORT=8080
LOGGING_LEVEL_GITHUB_AI_QA_SOLUTIONS=INFO

# Spring Boot
SPRING_PROFILES_ACTIVE=gigachat-openrouter
SPRING_APPLICATION_NAME=ai-test-data-generation-agent
```

Рисунок 3.11 – Змінні оточення програми

```
SPRING_AI_GIGACHAT_CHAT_ENABLED=true
SPRING_AI_GIGACHAT_CHAT_TOKEN=your_gigachat_api_token
SPRING_AI_GIGACHAT_CHAT_OPTIONS_MODEL=GigaChat-2-Pro  # or GigaChat-2-Max
SPRING_AI_GIGACHAT_CHAT_OPTIONS_TEMPERATURE=0.1
SPRING_AI_GIGACHAT_CHAT_OPTIONS_MAX_TOKENS=4000
SPRING_AI_GIGACHAT_CHAT_AUTHORIZATION_TYPE=Bearer
```

Рисунок 3.12 – Змінні оточення GigaChat

```
SPRING_AI_OPENAI_CHAT_ENABLED=true
SPRING_AI_OPENAI_API_KEY=sk-or-v1-your-openrouter-key
SPRING_AI_OPENAI_BASE_URL=https://openrouter.ai/api/v1
# Free models for cost-effective reasoning
SPRING_AI_OPENAI_CHAT_OPTIONS_MODEL=deepseek/deepseek-r1
# Premium alternatives: anthropic/claude-3.5-sonnet, openai/gpt-4
SPRING_AI_OPENAI_CHAT_OPTIONS_TEMPERATURE=0.1
SPRING_AI_OPENAI_CHAT_OPTIONS_MAX_TOKENS=4000
```

Рисунок 3.13 – Змінні оточення OpenRouter

Розроблена система автоматично виявляє та опрацьовує декілька версії схеми JSON:

- Draft 4 (default fallback).
- Draft 6.
- Draft 7.
- Draft 2019-09
- Draft 2020-12 (найсучасніша стабільна версія).

Генератор даних можна розгортати в докері і доступний інтефейс командної стрічки чи можна розробити веб-інтерфейс.

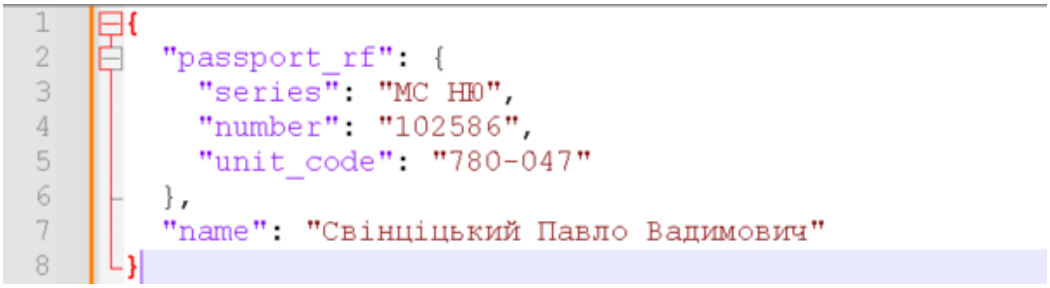
Робота генератора здійснюється шляхом постановки запиту у вигляді фрази людської мови. Відповідь виводиться у вигляді запису JSON. Приклад відповіді на запит для генерації паспортних даних разом зі схемою JSON показано на рисунку 3.14 – 3.16.

```
1
2
3  Generate realistic data for a Ukrainian passport holder from Ternopil
4
```

Рисунок 3.13 – Приклад запиту

```
{
  "type": "object",
  "properties": {
    "passport_rf": {
      "type": "object",
      "properties": {
        "series": {
          "type": "string",
          "pattern": "^\\d{2} \\d{2}$"
        },
        "number": {
          "type": "string",
          "pattern": "^\\d{6}$"
        },
        "unit_code": {
          "type": "string",
          "pattern": "^\\d{3}-\\d{3}$"
        }
      },
      "required": [
        "series",
        "number",
        "unit_code"
      ]
    },
    "name": {
      "type": "string",
      "minLength": 1
    }
  },
  "required": [
    "passport_rf",
    "name"
  ]
}
```

Рисунок 3.14 – Опис JSON-схеми



```
{
  "passport_rf": {
    "series": "МС НЮ",
    "number": "102586",
    "unit_code": "780-047"
  },
  "name": "Свінціцький Павло Вадимович"
}
```

Рисунок 3.15 – Згенеровані дані

Програмний код мовою Java наведений в додатках до роботи.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Питання щодо охорони праці

Тема кваліфікаційної роботи магістра пов’язана із дослідженням методів і засобів тестування програмного забезпечення комп’ютерних систем з використанням алгоритмів машинного навчання. Такі роботи передбачають використання комп’ютерної техніки на етапах формування пояснювальної записки, налаштування засобів автоматизації процесу управління тестуванням з використанням алгоритмів машинного навчання. Тому, при виконанні робіт з тестування програмних складових комп’ютерних систем необхідно враховувати вимоги з охорони праці при експлуатації комп’ютерної техніки.

В Україні діють ряд законів, нормативних документів та актів, які регулюють процеси забезпечення та управління охороною праці у різних галузях народного господарства. До них належать: Конституція України, Закони України "Про охорону праці", "Про охорону здоров'я", "Про пожежну безпеку", "Про забезпечення санітарного та епідемічного благополуччя населення", "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності", Кодекс законів про працю України (КЗпП).

Однією з основних вимог до приміщень, де робочі місця обладнані комп’ютерною технікою і планується використання програмного комплексу для забезпечення процесу тестування ПЗ, є вимоги щодо площі, яка відводиться на один ПК. При проектуванні автоматизованих робочих місць тестувальників програмного забезпечення необхідно дотримуватись вимог щодо розміщення комп’ютерів. На один ПК передбачено площу 6 м^2 та об’єм 20 м^3.

Однак робота з комп’ютером включає різні завдання, які об'єднуються такими загальними чинниками, як те, що робота проводиться в сидячому положенні і вимагає уважного, неперервного та іноді тривалого спостереження.

Перше правило, якого варто дотримуватись тестувальникам програмного забезпечення стосується правильного облаштування робочого столу. При цьому слід передбачити наступні його параметри: фіксована висота – 720 мм, забезпечення необхідного простору для рук по висоті, ширині і глибині, в області сидіння не повинно бути шухляд.

Друге правило визначає облаштування робочого стільця: можливість регулювання висоти стільця, забезпечення обертання конструкції стільця. У приміщеннях з ПК, на яких планується виконання задач з тестування програмних складових комп'ютерних систем, яскравість знаків і яскравість фону дисплею повинна бути спроектована таким чином, щоб не було великої відмінності з яскравістю навколишнього середовища, але знаки повинні чітко розпізнаватися на відстані читання. Характеристики освітлення, зокрема у приміщеннях, де експлуатується ПК, повинні відповідати ДБН В.2.5-28-2006 "Природнє і штучне освітлення". Основні вимоги даного нормативного документу стосуються забезпечення наступних вимог:

- освітлення з лівої сторони;
- рівномірне освітлення всього робочого простору;
- комп'ютерна техніка встановлюється у місцях, віддалених від вікон;
- встановлення непрямого штучного освітлення;
- світло, що поступає через вікна, «пом'якшують» за допомогою штор;
- робоче місце організовується так, щоб напрям погляду був паралельним фронту вікон.

Ще одне правило, якого слід дотримуватись тестувальникам програмного забезпечення, передбачає оптимальний метод роботи, що полягає у передбачені зміни завдань і навантажень, дотримання перерви в роботі: 5 хвилин через 1 годину роботи біля дисплея або 10 хвилин після 2-х годин роботи біля дисплея. Вимоги цього правила регламентовані нормативним документом «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електроннообчислювальних машин».

При створенні сприятливих умов для підвищення продуктивності і зменшення напруги значну роль грають чинники, що характеризують стан навколишнього середовища: мікроклімат приміщення, рівень шуму і освітлення.

Рекомендована величина відносної вологості, яка повинна бути забезпечена у приміщеннях з експлуатації програмного комплексу поведінкового тестування програмних складових комп’ютерних систем, повинна відповідати НПАОП 0.007.15-18 і становити 65 – 70%. При цьому робоче місце повинно бути добре вентильованим.

У даний час з погляду шумового навантаження досягнуто значного прогресу. Рівень шуму в приміщеннях (приблизно 40 Дб) не перевищує допустимого рівня, незалежно від кількості використовуваного обладнання. Для приміщень, в яких експлуатується програмний комплекс підтримки запропонованих методів поведінкового тестування, потрібно забезпечити виконання вимог пожежної безпеки, які визначені Правилами пожежної безпеки в Україні, НПАОП 0.00-7.1518 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Будівлі і ті їх частини, в яких розташовуються ПК можуть належати до II ступеня вогнестійкості. Над та під приміщеннями, де розташовуються ПК, а також у суміжних з ними приміщеннях не дозволяється розташування приміщень категорій А і Б за вибухопожежною небезпекою. Приміщення категорії В повинні бути відділеними від приміщень з ПК протипожежними стінами.

Таким, чином при дослідженні методів і засобів поведінкового тестування програмних складових комп’ютерних систем, встановлено, що найбільш повним нормативним документом щодо охорони праці користувачів ПК, до яких належать тестувальники програмного забезпечення, є НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Дотримання вимог, які неведені у цьому документі, сприяє зниженню негативного впливу ПК, його компонентів та інших зовнішніх

пристроїв на тестувальників, які проводять роботи щодо перевірки правильності функціонування програмних складових комп’ютерних систем.

4.2 Підвищення стійкості роботи об'єктів господарської діяльності у воєнний час

На основі вивчення факторів, які впливають на стійкість роботи об'єктів господарської діяльності, та оцінки стійкості елементів і галузей виробництва проти уражаючих факторів ядерної, хімічної і біологічної зброї, стихійних лих і виробничих аварій, необхідно завчасно організувати і провести організаційні, інженерно-технічні й технологічні заходи для підвищення стійкості роботи.

Здійснення організаційних заходів передбачає завчасну підготовку всіх структур цивільного захисту, служб і формувань до надзвичайних ситуацій, в тому числі і військових дій.

Вжиттям технологічних заходів підвищується стійкість роботи об'єктів шляхом змінювання технологічних процесів, режимів, можливих в умовах різних надзвичайних ситуацій.

Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу уражаючих факторів під час військових дій.

При проведенні цих заходів необхідно враховувати конкретні умови об'єкта народного господарства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

Одним з найбільш важливих завдань в умовах воєнного часу і надзвичайних ситуацій є забезпечення захисту людей та їх життєдіяльності.

Для підвищення стійкості об’єктів господарювання та захисту людей необхідно:

- створити на об'єкті надійну систему оповіщення про загрози нападу противника, радіоактивне забруднення, хімічне і біологічне зараження, загрозу стихійного лиха і виробничої аварії.

• організувати розвідку і спостереження за радіоактивним забрудненням, хімічним і біологічним зараженням;

• організувати гідрометеорологічне спостереження за рівнем води, напрямком і швидкістю вітру, рухом і поширенням хмари радіоактивного забруднення, сильнодіючих отруйних речовин і отруйних речовин.

• створити фонд захисних споруд ЦО, запасів засобів індивідуального захисту і забезпечення своєчасної видачі їх населенню.

• завчасно підготуватись до масової санітарної обробки населення і знезаражування одягу;

• організувати взаємодію з установами охорони здоров'я для медичного обслуговування населення в умовах воєнного часу.

Також в умовах воєнного часу необхідно провести підготовку до евакуації населення, розміщеного в зонах можливих руйнувань і катастрофічного затоплення. Це передбачає завчасну підготовку місць евакуації, організацію прийому евакуйованого населення на територію населених пунктів.

Окрім цього, необхідно забезпечити постачання продуктів харчування, питної води, предметів першої необхідності та провести заходи щодо морально-психологічної підготовки населення до виживання в умовах воєнного часу, забезпечити процес чіткого інформування про обстановку та правила дій і поведінки населення в надзвичайних ситуаціях воєнного часу.

Для забезпечення стійкості роботи об'єктів повинні проводитись інженерно-технічні заходи на мережах комунального господарства з метою захисту джерел тепла із заглибленням у ґрунт комунікацій. Котельні слід розміщувати в спеціальному окремо розміщеному приміщенні.

Якщо об'єкт одержує тепло з міської теплоцентралі, необхідно провести заходи для забезпечення стійкості трубопроводів і розподільних пристроїв, підведених до об'єкта.

Теплова мережа має будуватися за кільцевою системою з прокладанням труб у спеціальних каналах зі з'єднанням паралельних ділянок. Для відключення пошкоджених ділянок мають бути встановлені запірно-регулюючі засувки,

вентилі та ін. Ці пристосування необхідно розміщувати в оглядових колодязях, на території, що не завалюється при руйнуванні будівель.

Система каналізації має будуватись окремо: одна для дощових, друга для промислових і господарських вод. На об'єкті має бути не менше двох виводів з підключенням до міських каналізаційних колекторів, а також виводи і колодязі з аварійними засувками на об'єктових колекторах з інтервалом 50 м на території, що не завалюється, для аварійного скидання неочищеної води в найближчі штучні та природні заглиблення.

На деяких промислових об'єктах є системи для забезпечення технології виробництва: для подання кисню, аміаку, стиснутого повітря та інших рідких і газових реактивів. Для цих систем розробляють заходи для попередження виникнення вторинних факторів зброї, стихійних лих та виробничих аварій і катастроф.

Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні.

З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги. Повітряні лінії електропостачання замінити на підземно-кабельні.

Створення необхідних запасів (резервів) паливно-мастильних матеріалів та інших видів палива й організація їх безпечного зберігання.

Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ.

Для підвищення стійкості забезпечення водою слід провести такі заходи. Необхідно створити основні і резервні джерела водопостачання. Як резервне джерело краще мати артезіанську свердловину, яку необхідно підключити до системи водопостачання. Крім того, воду можна брати з близько розміщеної

природної водойми або спорудити штучну водойму чи резервуари з обладнанням пристроїв для збору і перекачування води.

Всі ділянки водопостачання повинні бути заглиблені в ґрунт з обладнанням пожежних гідрантів і пристроїв для відключення пошкоджених ділянок. Локальні мережі водопостачання окремих великих підприємств варто з'єднати із загальноміською системою водопостачання в єдине кільце.

Підвищенню стійкості забезпечення водою сприяє подавання води безпосередньо в мережу поза водонапірними баштами, спорудження обвідних ліній для подання води поза пошкодженими спорудами.

Завчасне вжиття заходів захисту вододжерел, водопровідних споруд, свердловин і шахтних колодязів від забруднення радіоактивними речовинами, зараження хімічними і біологічними засобами.

Підготовка меліоративних, гідротехнічних та іригаційних споруд і систем до експлуатації в надзвичайних умовах.

Для забезпечення виробництва продукції необхідні електроенергія, паливо, мастила, засоби захисту рослин, міндобрива, профілактичні й лікувальні препарати ветеринарної медицини, запасні частини, сировина та інші матеріально-технічні засоби. Забезпечення об'єктів цими ресурсами дасть можливість випускати необхідну продукцію в надзвичайних умовах мирного і воєнного часу. Тому повинні проводитись такі заходи, які б забезпечили стійкість постачання і сприяли підвищенню захисту мережі електро-, водо-, газопостачання, транспортних комунікацій і джерел постачання всім необхідним для забезпечення функціонування галузей сільського господарства в надзвичайних умовах.

З метою попередження аварій на електричних мережах необхідно встановити автоматичну систему відключення перенапруги. Повітряні лінії електропостачання слід замінити на підземно-кабельні.

Газ використовується як паливо і на хімічних підприємствах у технологічному процесі. Для безперебійного забезпечення газом, газові мережі необхідно підводити до об'єкта з двох напрямків, які мають бути з'єднані в єдине

кільце з обладнанням для можливого дистанційного автоматичного управління й у разі необхідності відключення пошкоджених ділянок.

На великих підприємствах необхідно мати підземні ємності із закачаним резервним газом.

На підприємствах, де використовується пара, необхідно захистити джерела його постачання, заглибити в ґрунт комунікації паропостачання і встановити запірні пристосування.

Запас резервних матеріалів необхідно розраховувати на такі строки роботи підприємства, за які можливе відновлення регулярного постачання.

Передбачити, на випадок перебоїв в постачанні підприємствамисуміжниками, створення місцевих матеріалів, сировини для виготовлення комплектуючих виробів і інструментів силами свого підприємства.

Для підвищення стійкості та забезпечення збереження (відновлення) будівель і споруд в умовах воєнного часу необхідно:

- провести оцінку можливих ступенів руйнування будівель і споруд господарства населеного пункту, визначити обсяг невідкладних ремонтних робіт, потреби в будівельних матеріалах.
- створити і підготувати спеціальні формування для ремонтновідновних, будівельних та інших робіт на об'єкті.
- розробити комплекс протипожежних заходів, які виключали б можливість виникнення масових пожеж.

Для забезпечення надійності системи управління і зв'язку потрібно організувати захищений пункт управління, забезпечити його засобами зв'язку, які б дали можливість швидко доводити сигнали ЦЗ до всіх виробничих підрозділів і населення у місцях проживання. При цьому необхідно здійснити планування збору даних про обстановку, передачу команд і розпоряджень в умовах впливу на об'єкт уражаючих факторів. Для підвищення стійкості системи управління і зв’язку в умовах воєнного часу необхідно організувати використання радіозасобів, засобів телефонного зв'язку, а також забезпечити зв’язок із колонами евакуйованого населення, що перебувають у дорозі, і

відповідальними особами, які супроводжують їх під час евакуації, забезпечити дублювання ліній і каналів зв'язку.

Отже, підвищення стійкості роботи об'єктів господарської діяльності у воєнний час можна забезпечити шляхом організації і проведення сукупності заходів, які включають організаційні, інженерно-технічні й технологічні заходи. Надзвичайно важливим є планування та організація захисту комплексів критичної інфраструктури, зокрема водопостачання, водовідведення, газопостачання та зв’язку.

ВИСНОВКИ

Щоб досягти успіху в інноваціях персоналізації, недостатньо бути методичним у наших дослідженнях; простір для дослідження практично безмежний. Ля кожної стрімінгової системи важливо зосередити дослідження на плідних напрямках для розвитку; недостатньо використані джерела даних, краще представлення функцій, більш відповідні моделі та метрики, а також втрачені можливості для персоналізації. Використовуються інтелектуальний аналіз даних та інші експериментальні підходи, щоб поступово інформувати нашу інтуїцію, а отже, пріоритезувати інвестування зусиль.

У більшості попередніх контекстів – будь то рядок Топ-10, жанри чи подібні – ранжування, вибір порядку розміщення елементів у рядку, є критично важливим для забезпечення ефективного персоналізованого досвіду. Мета даної системи ранжування – знайти найкращий можливий порядок набору елементів для учасника, в певному контексті, в режимі реального часу. Ми розкладаємо ранжування на оцінювання, сортування та фільтрацію наборів фільмів для показу учаснику. Бізнес-мета – максимізувати задоволеність учасників та щомісячне утримання підписки, що добре корелює з максимізацією споживання відеоконтенту. Тому ми оптимізуємо наші алгоритми, щоб давати найвищі оцінки тим програмам і фільмам, які учасник, найімовірніше, буде грати чи дивитись та які йому сподобаються.

Також потрібно враховувати такі фактори, як контекст, популярність фільму, інтерес, новизна, різноманітність та свіжість. Підтримка всіх різних контекстів, у яких ми хочемо надавати рекомендації, вимагає низки алгоритмів, налаштованих на потреби цих контекстів. У наступній частині цієї роботи детальніше розглянемо проблему ранжування. Також заглибимося в аналіз даних та моделі, які роблять все вищезазначене можливим, та обговоримо підхід до інновацій у цій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bodnarchuk, I., Skorenkyy, Y., Kramar, T., Duda, O., & Nykytyuk, V. (2022, November). Use of Analytical Hierarchy Process in Scenarios Design for a Digital Museum with XR components. In ITTAP (pp. 414-425).

2. Ihor, B., Oleksii, D., Aleksandr, K., Nataliia, K., Oleksandr, M., & Volodymyr, P. (2019, January). Multicriteria choice of software architecture using dynamic correction of quality attributes. In International Conference on Computer Science, Engineering and Education Applications (pp. 419-427). Cham: Springer International Publishing.

3. Lutskiv A. Adaptable Text Corpus Development for Specific Linguistic Research / Andriy Lutskiv, Nataliya Popovych // International Scientific-Practical Conference «Problems of Infocommunications. Science and Technology» (October 8-11, 2019), 2019. - C.217-223.

4. Lutskiv A. Big Data Approach to Developing Adaptable Corpus Tools /Andriy Lutskiv, Nataliya Popovych// Computational Linguistics and Intelligent Systems. Proc. 4thInt. Conf. COLINS 2020. Volume I:Workshop. Lviv, Ukraine, April23-24, 2020, CEUR-WS.org, online. pp.374-395. [Electronic resource] Access mode: URL: http://ceur-ws.org/Vol-2604/

5. Lutskiv A. Corpus-Based Translation Automation in Adaptable Corpus Translation Module /Andriy Lutskiv, Roman Lutsyshyn// Computational Linguistics and Intelligent Systems. Proc. 5th Int. Conf. COLINS 2021. Volume I: Workshop. Lviv, Ukraine, April 22-23, 2021, CEUR-WS.org, online. pp.374-395. [Electronic resource] Access mode: URL: http://ceur-ws.org/Vol-2604/R. Malhotra, “A systematic review of machine learning techniques for software fault prediction,” Appl. Soft Comput.vol. 27. Feb. 2015. pp. 504–518.

6. L. Son et al., “Empirical Study of Software Defect Prediction: A Systematic Mapping,” Symmetry (Basel)., vol. 11, no. 2. 2019. p. 212.

7. Голінько В. І. Охорона праці в галузі інформаційних технологій: навч. посіб. / В. І. Голінько, М. Ю. Іконніков, Я. Я. Лебедєв; М-во освіти і науки

України, Держ. вищий навч. закл. "Нац. гірн. ун-т". - Дніпропетровськ: НГУ, 2015. - 246 с.

8. Гандзюк М.П. Основи охорони праці: Підручник. 4-е вид./Гандзюк М.П., Желібо Є.П., Халімовський М.О. - Київ: Каревела, 2008. – 384с.

9. Техноекологія та цивільна безпека. Частина «Цивільна безпека»: Навчальний посібник; укл.: Стручок В. С. Тернопіль: ФОП Паляниця В.А., 2022. 150 с.

10. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання / укл.: Стручок В. С. Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

11. Умови праці працівників, які використовують у роботі персональні комп'ютери. Zolochiv.Net. URL: https://zolochiv.net/umovy-pratsi-pratsivnykiv-iaki-vykorystovuiut-u-roboti-personal-ni-komp-iutery/ (дата звернення: 25.10.2024).

ДОДАТКИ

Додаток А

Тези доповіді

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

58.	**М.І. Паламар, Ю.А. Удич, А.М. Паламар, М.Р. Франків** ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА МОНІТОРИНГУ ПАРАМЕТРІВ РОЗУМНОГО БУДИНКУ НА ОСНОВІ ІОТ ТЕХНОЛОГІЙ		320
59.	**Ю.Б. Паляниця, М.Ю. Ковальчук** МЕТОД ІНТЕЛЕКТУАЛЬНОЇ КЛАСИФІКАЦІЇ ЛІТАЮЧИХ ОБ'ЄКТІВ ЗА ЇХ АКУСТИЧНИМИ СИГНАТУРАМИ		321
60.	**С.М. Панасенко, Н.С. Луцик** ІНТЕГРАЦІЯ RULE-BASED АЛГОРИТМІВ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ТЕСТУВАННЯ ЕЛЕКТРОНІКИ		324
61.	**А. Пенцак, Ю. Лещишин** ВБУДОВАНА СИСТЕМА ДЛЯ КОМП'ЮТЕРНОГО АНАЛІЗУ ДИХАЛЬНИХ ЗВУКІВ		325
62.	**В.С. Пиж** ІННОВАЦІЙНИЙ ПІДХІД ДО ПОБУДОВИ ЗАХИЩЕНОГО ВІРТУАЛЬНОГО СЕРЕДОВИЩА ДЛЯ ОСВІТНЬОГО ПРОЦЕСУ		326
63.	**О.Б. Пйонтковський** ЕНЕРГОЕФЕКТИВНІ МІКРОКОНТРОЛЕРИ ДЛЯ АВТОНОМНИХ ІОТ–ПРИСТРОЇВ		328
64.	**І. І. Поліщук, Н.С. Луцик** АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ ПЕРЕЛОМІВ НА МЕДИЧНИХ ЗОБРАЖЕННЯХ		329
65.	**І.Р. Ралік** РЕЗОНАНСНИЙ СЕЛЕКТОР СТРАТЕГІЙ У ІНТЕЛЕКТУАЛЬНИХ CRM-СИСТЕМАХ		331
66.	**І.І. Рій, Є.В. Тиш** ПОРІВНЯЛЬНИЙ АНАЛІЗ ХАОТИЧНИХ І КЛАСИЧНИХ МЕТОДІВ ШИФРУВАННЯ З ТОЧКИ ЗОРУ ІНФОРМАЦІЙНОЇ ЕНТРОПІЇ		332
67.	**Рожик А. М.** МЕТОДИ ТА ПРОГРАМНО-АПАРАТНІ ЗАСОБИ ІДЕНТИФІКАЦІЇ ПРАЦІВНИКІВ З МЕТОЮ ВИЗНАЧЕННЯ РОБОЧОГО ЧАСУ ТА ДОСТУПУ ДО ПРИМІЩЕННЯ		333
68.	**В. М. Руснак, Н. Б. Стадник** МЕТОДИ АДАПТИВНОГО ВИБОРУ ЕВРИСТИК ДЛЯ ЗАДАЧ ГІЛЬЙОТИННОГО РОЗКРОЮ		335
69.	**П.В. Свінціцький, Н.М. Пановик, О.В. Доберчак, І.О. Боднарчук** СУЧАСНА АРХІТЕКТУРА СТРІМІНГОВИХ ПЛАТФОРМ В РЕАЛЬНОМУ ЧАСІ		336
70.	**П.М. Семчишин** АРХІТЕКТУРНІ РІШЕННЯ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ		340
71.	**С.О.Синишен** МІКРО- ТА НАНОРОБОТИ: ТЕХНІЧНІ ВІДМІННОСТІ, ПЕРЕВАГИ ТА НЕДОЛІКИ MPI І GDI		342
72.	**Я.Р. Сиротинський; А.М. Паламар, к.т.н., доц.; А.П. Шмігель** КОМП'ЮТЕРИЗОВАНА СИСТЕМА ВИМІРЮВАННЯ ШВИДКОСТІ ВІТРУ НА ОСНОВІ ІОТ ТЕХНОЛОГІЙ		343
73.	**М. Смоляк** СИСТЕМА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ В АВТОМОБІЛЯХ		344
74.	**А.А. Станько, С.М. Куриляк, Я.О. Тригуб** АВТОМАТИЗОВАНА СИСТЕМА ОЦІНЮВАННЯ ТЕРМІНУ ЗБЕРІГАННЯ ХАРЧОВИХ ПРОДУКТІВ НА ОСНОВІ БАГАТОКАНАЛЬНИХ ГАЗОВИХ СЕНСОРІВ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ		347

2. Чуприна О.С. Гібридні алгоритми розв'язання задач двовимірного пакування з урахуванням технологічних обмежень. *Системні технології*. 2019. № 2 (121). С. 45–53.

УДК 004.738.5:004.451:621.397.13

П.В. Свінціцький, Н.М. Пановик, О.В. Доберчак, І.О. Боднарчук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

СУЧАСНА АРХІТЕКТУРА СТРІМІНГОВИХ ПЛАТФОРМ В РЕАЛЬНОМУ ЧАСІ

P.V. Svintsitsky, N.M. Panovyk, O.V. Doberchak, I.O. Bodnarchuk

MODERN ARCHITECTURE OF REAL-TIME STREAMING PLATFORMS

Потокове передавання даних у реальному часі передбачає безперервне введення, обробку та виведення даних, що відбувається майже миттєво. Воно дозволяє отримувати, аналізувати та реагувати на дані в міру їх створення, надаючи цінну аналітику без затримки. Ця технологія має вирішальне значення в умовах, коли своєчасна інформація є критично важливою, що дозволяє швидко приймати рішення та проводити аналітику в реальному часі.

Основною характеристикою систем потокового передавання даних у реальному часі є їхня здатність обробляти величезний обсяг даних з низькою затримкою [1, 3]. Це скорочує час між генерацією даних та отриманням корисної аналітики, що робить їх безцінними для різних галузей, таких як фінанси, охорона здоров'я та технології. Незалежно від того, чи йдеться про потокове передавання даних фондового ринку, чи про моніторинг пристроїв Інтернету речей, потокове передавання даних у реальному часі забезпечує ефективність та швидку реакцію на обробку даних.

Наведемо декілька загальних прикладів використання стрімінгу в реальному часі.

1. Моніторинг соціальних мереж.

Платформи соціальних мереж постійно генерують величезні обсяги даних. Потокове передавання даних у режимі реального часу дозволяє компаніям моніторити канали соціальних мереж на предмет згадок брендів, настроїв клієнтів та нових тенденцій. Це допомагає організаціям проактивно взаємодіяти з клієнтами, керувати своєю онлайн-репутацією та коригувати маркетингові стратегії на ходу.

2. Обробка фінансових даних.

У фінансових послугах потокове передавання даних у режимі реального часу використовується для аналізу ринку, торгових стратегій та управління ризиками. Фондові біржі та трейдери покладаються на дані в режимі реального часу, щоб приймати обґрунтовані рішення та здійснювати угоди. Ця можливість дозволяє фінансовим установам миттєво реагувати на коливання ринку, максимізуючи можливості отримання прибутку та мінімізуючи збитки.

3. Виявлення шахрайства.

Системи виявлення шахрайства використовують потокове передавання даних у режимі реального часу для виявлення підозрілої діяльності в міру її виникнення. Аналізуючи моделі транзакцій та поведінку користувачів, ці системи можуть виявляти аномалії, що свідчать про шахрайські дії. Такий моніторинг у режимі реального часу допомагає запобігти шахрайству, дозволяючи вживати негайних заходів, таких як блокування транзакцій або сповіщення користувачів.

4. Прогнозне обслуговування.

Прогностичне обслуговування використовує потокове передавання даних у режимі реального часу для моніторингу продуктивності обладнання та прогнозування збоїв до їх виникнення. Датчики на обладнанні безперервно надсилають дані про такі параметри, як температура, тиск і вібрація. Ці дані обробляються в режимі реального часу для виявлення аномалій та прогнозування потенційних поломок, що дозволяє своєчасно втручатися в технічне обслуговування.

Далі опишемо основні компоненти архітектури потокової передачі даних у режимі реального часу.

1. Джерело даних.

Компонент джерела даних – це місце, звідки дані походять. Це можуть бути різні системи, програми або пристрої, включаючи датчики, журнали, бази даних та платформи соціальних мереж. Ці джерела безперервно генерують необроблені дані, передаючи їх у конвеєр потокової передачі даних. Ефективне керування кількома джерелами даних має вирішальне значення для безперебійного потоку даних. Інтеграція різноманітних джерел даних часто вимагає конекторів або API для забезпечення сумісності та передачі даних.

2. Забір потоку.

Забір потоку – це процес захоплення та імпорту потоків даних на платформу потокової передачі даних. Для виконання цього завдання зазвичай використовуються такі технології, як Apache Kafka, Amazon Kinesis та Azure Event Hubs. Вони можуть забирати величезні обсяги даних у режимі реального часу, гарантуючи, що жодні дані не будуть втрачені під час передачі. Ефективний забір потоку вимагає низької затримки та високої пропускної здатності для підтримки цілісності потоку даних. Цей етап також включає завдання попередньої обробки (фільтрація, перетворення, тощо.

3. Потокове сховище.

Після отримання дані необхідно зберігати для подальшого аналізу. Рішення для потокового сховища, такі як внутрішнє сховище Apache Kafka, AWS S3 та Azure Blob Storage, підтримують тимчасове або довгострокове зберігання потоків даних. Ці системи обробляють великі обсяги даних, забезпечуючи швидкий доступ та пошук.

4. Потокова обробка.

Потокова обробка включає аналіз потоків даних у режимі реального часу для отримання корисної аналітики. Такі фреймворки, як Apache Flink, Apache Storm та Spark Streaming, використовуються для обробки великомасштабних потоків даних з мінімальною затримкою. Ці інструменти дозволяють виконувати складну обробку подій, агрегацію, об'єднання та операції вікон.

5. Пункт призначення.

Останнім компонентом є пункт призначення, куди доставляються оброблені дані. Це може бути сховище даних, база даних, озеро даних або програми кінцевих користувачів. Системи призначення зберігають кінцеві оброблені дані або запускають дії, такі як системи сповіщень, панелі моніторингу або автоматизовані робочі процеси.

Найбільш загальна схема стрімінгового сервісу показана на рисунку 1.

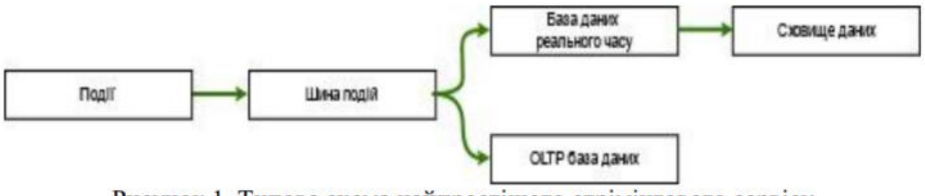


Рисунок 1. Типова схема найпростішого стрімінгового сервісу

Ця схема відображає базову архітектуру стрімінгового сервісу реального часу та ілюструє шлях обробки даних від джерела до споживача. В даному випадку споживачами є сховище даних і/або OLTP-база даних, хоча перелік таких споживачів може бути набагато ширший. Наприклад, це може бути платформа демонстрації медіа-контенту з можливостями ведення прямих ефірів, чат-бот [6], система розпізнавання образів [4] або система контролю за дорожним рухом [2].

Процес починається з події, яка надходить до шини подій – центрального компонента архітектури, що виконує роль посередника для розподілу потоків даних. Від шини подій дані розгалужуються на три напрямки: перший веде до бази даних реального часу з подальшим зберіганням у сховищі даних, другий спрямовує інформацію до OLTP бази даних для обробки транзакційних операцій, а третій забезпечує альтернативний шлях обробки.

Така архітектура забезпечує паралельну обробку подій у реальному часі, розділяючи функції оперативного зберігання, роботи з актуальними даними та довготривалого архівування, що є типовим підходом для стрімінгових платформ.

Перерахуємо кілька способів ефективного впровадження потокової передачі даних у реальному.

1. Розділення даних на розділи для паралельної обробки

Паралельна обробка має вирішальне значення для масштабування рішень потокової передачі даних у реальному часі. Розділення даних на розділи дозволяє кільком процесам одночасно обробляти різні сегменти даних, підвищуючи пропускну здатність та зменшуючи затримку. Такі технології, як Apache Kafka, використовують розділи для ефективного керування вхідними потоками даних.

2. Додавання більшої кількості вузлів до системи для обробки збільшеного навантаження.

Масштабованість є важливою для потокової передачі даних у реальному часі, і додавання більшої кількості вузлів до системи є поширеною практикою для врахування збільшених навантажень. Кожен додатковий вузол може обробляти підмножину потоку даних, розподіляючи робоче навантаження на обробку та підвищуючи загальну ємність системи. Динамічна масштабованість дозволяє системі реагувати на різні навантаження даних без шкоди для продуктивності. Впровадження масштабованої архітектури гарантує, що потокове передавання даних у реальному часі залишається ефективним та швидким у сценаріях високого навантаження, сприяючи безперебійній обробці та аналізу даних.

3. Оптимізація мережевих протоколів та логіки обробки для зменшення затримки.

Мінімізація затримки є критично важливою для потокового передавання даних у реальному часі. Оптимізація мережевих протоколів та логіки обробки може значно зменшити час затримки. Такі методи, як мінімізація накладних витрат на серіалізацію/десеріалізацію даних та використання ефективних протоколів передачі даних, сприяють зниженню затримки. Ефективна логіка обробки включає оптимізацію алгоритмів та використання обчислень у пам'яті для пришвидшення обробки даних. Ці оптимізації покращують можливості системи в реальному часі, забезпечуючи своєчасний та точний аналіз даних та прийняття рішень.

4. Налаштування розподілу ресурсів для оптимізації продуктивності.

Динамічний розподіл ресурсів гарантує, що ресурси системи відповідають поточному навантаженню даних, оптимізуючи продуктивність та економічну ефективність. Такі методи, як автоматичне масштабування, дозволяють системі автоматично регулювати обчислювальну потужність на основі попиту в реальному часі, забезпечуючи оптимальну продуктивність. Стратегії розподілу ресурсів включають

моніторинг системних показників та налаштування ресурсів процесора, пам'яті та сховища для задоволення вимог обробки. Ця адаптивність допомагає підтримувати стабільність та продуктивність системи під час пікових навантажень, підвищуючи надійність та ефективність потокової передачі даних у реальному часі.

5. Зберігання інформації про стан для обробки складних подій.

Обробка з урахуванням стану є важливою для обробки складних подій у потоках даних у реальному часі. Зберігання інформації про стан дозволяє системі відстежувати поточні події, керувати даними сеансів та співвідносити події з часом. Такі фреймворки, як Apache Flink, надають надійні можливості управління станом для підтримки цих завдань. Ефективне управління станом забезпечує точну та своєчасну аналітику, дозволяючи системі обробляти складну логіку обробки подій. Завдяки обробці з урахуванням стану, рішення для потокової передачі даних у реальному часі можуть впроваджувати розширену аналітику, забезпечуючи прийняття обґрунтованих рішень на основі безперервних потоків даних.

6. Налаштування сповіщень про незвичайну поведінку або погіршення продуктивності.

Налаштування сповіщень про незвичайну поведінку або погіршення продуктивності має вирішальне значення для підтримки справності та надійності систем потокової передачі даних у реальному часі. Моніторинг ключових показників продуктивності, таких як затримка обробки, коефіцієнти помилок та завантаження системи, дозволяє швидко виявляти проблеми. Автоматизоване сповіщення дозволяє негайно реагувати на аномалії, зменшуючи час простою та зменшуючи ризики. Та підвищуючи надійність системи.

Здатність обробляти та аналізувати дані в режимі реального часу стала критично важливою для бізнесу, дозволяючи їм отримувати, обробляти та реагувати на дані безперервно та масштабовано. Типово системи опрацювання даних в реальному часі базуються на технологіях Apache Sparks та релевантних сервісах. В рамках проведення подальших досліджень буде деталізована архітектура рішень для різних предметних областей та з використанням відповідних архітектурних програмних рішень.

Література

1. Ланевич Т. Apache Kafka та Rabbitmq: порівняння архітектур та можливостей. Тези XIII МНПК „*Актуальні задачі сучасних технологій*“, 11-12 грудня 2024 року. Тернопіль. : ФОП Паляниця В. А., 2024. С. 402–403.
2. Мадяк О. П. Використання даних про дорожній рух у реальному часі для керування дорожньо-транспортною системою. Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 16-17 листопада 2017 року. – Т. : ТНТУ, 2017. – Том 2. – С. 108.
3. Остапчук О. Цуприк Г. Технічні особливості взаємодії між клієнтом та сервером у реальному часі. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології“, 7–8 грудня 2022 року. – Т. : ТНТУ, 2022. С. 124. – (Програмна інженерія та моделювання складних розподілених систем).
4. Панчишин П. С. Підвищення якості розпізнавання об’єктів у відеопотоці в реальному часі / П. С. Панчишин, Михайло Іванович Паламар // Матеріали XII НТК „ІМСТ“, 18-19 грудня 2024 року. – Т. : ТНТУ, 2024. – С. 186–187.
5. Федорович І. Використання Spark Streaming та Spark Structured Streaming для обробки даних в реальному часі / Ілля Федорович, Галина Осухівська // ІМСТТ, 13-14 грудня 2023 року. Тернопіль : ТНТУ, 2023. С. 225.
6. Хом’як А. С. Підвищення ефективності обробки в реальному часі у службах чат-ботів через інтеграцію черги запитів для розподілення навантаження / А.

Додаток Б

Програмний код Java фрагментарно

NodeModelChatClientRouter.java

```
package github.ai.qa.solutions.services;

import github.ai.qa.solutions.configuration.AiClientsConfiguration;
import java.util.Map;
import java.util.Set;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.ai.chat.client.ChatClient;
import org.springframework.beans.factory.ObjectProvider;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

/**
 * Routes a node/tool name to a {@link ChatClient} family (GigaChat or
OpenRouter).
 *
 * <p>Resolution order:</p>
 * - explicit model in properties → family inferred from model string
 * - default lists per role → family by curated defaults
 * - heuristic by name → OpenRouter for validate/think/reason, otherwise
GigaChat
 *
 * <p>For testability, the pure decision is exposed by {@link
#decideFamily(String, Map)}.</p>
 */
@Service
public class NodeModelChatClientRouter implements ChatClientRouter {
    /** Logs routing decisions. */
    private static final Logger log =
LoggerFactory.getLogger(NodeModelChatClientRouter.class);
    /** Provider for GigaChat-backed {@link ChatClient} (generative/produce
JSON). */
    private final ObjectProvider<ChatClient> gigaChatClient;
    /** Provider for OpenRouter-backed {@link ChatClient}
(validate/think/reason). */
    private final ObjectProvider<ChatClient> openRouterClient;
    /** Node/tool → model mapping from external configuration. */
    private final AiClientsConfiguration.NodeModelRoutingProperties props;
```

```
    /** Node/tool simple names routed to OpenRouter by default. */
    static final Set<String> DEFAULT_OPENROUTER = Set.of(
            // Nodes
            "ValidateJsonSchemaNode",
            "VerifyJsonByJsonSchemaNode",
            "ReasonAndRouteNode",
            // Tools
            "ThinkHowToGenerateTool",
            "ThinkHowToFixJsonTool");

    /** Node/tool simple names routed to GigaChat by default. */
    static final Set<String> DEFAULT_GIGACHAT = Set.of(
            // Nodes
            "GenerateJsonNode", "FixErrorsInJsonNode",
            // Tools
            "GenerateJsonBySchemaTool", "FixValidationErrorsInJsonTool");

    /**
     * Creates a router with two families and external routing properties.
     *
     * @param gigaChatClient   provider for GigaChat chat client
     * @param openRouterClient provider for OpenRouter chat client
     * @param props            configured node→model overrides
     */
    public NodeModelChatClientRouter(
            @Qualifier("generativeChatClient") final ObjectProvider<ChatClient>
gigaChatClient,
            @Qualifier("thinkingChatClient") final ObjectProvider<ChatClient>
openRouterClient,
            final AiClientsConfiguration.NodeModelRoutingProperties props) {
        this.gigaChatClient = gigaChatClient;
        this.openRouterClient = openRouterClient;
        this.props = props;
    }

    /** Minimal holder for a routing decision. */
    static final class Decision {
        /** Selected family: "GigaChat" or "OpenRouter". */
        final String family;
        /** Model label: explicit model or one of {@code <default>} / {@code
<heuristic>}. */
```

```
        final String modelLabel;

        Decision(final String family, final String modelLabel) {
            this.family = family;
            this.modelLabel = modelLabel;
        }
    }

    /**
     * Pure decision logic that chooses target family and model label.
     *
     * @param nodeOrToolSimpleName simple class name of node/tool
     * @param configured optional overrides mapping simple name to model string
     * @return routing decision including family and label
     */
    static Decision decideFamily(final String nodeOrToolSimpleName, final
Map<String, String> configured) {
        final String model = configured == null ? null :
configured.get(nodeOrToolSimpleName);
        if (model != null && !model.isBlank()) {
            if (isGigaChatModel(model)) return new Decision("GigaChat", model);
            if (isOpenRouterModel(model)) return new Decision("OpenRouter",
model);
        }
        if (DEFAULT_GIGACHAT.contains(nodeOrToolSimpleName)) return new
Decision("GigaChat", "<default>");
        if (DEFAULT_OPENROUTER.contains(nodeOrToolSimpleName)) return new
Decision("OpenRouter", "<default>");
        final String nodeLow = nodeOrToolSimpleName.toLowerCase();
        if (nodeLow.contains("validate") || nodeLow.contains("think") ||
nodeLow.contains("reason")) {
            return new Decision("OpenRouter", "<heuristic>");
        }
        return new Decision("GigaChat", "<heuristic>");
    }

    /**
     * Resolves the {@link ChatClient} for the given node/tool simple name.
     *
     * @param nodeOrToolSimpleName simple class name of node/tool
     * @return a non-null {@link ChatClient}
     * @throws IllegalStateException when neither family is available
```

```
     */
    @Override
    public ChatClient forNode(final String nodeOrToolSimpleName) {
        final Decision d = decideFamily(nodeOrToolSimpleName, props.nodes());
        if ("GigaChat".equals(d.family)) {
            return pickAndLog(nodeOrToolSimpleName, d.family, d.modelLabel,
gigaChatClient, openRouterClient);
        }
        return pickAndLog(nodeOrToolSimpleName, d.family, d.modelLabel,
openRouterClient, gigaChatClient);
    }

    /**
     * Heuristic: whether model string denotes a GigaChat model.
     *
     * @param model model identifier string
     * @return true when likely a GigaChat model
     */
    private static boolean isGigaChatModel(final String model) {
        final String m = model.toLowerCase();
        return m.contains("gigachat") || m.startsWith("giga");
    }

    /**
     * Heuristic: whether model string denotes an OpenRouter model.
     *
     * @param model model identifier string
     * @return true when likely an OpenRouter model
     */
    private static boolean isOpenRouterModel(final String model) {
        // OpenRouter models typically include provider prefix like
provider/model
        final String m = model.toLowerCase();
        return m.contains("/") || m.contains(":");
    }

    private ChatClient pickAndLog(
            final String nodeName,
            final String preferredFamily,
            final String modelLabel,
            final ObjectProvider<ChatClient> preferred,
            final ObjectProvider<ChatClient> fallback) {
```

```
        ChatClient chatClient = preferred.getIfAvailable();
        String family;
        if (chatClient != null) {
            family = preferredFamily;
        } else {
            chatClient = fallback.getIfAvailable();
            family = preferredFamily.equals("GigaChat") ? "OpenRouter" :
"GigaChat";
        }
        if (chatClient == null) {
            throw new IllegalStateException(
                    "No ChatClient beans available for routing. Ensure profiles
are configured.");
        }
        log.info(" Route [{}] → family={} model={}", nodeName, family,
modelLabel);
        return chatClient;
    }
}
```

AgentState.java

```
package github.ai.qa.solutions.state;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;
import org.bsc.langgraph4j.state.Channel;
import org.bsc.langgraph4j.state.Channels;

public class AgentState extends org.bsc.langgraph4j.state.AgentState {

    /**
     * State keys carried through the graph.
     */
    public enum StateKey {
        /** User prompt driving generation. */
        USER_PROMPT,
        /** Latest generated JSON. */
        GENERATED_JSON,
        /** Validator display text (OK or newline-joined errors). */
        VALIDATION_RESULT,
```

```
        /** Sorted signature of validation errors. */
        VALIDATION_SIGNATURE,
        /** JSON Schema provided by the user. */
        JSON_SCHEMA,
        /** Detected JSON Schema version label. */
        SCHEMA_VERSION,
        /** Heuristic warnings gathered during normalization. */
        HEURISTIC_SIGNATURE,
        /** Plan for generation. */
        PLAN_GENERATION,
        /** Plan for fixing invalid JSON. */
        PLAN_FIX,
        /** Routing decision. */
        DECISION,
        /** Reasoning behind the decision. */
        REASONING,
        /** Iteration counter for repeated FIX attempts. */
        ITERATION,
        /** Previous validator display text. */
        PREV_VALIDATION_RESULT,
        /** Previous validation signature. */
        PREV_VALIDATION_SIGNATURE
    }

    /** Unmodifiable schema mapping state keys to channels. */
    public static final Map<String, Channel<?>> SCHEMA = initSchema();

    private static Map<String, Channel<?>> initSchema() {
        final Map<String, Channel<?>> schemaMap = new HashMap<>();
        final Channel<?> defaultChannel = Channels.base((current, update) ->
update);
        for (StateKey key : StateKey.values()) {
            schemaMap.put(key.name(), defaultChannel);
        }
        return java.util.Collections.unmodifiableMap(schemaMap);
    }

    public AgentState(Map<String, Object> initData) {
        super(initData);
    }

    public String get(final StateKey key) {
```

```
        return this.<String>value(key.name())
                .orElseThrow(() -> new IllegalStateException("Key did not found
into AgentState: " + key));
    }

    public Optional<String> getOptional(final StateKey key) {
        return this.value(key.name());
    }
}
```

AgentApplication.java

```
package github.ai.qa.solutions;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AgentApplication {

    private AgentApplication() {}

    public static void main(String[] args) {
        SpringApplication.run(AgentApplication.class, args);
    }
}
```