ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі із застосуванням машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ЩЕРБАЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62
Олександлр ЩЕРБАЧЕНКО

Керівник: В'ячеслав ТРЕЙТЯК
канд. техн. наук

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Щербаченку Олександру Сергійовичу

1. Тема кваліфікаційної роботи: «Методика автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі із застосуванням машинного навчання»

керівник кваліфікаційної роботи В'ячеслав ТРЕЙТЯК, канд. техн. наук,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з машинного навчання та обробки природної мови, статистичні дані сервіс-деск систем, методології ITIL та ITSM, документація фреймворків машинного навчання, історичні дані інцидентів для навчання моделей.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Проаналізувати сучасні підходи до автоматизації процесів класифікації та пріоритизації інцидентів у сервіс-деск системах.
2. Дослідити методи машинного навчання та обробки природної мови для аналізу текстових описів інцидентів.
3. Розробити гібридну методику автоматизованої обробки інцидентів, що поєднує ML-класифікатори, контекстні ембеддинги та мовні моделі.

4. Спроектувати та реалізувати програмний прототип сервіс-деск системи з інтегрованими модулями автоматизованої обробки.
5. Провести експериментальне дослідження ефективності розробленої методики.

5. Перелік ілюстративного матеріалу: *презентація*

1. Етапи автоматизованої обробки інцидентів у сервіс-деск системі.
2. Потік взаємодії користувача ч.1.
3. Потік взаємодії користувача ч.2.
4. Математична модель ансамблю ML + LLM
5. Стратегії прийняття рішення:
6. Математична модель Smart Assignment: 5-факторний скоринг виконавців
7. Порівняння 4-рівневої методики з існуючими підходами
8. Методика 4-рівневої адаптивної системи зважування для навчання
9. Порівняння функціональності прототипу із типовими підходами
10. LLM-маршрутизація тікетів та структура промту:
11. Схема даних: тікети, користувачі та ML-логи
12. Робота прототипу

6. Дата видачі завдання«31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10-04.11.2025	
2	Дослідження існуючих підходів до автоматизації сервіс-деск	05.11-07.11.2025	
3	Аналіз методів машинного навчання для класифікації текстів	07.11-10.11.2025	
4	Розробка гібридної методики HYBRID v2.0	11.11-18.11.2025	
5	Проектування архітектури програмного прототипу	19.11-24.11.2025	
6	Реалізація ML/LLM-пайплайну та механізму triage	25.11-1.12.2025	
7	Реалізація 4-рівневої системи активного навчання	02.12-04.12.2025	
8	Розробка програмного прототипу з використянням методики HYBRID v2.0.	05.12-15.12.2025	

10	Проведення експериментального тестування	16.12.2025	
11	Розробка демонстраційних матеріалів	17.12-18.12.2025	
12	Попередній захист роботи	19.12.2025	

Здобувач вищої освіти _____ Олександр ЩЕРБАЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ В'ячеслав ТРЕЙТЯК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 78 стор., 10 табл., 18рис., 55 джерел.

Мета роботи – удосконалення процесу автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі гібридної моделі з узгодженням рішень ML та LLM компонентів.

Об'єкт дослідження – процеси обробки інцидентів у сервіс-деск системі,

Предмет дослідження – моделі та методи автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі аналізу історичних тікетів і зворотного зв'язку.

У роботі проаналізовано існуючі підходи до автоматизації сервіс-деск процесів, зокрема правила та експертні системи, класичні ML-моделі для текстової класифікації, контекстні ембеддинги та великі мовні моделі. Визначено переваги та обмеження кожного підходу, обґрунтовано доцільність їх комбінування. Досліджено методи представлення тексту та їх вплив на якість класифікації інцидентів. Проаналізовано підходи human-in-the-loop та активного навчання для забезпечення контролю якості в умовах невизначеності.

Спроектовано архітектуру гібридної методики HYBRID v2.0, що поєднує ML-класифікатор на контекстних ембеддингах із локальною мовною моделлю. Розроблено модель ансамблевого узгодження рішень із пороговими значеннями впевненості та контрольованим механізмом triage для обробки невизначених випадків. Передбачено структурну валідацію відповідей LLM, що знижує ризик некоректних результатів та підвищує відтворюваність роботи системи. Запропоновано модель Smart Assignment із багатофакторним скорингом для автоматизованого призначення виконавців, що враховує спеціалізацію агента, належність до підрозділу, поточну доступність та історичну ефективність.

Реалізовано fallback-логіку для перенаправлення інцидентів на ручне призначення у нестандартних ситуаціях.

Реалізовано програмний прототип сервіс-деск системи з використанням сучасних технологій, що забезпечує повний цикл обробки інциденту: від реєстрації звернення до призначення виконавця. Розроблено модель даних для фіксації ML/LLM-прогнозів та протоколювання прийнятих рішень. Створено API для роботи з інцидентами та інтеграції модулів класифікації, пріоритизації та маршрутизації. Для адаптивного вдосконалення моделей впроваджено 4-рівневу схему активного навчання, яка дозволяє поступово розширювати навчальну вибірку з мінімальним обсягом ручної розмітки та забезпечує накопичення прикладів різної надійності.

Проведено експериментальне тестування на тестовому наборі інцидентів, яке підтвердило працездатність та ефективність запропонованих рішень. Виконано порівняння з базовим підходом на основі традиційного ML-класифікатора за метриками якості класифікації та пріоритизації. Досліджено часові характеристики системи та поведінку при тайм-ауті LLM-компонента.

Результати дослідження мають практичну цінність для організацій, що використовують сервіс-деск системи. Впровадження розробленої методики дозволить скоротити час обробки інцидентів, знизити навантаження на операторів першої лінії підтримки та підвищити якість маршрутизації звернень. Визначено перспективи подальшого розвитку, зокрема розширення набору реальних даних, формалізацію SLA-орієнтованих правил та інтеграцію з промисловими ITSM-платформами.

КЛЮЧОВІ СЛОВА: SERVICE DESK, КЛАСИФІКАЦІЯ ІНЦИДЕНТІВ, ПРІОРИТИЗАЦІЯ, МАШИННЕ НАВЧАННЯ, МОВНІ МОДЕЛІ, TRIAGE, SMART ASSIGNMENT, АКТИВНЕ НАВЧАННЯ, ГІБРИДНА МЕТОДИКА, КОНТЕКСТНІ ЕМБЕДДИНГИ.

ABSTRACT

Text part of the qualification work for obtaining a master's degree: 78 pages, 10 tables, 18 figures, 55 sources.

The purpose of the work is to improve the process of automated classification and prioritization of incidents in the service desk system based on a hybrid model with coordination of ML and LLM component decisions.

The object of the study is the processes of incident processing in the service desk system.

The subject of the study is models and methods of automated classification and prioritization of incidents in the service desk system based on the analysis of historical tickets and feedback.

The paper analyzes existing approaches to service desk process automation, including rules and expert systems, classical ML models for text classification, contextual embeddings, and large language models. The advantages and limitations of each approach are identified, and the feasibility of their combination is substantiated. Text representation methods and their impact on incident classification quality are investigated. Human-in-the-loop and active learning approaches for quality control under uncertainty are analyzed.

The architecture of the hybrid methodology HYBRID v2.0, which combines an ML classifier based on contextual embeddings with a local language model, has been designed. A model for ensemble decision coordination with confidence thresholds and a controlled triage mechanism for handling uncertain cases has been developed. Structural validation of LLM responses is provided, which reduces the risk of incorrect results and improves system reproducibility. A Smart Assignment model with multifactor scoring for automated agent assignment has been proposed, which takes into account agent specialization, department affiliation, current availability, and historical performance.

Fallback logic for redirecting incidents to manual assignment in non-standard situations has been implemented.

A software prototype of the service desk system has been developed using modern technologies, providing a complete incident processing cycle: from request registration to agent assignment. A data model for recording ML/LLM predictions and logging decisions has been developed. An API for working with incidents and integrating classification, prioritization, and routing modules has been created. For adaptive model improvement, a 4-level active learning scheme has been implemented, which allows gradual expansion of the training dataset with minimal manual labeling and ensures accumulation of samples with varying reliability levels.

Experimental testing on a test set of incidents has been conducted, which confirmed the operability and effectiveness of the proposed solutions. A comparison with the baseline approach based on a traditional ML classifier using classification and prioritization quality metrics has been performed.

The research results have practical value for organizations using service desk systems. Implementation of the developed methodology will reduce incident processing time, decrease the workload on first-line support operators, and improve the quality of request routing. Prospects for further development have been identified, including expansion of real data sets, formalization of SLA-oriented rules, and integration with industrial ITSM platforms.

KEYWORDS: SERVICE DESK, INCIDENT CLASSIFICATION, PRIORITIZATION, MACHINE LEARNING, LANGUAGE MODELS, TRIAGE, SMART ASSIGNMENT, ACTIVE LEARNING, HYBRID METHODOLOGY, CONTEXTUAL EMBEDDINGS.

ЗМІСТ

ВСТУП

Сервіс-деск системи сучасних організацій щоденно обробляють значні обсяги інцидентів, пов’язаних із відмовами сервісів, збоями інфраструктури, помилками користувачів та запитами на доступ. Це вимагає швидкого та ефективного механізму первинної обробки звернень, який включає класифікацію, пріоритизацію та маршрутизацію інцидентів до відповідних команд підтримки. Однією з ключових проблем, яку покликана вирішувати автоматизація цих процесів, є зменшення часу реакції на інциденти та зниження навантаження на операторів першої лінії підтримки, з одночасним підвищенням якості прийнятих рішень.

Завдяки застосуванню методів машинного навчання та сучасних моделей обробки природної мови автоматизована обробка інцидентів дає можливість:

-зменшити затримки під час взаємодії користувачів із службою підтримки та скоротити час до отримання початкового рішення;

- підвищити точність класифікації інцидентів за категоріями та пріоритетами без прямої участі оператора;

- знизити навантаження на Service Desk за рахунок автоматичного опрацювання типових звернень і більш обґрунтованої маршрутизації складних випадків до профільних команд;

- забезпечити більш рівномірний розподіл інцидентів між виконавцями завдяки використанню правил Smart Assignment.

Мета роботи – удосконалення процесу автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі гібридної моделі з узгодженням рішень ML та LLM компонентів

Об'єкт дослідження – процеси обробки інцидентів у сервіс-деск системі,

Предмет дослідження – моделі та методи автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі аналізу історичних тікетів і зворотного зв'язку ,зокрема етапи їхньої класифікації, пріоритизації та призначення виконавців.

Завдання дослідження:

1. Аналіз існуючих методів і підходів до автоматизованої класифікації, пріоритизації та маршрутизації інцидентів у сервіс-деск системах; виявлення їхніх переваг, недоліків та обмежень у контексті невеликих і середніх організацій.
2. Розроблення математичних моделей для автоматизованої обробки інцидентів: моделі ансамблевого узгодження рішень ML-класифікатора та LLM із пороговими значеннями впевненості, а також моделі Smart Assignment з багатофакторним скорингом виконавців.
3. Розроблення гібридної методики автоматизованої обробки інцидентів, що поєднує швидкий ML-класифікатор на текстових поданнях із мовною моделлю та механізмами triage; визначення критеріїв прийняття рішень для вибору фінальної категорії й пріоритету.
4. Впровадження розробленої методики у програмний прототип сервіс-деск системи: реалізація модулів класифікації, пріоритизації, triage та Smart Assignment, а також інтеграція цих модулів із інтерфейсом створення й обробки тікетів.
5. Проведення експериментального дослідження та тестування розробленої методики на тестовому наборі інцидентів; порівняння її ефективності з базовим підходом на основі традиційного ML-класифікатора за метриками якості класифікації, пріоритизації та коректності автоматичного призначення виконавців.

Методи дослідження включають аналіз наукових підходів до автоматизації роботи сервіс-деск систем, методи машинного навчання та обробки природної мови для побудови моделей класифікації й пріоритизації, проєктування та розроблення веб-застосунків, експериментальне моделювання та тестування програмних рішень, статистичний аналіз результатів експериментів.

Наукова новизна одержаних результатів полягає в розробці гібридної методики HYBRID v2.0, яка поєднує ML-класифікатор на контекстних

ембеддингах, локальну мовну модель, ансамблеві правила прийняття рішень та 4-рівневу схему активного навчання, що дозволяє досягти підвищення точності класифікації.

Практичне значення одержаних результатів грунтується можливістю впровадження розробленої методики в реальні сервіс-деск системи для зменшення часу обробки інцидентів, підвищення якості маршрутизації та зниження операційних витрат.

1 АНАЛІЗ АВТОМАТИЗОВАНОЇ ОБРОБКИ ІНЦИДЕНТІВ У SERVICE DESK ТА ОБҐРУНТУВАННЯ ГІБРИДНОЇ МЕТОДИКИ

У цьому розділі розглядається предметна область автоматизованої обробки інцидентів у сервіс-деск системах, аналізуються сучасні підходи до класифікації, пріоритизації та маршрутизації звернень, а також обґрунтовується необхідність запровадження гібридної методики HYBRID v2.0, що поєднує методи машинного навчання, контекстні ембеддинги та локальну мовну модель. Окрему увагу приділено ролі triage та Smart Assignment, а також 4-рівневій схемі активного навчання, яка дозволяє системі поступово самовдосконалюватися.

1.1 Аналіз процесу обробки інцидентів у Service Desk

У сучасних організаціях сервіс-деск виконує роль єдиної точки контакту між користувачами та IT-підрозділом. Кожен інцидент реєструється у вигляді тікета з коротким заголовком, текстовим описом проблеми, а іноді — із додатковими атрибутами (сервіс, додаток, підрозділ, критичність, вкладення). Після реєстрації інцидент має пройти низку етапів:

— класифікація за категорією (мережа, робоче місце, прикладні системи, доступ, білінг, інші);

— визначення пріоритету (P1–P3) з урахуванням бізнес-впливу та терміновості;

— маршрутизація до відповідного департаменту підтримки;

— призначення конкретному виконавцю;

— за потреби — ескалація та triage складних або неоднозначних випадків.

У традиційних системах більшість цих рішень приймається операторами вручну, що призводить до суб’єктивності класифікації, помилок маршрутизації, повторних переадресацій тікетів між командами та збільшення часу реакції на критичні інциденти [4].

На рисунку. 1.1 подано схему інтеграції автоматизації: після створення тікета він проходить автоматичну класифікацію та пріоритизацію, за потреби — triage, далі через Smart Assignment призначається відповідній команді підтримки..

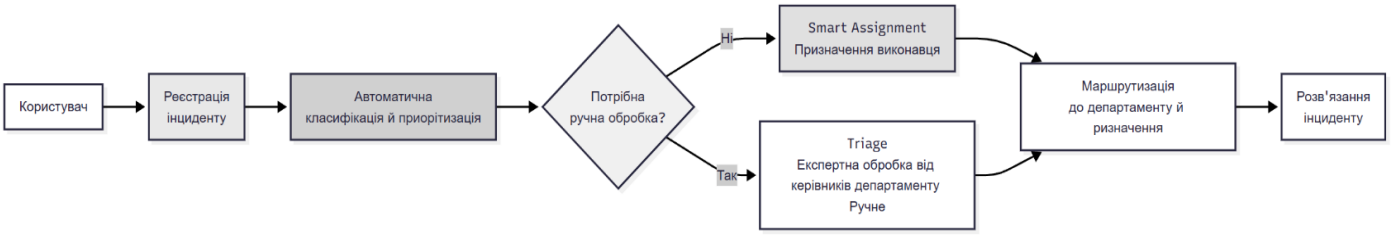


Рис. 1.1 Узагальнена схема автоматизованої обробки інцидентів у сервіс-деск системі

Процес управління інцидентами у сервіс-деск системах регламентується практиками IT Service Management, зокрема бібліотекою ITIL, яка визначає етапи реєстрації, класифікації, пріоритизації та маршрутизації звернень користувачів [1]. Сучасні дослідження у сфері AIOps підкреслюють важливість автоматизованого triage як механізму зменшення впливу інцидентів та підвищення керованості процесу підтримки [4; 5; 22].

1.2 Традиційні ML-підходи до класифікації та пріоритизації інцидентів

Першим етапом автоматизації є впровадження моделей машинного навчання для класифікації категорії та пріоритету тікетів. Найчастіше такі моделі ґрунтуються на класичних текстових поданнях і лінійних класифікаторах.

Практична ефективність застосування методів машинного навчання для автоматизованої класифікації та маршрутизації інцидентів підтверджується низкою досліджень, присвячених автоматизації Service Desk та Helpdesk-систем [2; 3; 18]. При цьому класичні підходи до текстової класифікації залишаються базовим рішенням завдяки відтворюваності результатів і помірній обчислювальній складності [6; 7].

У низці робіт запропоновано підхід, де текст інциденту перетворюється у вектор за допомогою TF–IDF, після чого застосовуються Logistic Regression, SVM, Random Forest або ансамблеві моделі для визначення цільового класу [1–3], [5]. Показано, що навіть відносно прості лінійні моделі здатні забезпечити високу точність класифікації IT service desk тікетів і суттєво скоротити обсяг ручної роботи операторів [1], [2].

Загальна структура такого ML-пайплайну включає:

— попередню обробку тексту (нормалізація, токенізація, очищення від шуму);

— перетворення тексту в числовий вектор (TF–IDF або Bag-of-Words);

— навчання багатокласового класифікатора на історичних тікетах;

— обчислення метрик якості (accuracy, F1-score, precision, recall) та бізнес-показників (частка правильно маршрутизованих інцидентів, кількість повторних переадресацій) [3], [5].

Порівняння традиційних підходів із ручною обробкою та гібридними методами наведено в табл. 1.1, де зазначені їх основні переваги, недоліки та типові сценарії використання.

Таблиця 1.1

Порівняння підходів до обробки інцидентів у сервіс-деск системі

Підхід	Опис	Переваги	Недоліки
Ручна обробка	Усі рішення щодо категорії, пріоритету та виконавця приймає оператор Service Desk.	Гнучкість; урахування контексту, який важко формалізувати; можливість нестандартних рішень.	Низька швидкість; висока вартість ручної праці; суб’єктивність рішень; слабка масштабованість.

Продовження таблиці 1.1

Порівняння підходів до обробки інцидентів у сервіс-деск системі

Підхід	Опис	Переваги	Недоліки
Класичний ML TF–IDF + LR/SGD	Автоматична класифікація тікетів за категоріями чи пріоритетами на основі векторизації тексту та лінійних моделей.	Висока швидкість передбачень; простота реалізації; робота у CPU-only середовищі; прозорість моделі.	Обмежене врахування контексту; чутливість до якості тексту; проблеми з рідкісними класами та новими типами інцидентів.
BERT/ембеддинги + ML	Використання трансформерних ембеддингів (BERT, MiniLM, SBERT) з подальшою класифікацією лінійною моделлю.	Краще захоплення семантики; стійкість до перефразувань і помилок; вища якість класифікації.	Складніша архітектура; потреба в налаштуванні ембеддингів; вища вартість обчислень порівняно з TF–IDF.
Тільки LLM	Усі рішення щодо категорії, пріоритету та маршрутизації приймаються великою мовною моделлю без окремого ML-класифікатора.	Глибоке розуміння контексту; можливість пояснювати рішення; гнучка адаптація до нових типів інцидентів.	Висока затримка відповіді; значні обчислювальні витрати; нестабільність результатів;
Гібридний прототип HYBRID v2.0	Поєднання ML-класифікатора на ембеддингах, локальної LLM, triage, 4-рівневого активного навчання та Smart Assignment.	Баланс якості й продуктивності; можливість поетапного впровадження; самонавчання на реальних тікетах; зменшення ручної роботи.	Більш складна реалізація та підтримка; потреба в журналюванні рішень і налаштуванні порогів triage.

Водночас традиційні ML-рішення мають обмеження:

— залежність від якості та повноти текстового опису;

— невисока стійкість до неформальних формулювань, помилок і змішаних мов;

— проблеми з рідкісними класами (дисбалансом) і новими типами інцидентів [1–3].

Ці недоліки стимулювали перехід до контекстних ембеддингів та гібридних knowledge-based/ML-схем.

Таким чином, класичні ML-підходи є ефективними для автоматизованої обробки типових інцидентів, однак демонструють обмеження у випадках неоднозначних або нетипових звернень, що обумовлює доцільність поєднання їх з іншими підходами до аналізу тексту.

1.3 Контекстні ембеддинги, трансформери та гібридні підходи

Ефективність автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі визначається насамперед тим, наскільки адекватно текст звернення подано у вигляді числових ознак для подальшого навчання моделей. У практичному середовищі Service Desk тексти мають специфіку: вони часто короткі, містять технічні терміни, коди помилок, назви сервісів, скорочення, фрагменти англійською, а також варіативні формулювання одних і тих самих проблем. Саме тому в межах розроблюваної методики ключовою є не лише “наявність ML-класифікатора”, а й вибір такого представлення тексту, яке забезпечує стійке узагальнення для нових звернень і зменшує частку помилок у нетипових випадках.

Класичні підходи на основі bag-of-words/TF-IDF будують ознаки, що відображають частоти термінів або n-грам. Вони добре масштабуються та швидко працюють у CPU-only середовищі, що важливо для експлуатації. Однак такі ознаки переважно відображають лексичні збіги, слабо враховують контекст і семантичні зв’язки між словами, а тому є чутливими до перефразувань, синонімів та “шуму” в описі інциденту. Для сервіс-деск сценаріїв це проявляється особливо різко: два

звернення з однаковою суттю можуть мати низьку схожість у TF-IDF-просторі лише через різний словник або порядок слів. Як наслідок, ускладнюється стабільне узагальнення та зростає ризик помилки маршрутизації або визначення пріоритету у випадках, коли помилка є дорогою для бізнес-процесу.

Подальший розвиток NLP призвів до поширення трансформерних моделей, які формують контекстно-залежні представлення тексту. В основі BERT-класу лежить ідея двоспрямованого кодування, коли представлення токена визначається його оточенням у реченні, а не є фіксованим “словниковим” вектором [41]. Для задач сервіс-деску це принципово: значення термінів і абревіатур часто залежить від контексту (“access”, “account”, “auth”, “token”, “VPN”), а ключовий зміст може міститися в короткому фрагменті або одному повідомленні про помилку. Відповідно, контекстні представлення дають можливість точніше відобразити семантику інциденту і зменшити залежність від “точного збігу слів”.

Однак пряме використання повнорозмірних трансформерів у production-контурі (особливо без GPU) може створювати надмірні обчислювальні витрати. Тому в межах методики доцільним є компромісний, але інженерно практичний підхід: трансформер використовується як енкодер для отримання щільного семантичного вектора (embedding), а класифікація/пріоритизація виконується легким ML-класифікатором. Такий підхід широко відомий як використання “sentence embeddings” у поєднанні з моделлю верхнього рівня. Концептуальну основу цієї практики формалізовано в Sentence-BERT, де трансформер адаптовано для формування порівнюваних реченнєвих векторів у спільному семантичному просторі [42]. Для сервіс-деску це важливо, оскільки тікети часто є короткими й близькими за змістом, але різними за формулюванням; саме sentence-embeddings зменшують цю проблему, підвищуючи стійкість класифікації до перефразувань.

З практичної точки зору в методиці раціонально застосовувати компактні трансформерні архітектури, отримані шляхом дистиляції, які забезпечують нижчу затримку та менші ресурси при збереженні ключових властивостей контекстного кодування. Прикладом такого підходу є MiniLM, де стиснення досягається дистиляцією механізмів self-attention, що дозволяє використовувати модель у

продуктивних контурах із обмеженнями CPU-only [44]. Для мультимовного Service Desk це додатково важливо, оскільки звернення можуть бути змішаними за мовою, а модель має формувати узгоджені embeddings незалежно від того, чи домінує українська, чи англійська технічна лексика.

З огляду на ці вимоги, доцільним є застосування мультимовної sentence-transformers моделі paraphrase-multilingual-MiniLM-L12-v2, яка формує компактні embeddings (384 виміри) та орієнтована на збереження семантики при перефразуванні [46], [47]. Документація Sentence-Transformers і модельна картка підтверджують призначення цієї моделі як універсального енкодера для перетворення речень/абзаців у щільний семантичний простір і мультимовний характер навчання [46], [47]. Саме така конфігурація є узгодженою з вимогами методики: отриманий embedding може використовуватися як спільна ознакова база для окремих моделей прогнозування категорії та пріоритету, забезпечуючи уніфікований семантичний шар для подальшого ансамблювання рішень.

Окрім SBERT-класу, у сучасних системах активно застосовуються retrieval-орієнтовані embeddings, зокрема сімейство E5, де ефективність забезпечується weakly-supervised contrastive pre-training [43]. Такі моделі часто демонструють високу якість у задачах пошуку та семантичного зіставлення. Проте з позиції експлуатації в CPU-контурі важливими є розмірність і вартість обчислень: наприклад, multilingual-e5-base використовує embedding size 768, що збільшує як обсяг ознак, так і потенційні витрати на інференс та зберігання [48]. У методиці, орієнтованій на інтеграцію в реальну сервіс-деск систему, такі параметри мають оцінюватися разом із вимогами до затримки обробки й масштабованості. Узагальнене порівняння найбільш релевантних підходів до подання тексту для задач Service Desk наведено в табл. 1.2. Порівняння виконано з урахуванням вимог методики: стійкість до перефразувань і шуму, мультимовність, придатність до CPU-only експлуатації та інженерна простота інтеграції.

Таблиця 1.2

Порівняння підходів до подання тексту для задач класифікації та пріоритизації інцидентів Service Desk

Критерій	TF-IDF (baseline)	SBERT: paraphrase-multilingual-MiniLM-L12-v2	multilingual-e5-base	RuBERT / мовно-спеціалізований BERT	Universal Sentence Encoder (USE)
Тип представлення	Розріджені лексичні ознаки (терми, n-грами)	Щільні sentence embeddings (семантичний простір) [42]	Щільні embeddings (retrieval/semantic matching) [43]	Контекстні представлення (BERT-клас) [45]	Sentence embeddings універсального призначення [49]
Урахування контексту	Низьке: залежить від “точних” слів	Високе: контекст + pooling; орієнтація на подібність/перефразування [42]	Високе: контрастивне навчання для семантичної близькості [43]	Високе: контекстна токенізація/кодування [45]	Середнє/високе: універсальне семантичне кодування [49]
Стійкість до перефразування	Низька (семантика не моделюється напряму)	Висока (paraphrase-орієнтоване навчання) [47]	Висока (retrieval-focus) [43]	Середня/висока (залежить від домену та мови) [45]	Середня/висока для загальних текстів [49]
Стійкість до «шумних» і коротких тікетів	Часто знижується через розрідженість і технічні скорочення	Вища стійкість за рахунок семантики; краще узагальнення на коротких описах [42]	Загалом стійке, але з вищими ресурсними витратами [48]	Варіюється: ризик деградації на mixed-language і технічних фрагментах	Стабільне для загальних текстів; доменні технічні скорочення можуть вимагати адаптації
Мультимовність (UA/EN + технічні фрагменти)	Формально можлива, але без семантичного вирівнювання між мовами	Підтримка 50+ мов, спільний простір [47]	Мультимовна модель; embedding size 768 [48]	Переважно одна мова/обмежений набір мов [45]	Є мультимовні варіанти; базово універсальний підхід [49]

Продовження таблиці 1.2

Порівняння підходів до подання тексту для задач класифікації та пріоритизації інцидентів Service Desk

Критерій	TF-IDF (baseline)	SBERT: paraphrase-multilingual-MiniLM-L12-v2	multilingual-e5-base	RuBERT / мовно-спеціалізований BERT	Universal Sentence Encoder (USE)
Розмірність ознак	Дуже велика (залежить від словника)	384-D [46]	768-D [48]	Типово 768-D (base-BERT; залежить від моделі) [45]	512-D [49]
Продуктивність у CPU-only	Найвища (дуже швидкий inference)	Висока для трансформера за рахунок дистиляції MiniLM [44]	Середня/нижча (дорожче, ніж MiniLM) [48]	Часто нижча (важча архітектура) [45]	Середня [49]
Потреба у донавчанні під домен	Низька (baseline), але якість обмежена	Часто достатньо “frozen embeddings” + легкий класифікатор [42]	Може вимагати тонкої доменної настройки для максимального ефекту [43]	Часто потребує доменної адаптації	Може працювати без донавчання, але доменна адаптація інколи потрібна
Інженерна складність інтеграції	Низька (векторизація + модель)	Середня (embedding-service + downstream-моделі)	Середня/вища (ресурси, inference)	Вища (важча модель, залежності)	Середня
Рекомендоване використання в методиці	Baseline та порівняльні експерименти	Основний варіант для production-контурів з мультимовністю та короткими тікетами	Альтернатива при пріоритеті retrieval-якості й наявності ресурсів	Доцільно при домінуванні однієї мови та наявності ресурсів	Універсальний варіант, якщо потрібен загальний sentence encoder

Як видно з таблиці. 1.2, TF-IDF забезпечує мінімальні витрати ресурсів, проте поступається за стійкістю до перефразувань, шуму та мультимовних сценаріїв. З огляду на вимоги методики автоматизованої класифікації та пріоритизації

інцидентів, найбільш збалансованим рішенням є використання мультимовного SBERT-енкодера на основі архітектури MiniLM із подальшою класифікацією легкими ML-моделями.

Альтернативною лінією є мовно-спеціалізовані BERT-моделі (наприклад, RuBERT-підхід), які можуть бути ефективними в моно-мовних сценаріях та при доменній адаптації [45]. Водночас для Service Desk типовою є домішка англомовних назв продуктів і системних повідомлень, що знижує практичну перевагу вузькомовних моделей. Тому в межах методики пріоритетним є мультимовний енкодер із достатньою семантичною виразністю, який не обмежує систему одним мовним сегментом.

Окремо доцільно враховувати й універсальні sentence encoders, такі як Universal Sentence Encoder, який кодує текст у 512-вимірні embeddings [49]. USE може використовуватися як загальне рішення для семантичних задач, однак для сервіс-деск сценарію з високою часткою перефразувань та технічних домішок більш релевантним є саме paraphrase-орієнтований клас Sentence-Transformers [42], [47].

Разом із переходом на контекстні embeddings для практичної системи важливо забезпечити контрольованість та керованість поведінки в “прикордонних” випадках. Тому в методиці доцільним є використання гібридних підходів, де ML-рішення доповнюється експертними правилами та доменними обмеженнями. Така композиція knowledge-based + ML дозволяє:

— враховувати бізнес-критичні правила наприклад , окреме трактування інцидентів VIP-користувачів або критичних сервісів;

— зменшувати частку помилок у неоднозначних сценаріях, де текстового сигналу недостатньо;

— забезпечувати контрольовану поведінку системи при появі нових типів інцидентів або зміні процесів.

Отже, аналіз методів подання тексту та трансформерних моделей у даному підрозділі формує обґрунтування вибору семантичного шару методики: контекстні embeddings як базова ознакова основа в поєднанні з легкими ML-класифікаторами

для категоризації та пріоритизації, а також із гібридними правилами для зменшення ризику помилок у критичних або нетипових випадках. Така постановка відповідає практичним вимогам сервіс-деск середовища (мультимовність, короткі тексти, технічний шум, CPU-обмеження) і створює основу для подальшого опису алгоритмічних стратегій прийняття рішення та інтеграції компонентів у наступних розділах роботи.

1.4 LLM, triage та автоматизоване прийняття рішень

Попри значні успіхи ML-класифікаторів на ембеддингах, у практиці сервіс-деск завжди існує частка інцидентів, які є:

— неоднозначними або недостатньо описаними;

— прикордонними між кількома категоріями;

— нетиповими для наявної навчальної вибірки.

Для таких випадків усе ширше використовуються великі мовні моделі (LLM), здатні інтерпретувати складні текстові описи, узагальнювати ключові ознаки інциденту та пропонувати обґрунтовані категорії й пріоритети [10], [11]. У роботі COMET продемонстровано, що LLM, інтегровані в процес triage інцидентів, дозволяють підвищити точність та інтерпретованість рішень, а також скоротити час до виявлення та локалізації проблеми в cloud-системах [10].

Інші дослідження розглядають LLM як частину цілісної AI-дрівної системи incident management, яка допомагає інженерам при аналізі мережевих подій, пошуку root cause, генерації рекомендацій та узагальнень [11].

З точки зору автоматизації triage особливу роль відіграють підходи, орієнтовані на:

— виявлення інцидентів, які з високою ймовірністю потребують ручної обробки;

— автоматичне перекидання (transfer) тікетів між чергами у великих системах, якщо початковий assignment виявився помилковим [12].

У роботі DeepTriage запропоновано ML/ensemble-підхід до автоматизованої підтримки transfer-рішень, що дозволяє зменшити кількість ручних переадресацій і пришвидшити доставку інциденту до «правильної» команди [12].

У гібридній методиці HYBRID v2.0 використовується аналогічна ідея:

— базовий ML-класифікатор формує прогноз категорії та пріоритету із оцінкою впевненості;

— локальна LLM (через Ollama) формується як альтернативний прогноз (second opinion) у межах того ж сценарію обробки тікета (за умови, що skip_llm не активований);

— triage-блок визначає, чи є фінальне рішення достатньо надійним для автоматичного прийняття, або інцидент слід передати оператору.

Останні дослідження демонструють потенціал великих мовних моделей для підтримки triage та аналізу складних інцидентів у виробничих системах [12; 17]. Водночас у наукових роботах підкреслюється доцільність використання LLM як допоміжного експертного агента у межах керованих human-in-the-loop систем, що дозволяє зменшити ризик неконтрольованих помилок автоматизації [15; 16; 23]. Двоагентну схему взаємодії ML та LLM,а також роль triage показано на рисунку. 1.2

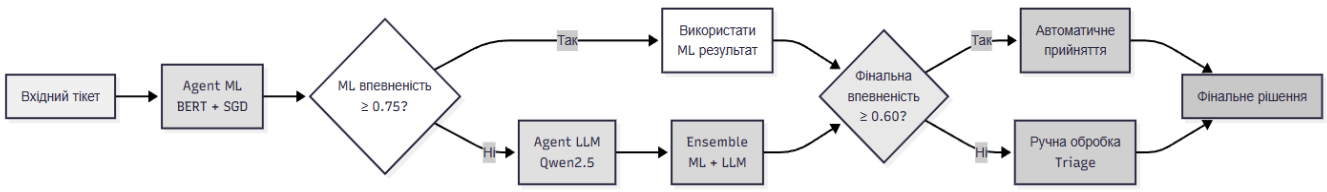


Рис. 1.2. Двоагентна схема прийняття рішень ML–LLM з triage

1.5 Автоматизоване призначення виконавців

Одним із критично важливих етапів обробки інцидентів у сервіс-деск системах є коректне призначення відповідального виконавця. Навіть за умови правильної класифікації та визначення пріоритету, помилки на етапі маршрутизації інцидентів призводять до повторних перенаправлень (re-assignments), збільшення часу реакції та вирішення, нерівномірного навантаження між командами, а також зниження узгодженості процесу підтримки. На практиці це прямо впливає на показники SLA/OLA, якість сервісу та загальну операційну ефективність.

Проблема автоматизованого призначення тісно пов’язана з напрямами ticket routing, assignee recommendation та bug triage, які активно досліджуються в літературі [26–30]. Однією з ранніх робіт є підхід EasyTicket (Shao et al., 2008) [26], де маршрутизація звернень поєднана з рекомендаціями можливих рішень на основі історії попередніх виконань і повторюваних шаблонів розв’язання. Такий клас підходів демонструє високу точність у “зрілих” організаціях із великим репозиторієм інцидентів, однак ускладнюється у випадку cold start (малий обсяг навчальних даних), частих змін процесів або появи нових сервісів.

Подальші дослідження підкреслюють важливість комплексного підходу, де враховуються як текстові ознаки звернення, так і контекстні атрибути (метадані інциденту та інформація про агента). Зокрема, Agarwal et al. (2020) [27] розглядають end-to-end цикл впровадження AI для автоматизованого призначення helpdesk-звернень і показують, що використання контекстних ознак (категорія/тип запиту, текст, часові характеристики, параметри виконавця) підвищує узгодженість призначень і скорочує час реакції команди. У прикладній площині для ITSM також характерна вимога до керованості та відтворюваності рішень, оскільки операційні команди мають пояснювати, чому саме певний виконавець був обраний для конкретного інциденту особливо для P1/P2.

З розвитком глибоких мовних моделей з’явилися підходи, де текст інциденту обробляється семантично (а не лише через ключові слова), що критично для коротких, “шумних” або технічних описів. Трансформерні архітектури, зокрема

BERT [28], дозволяють підвищити якість розуміння контексту звернення та узгодження його зі “знаннями” про профіль виконавця або історичні патерни маршрутизації. Окремою практично важливою проблемою є дефіцит розмічених даних і різноманітність формулювань тікетів. Тому техніки розширення даних, зокрема EDA [29], застосовують для генерації варіантів текстів і покращення стійкості моделей до перефразувань та стилістичних відмінностей.

У більшості сучасних робіт також підкреслюється, що жоден один підхід (суто правиловий або суто статистичний) не забезпечує достатньої гнучкості для виробничого ITSM-середовища, де одночасно присутні: неоднорідні канали звернень, дрейфи тематики інцидентів, зміни команд/ролей, “пікові” навантаження та вимоги до контрольованості рішень. Тому доцільним вважається гібридний механізм призначення, який поєднує інтерпретовані критерії (компетенції, департамент, поточне навантаження, доступність) із ML/семантичними компонентами, що враховують історичну успішність призначень та подібність звернень [31–36].

У межах даної роботи реалізовано механізм Smart Assignment, що концептуально продовжує згадані підходи, але адаптований під архітектуру сервіс-деск прототипу та вимоги керованості. На відміну від статичних правил або повністю “чорних” ML-рішень, Smart Assignment забезпечує:

— багатофакторність: одночасне врахування спеціалізації, департаменту/категорії, доступності, поточного навантаження, історичної результативності та ML-ймовірності призначення;

— пояснюваність: можливість декомпозиції фінального скорингу на вагові внески факторів (що саме “підштовхнуло” вибір);

— контроль впевненості: за низької впевненості або конфліктів сигналів — ініціація ручної перевірки (triage) для зменшення ризику помилкових автопризначень;

— готовність до донавчання: накопичення зворотного зв’язку та даних про якість призначень як основи для подальшого вдосконалення.

Таким чином, Smart Assignment формує більш прозору й керовану схему маршрутизації інцидентів: автоматизація застосовується там, де система має достатню впевненість і пояснюваність, а triage забезпечує підстрахування для складних випадків, зменшуючи кількість повторних перенаправлень і підвищуючи стабільність процесу підтримки.

Таблиця 1.3

Основні фактори алгоритму Smart Assignment

Фактор	Опис	Вплив на рішення
Категорія інциденту	Тип інциденту, визначений гібридною методикою	Визначає профіль необхідної спеціалізації
Пріоритет	Рівень критичності інциденту	Впливає на черговість обробки
Спеціалізація виконавця	Набір навичок та досвід виконавця	Забезпечує відповідність задачі
Поточне навантаження	Кількість активних інцидентів у виконавця	Запобігає перевантаженню
Контекстні ознаки	Додаткові характеристики інциденту	Уточнює вибір виконавця

1.6 Введення та структура гібридної методики HYBRID v2.0

У межах даної роботи запропоновано гібридну методику автоматизованої класифікації та пріоритизації інцидентів HYBRID v2.0, яка поєднує класичні методи машинного навчання, великі мовні моделі та механізми активного навчання. Методика орієнтована на підвищення обґрунтованості та автоматизованості прийняття рішень і зменшення кількості невизначених або помилкових випадків у сервіс-деск системах.

Запропонована гібридна методика ґрунтується на поєднанні ансамблевих методів машинного навчання та механізмів активного навчання, ефективність яких підтверджена у численних дослідженнях [10; 11]. Додаткове використання

пояснювальних підходів дозволяє підвищити інтерпретованість прийнятих рішень і відповідає сучасним вимогам до довіри в автоматизованих системах [25].

Основою методики є базовий ML-класифікатор, навчений на історичних даних інцидентів. Для представлення текстових описів інцидентів використовуються контекстні ембеддинги, що дозволяють враховувати семантичний зміст звернень. ML-модель забезпечує швидке та відтворюване прийняття рішень для типових інцидентів, а також формує числову оцінку впевненості результату.

У випадках, коли впевненість ML-класифікатора є недостатньою або виникає потенційна неоднозначність рішення, до процесу залучається велика мовна модель, яка виконує роль другого експертного агента. LLM аналізує текст інциденту з урахуванням контексту та формує альтернативну пропозицію щодо категорії та пріоритету. Рішення ML та LLM об’єднуються за допомогою ensemble-логіки, що дозволяє підвищити стабільність результатів.

Важливою складовою методики є механізм triage, який використовується для обробки конфліктних або низьковпевнених випадків. Triage розглядається як контрольний етап, що запобігає прийняттю необґрунтованих автоматизованих рішень та забезпечує залучення експертної оцінки.

Методика HYBRID v2.0 також включає механізми активного навчання, реалізовані у вигляді 4-рівневої системи зважування зворотного зв’язку. Залежно від джерела підтвердження формується різний внесок у процес подальшого навчання моделей, що дозволяє покращувати якість класифікації навіть за обмеженої кількості навчальних даних.

Завершальним елементом гібридної методики є інтеграція алгоритму Smart Assignment, який забезпечує автоматизоване призначення інцидентів відповідним виконавцям з урахуванням категорії, пріоритету та контексту інциденту. Таким чином, методика HYBRID v2.0 формує цілісний підхід до автоматизованої обробки інцидентів.

Таблиця 1.4

Структура гібридної методики HYBRID v2.0

Компонент	Функціональне призначення	Роль у методиці
ML-класифікатор	Автоматична класифікація категорії та пріоритету інцидентів	Базове швидке та відтворюване прийняття рішень
LLM	Контекстний аналіз складних інцидентів	Підвищення обґрунтованості рішень
Ensemble-логіка	Об’єднання рішень ML та LLM	Зменшення конфліктних випадків
Triage	Виявлення невизначених інцидентів	Контроль ризику помилкових рішень
4-Tier Active Learning	Адаптивне донавчання моделей	Поступове підвищення якості класифікації
Smart Assignment	Автоматизоване призначення виконавців	Завершення циклу обробки інцидентів

1.7 Розширений огляд наукових досліджень та існуючих підходів до автоматизації в сервіс-деск системах

1.7.1 Класичні підходи до автоматизації сервіс-деск процесів

Перші наукові та прикладні роботи, присвячені автоматизації сервіс-деск систем, були орієнтовані на застосування правил, експертних систем та жорстко заданих бізнес-логік для первинної обробки інцидентів. У таких підходах використовувалися фіксовані словники ключових слів, регулярні вирази, а також ручно сформовані правила, які дозволяли визначати категорію або пріоритет звернення користувача [1]. Подібні рішення широко застосовувалися на ранніх етапах розвитку IT Service Management, зокрема у рамках рекомендацій ITIL.

Основною перевагою правил-орієнтованих підходів була їх прозорість і повна контрольованість: кожне рішення системи можна було пояснити набором

правил, що спрацювали. Однак такі системи мали суттєві обмеження, зокрема низьку масштабованість, чутливість до змін у термінології та необхідність постійного ручного оновлення правил. Зі збільшенням кількості інцидентів та різноманітності їх описів підтримка таких систем ставала економічно недоцільною.

Подальший розвиток досліджень призвів до використання статистичних методів машинного навчання, які навчаються на історичних даних сервіс-деск систем. У роботах [2], [3] показано, що застосування алгоритмів Logistic Regression, Naive Bayes та Support Vector Machines у задачах класифікації інцидентів дозволяє автоматизувати до 60–70% первинних звернень користувачів. Такі моделі демонструють хорошу швидкодію та стабільність результатів, що є критично важливим для систем з великим потоком тікетів.

Водночас, автори зазначають, що якість роботи класичних ML-моделей суттєво залежить від способу представлення тексту, а також від повноти та якості навчального набору даних. Це обумовлює необхідність подальших досліджень у напрямі більш ефективних методів обробки текстових описів інцидентів.

Рисунок. 1.3 ілюструє типовий pipeline класичної сервіс-деск ML-системи, у якій текст інциденту перетворюється у набір ознак, що подаються на вхід класифікатору для визначення категорії та пріоритету.

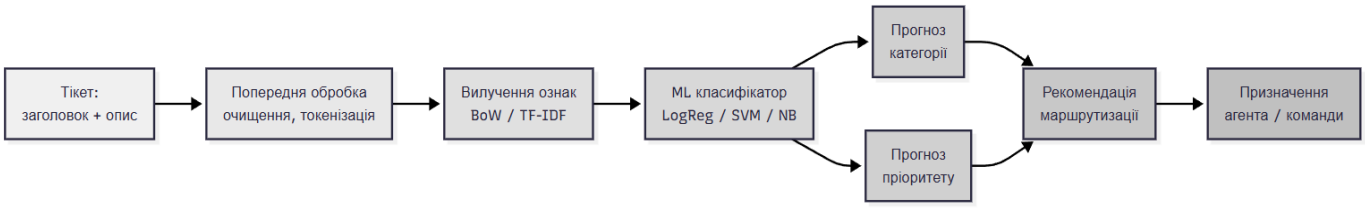


Рис. 1.3. Класичний ML-pipeline для Service Desk

1.7.2 Методи представлення тексту та їх вплив на якість класифікації

Ключовим елементом будь-якої ML-системи для аналізу текстових звернень є спосіб представлення тексту у числовій формі. У ранніх дослідженнях домінували

підходи на основі bag-of-words та TF-IDF, які дозволяють відобразити текст інциденту як вектор частот термінів [4]. Такі методи є простими в реалізації та добре масштабуються, однак вони не враховують порядок слів та семантичні зв’язки між ними.

Обмеження TF-IDF стають особливо помітними у задачах сервіс-деск, де описи інцидентів часто є короткими, неструктурованими та містять технічні скорочення або помилки. У таких умовах однакові за змістом інциденти можуть мати суттєво різні векторні представлення, що негативно впливає на якість класифікації.

Подальші дослідження показали, що використання контекстних мовних моделей, зокрема BERT та його похідних, дозволяє значно підвищити якість аналізу текстів сервіс-деск звернень [5], [6]. Контекстні ембеддинги враховують як локальний, так і глобальний контекст слів у реченні, що дає змогу точніше відображати семантику інцидентів.

Результати експериментів, наведених у відповідних роботах, демонструють, що перехід від TF-IDF до контекстних ембеддингів забезпечує стабільніше узагальнення на нові дані та зменшує кількість помилок у складних або нетипових випадках. Рисунок 1.4 та таблиця 1.5 наочно відображають концептуальні відмінності між цими підходами.

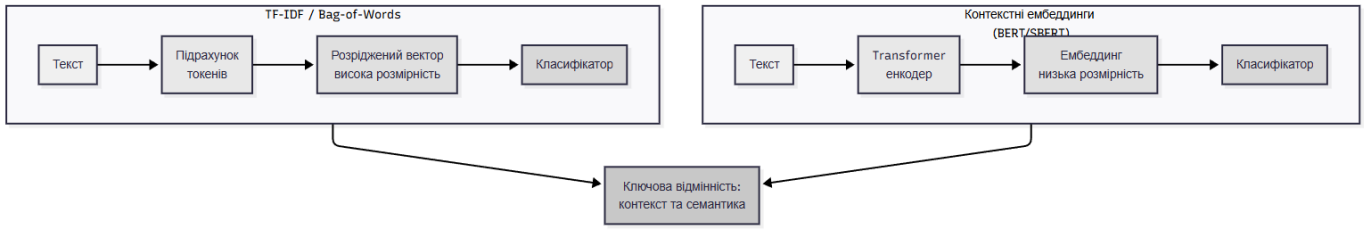


Рис. 1.4. Порівняння TF-IDF та контекстних ембеддингів

Таблиця 1.5

Порівняння TF-IDF та BERT-ембеддингів у задачі сервіс-деску

Критерій	**TF-IDF + класичний класифікатор**	**BERT/SBERT-ембеддинги + легкий класифікатор**
Представлення тексту	Розріджені лексичні ознаки (терми/н-грами) [6]	Щільні контекстні вектори, семантика в реченні [8], [9]
Контекст і синонімія	Слабко враховує контекст, залежить від «точних» слів [6]	Краще працює з контекстом, синонімами, варіативністю формулювань [8]
Якість на «шумних» тікетах	Часто знижується на коротких/неформальних описах	Стійкіше до шуму, коротких описів, змішаних мов [8], [11]
Обчислювальні витрати	Низькі, швидке навчання й inference	Вищі витрати на ембеддинг; можливий режим «frozen embeddings» [9]
Інженерна складність	Проста реалізація, легка підтримка	Потрібна модель ембеддингу, контроль залежностей та ресурсів [9]
Рекомендоване застосування	Baseline, прості класи з характерними ключовими словами [2]	Фінальна модель для складних кейсів/вищої якості; поєднання з triage [15], [16]

1.7.3 Автоматична маршрутизація та triage інцидентів

Окрім класифікації та пріоритизації, важливим завданням сервіс-деск систем є автоматична маршрутизація інцидентів до відповідних груп підтримки. У роботі DeepTriage [7] запропоновано підхід, заснований на нейронних мережах, який використовує історичні дані для визначення команди, відповідальної за обробку інциденту. Автори показали, що автоматизація цього етапу дозволяє скоротити час призначення виконавця та зменшити кількість повторних перенаправлень тікетів.

Разом з тим, дослідники наголошують на необхідності обережного застосування автоматичної маршрутизації, особливо у випадках критичних або нестандартних інцидентів. У зв’язку з цим у наукових публікаціях все частіше розглядається концепція triage — процесу відокремлення інцидентів, які можуть бути оброблені автоматично, від тих, що потребують ручної перевірки [8].

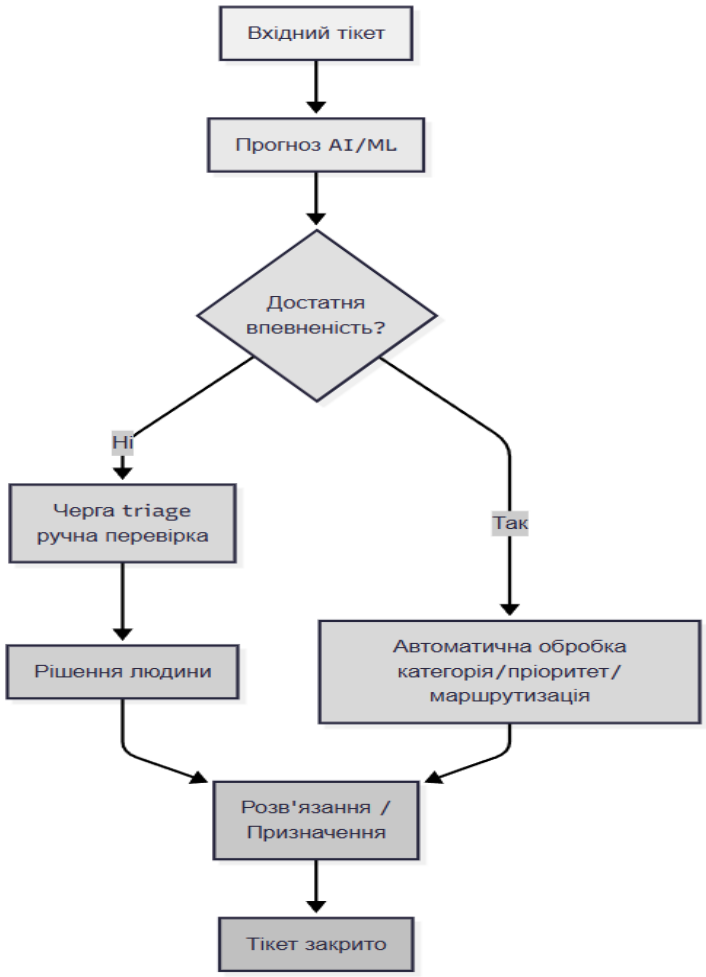


Рис. 1.5. Схема прийняття рішень з triage

Механізми triage зазвичай ґрунтуються на рівні впевненості моделі, історії помилок або ступені розбіжності між альтернативними прогнозами.

Рисунок 1.5 демонструє типову логіку прийняття рішення щодо автоматичної або ручної обробки інциденту.

1.7.4 Human-in-the-loop та активне навчання

Сучасні дослідження вказують на те, що найбільш ефективними є гібридні підходи типу human-in-the-loop, у яких автоматизовані рішення доповнюються ручним контролем експертів [9]. У таких системах людина не лише виправляє помилки, але й виступає джерелом зворотного зв’язку для подальшого навчання моделей.

Методи активного навчання (Active Learning) дозволяють оптимізувати процес залучення експертів, концентруючи їх увагу на найбільш складних або невизначених випадках [10]. Замість суцільної ручної анотації система вибірково передає на перевірку ті інциденти, для яких рівень впевненості є найнижчим.

Рисунок 1.6 ілюструє типовий цикл активного навчання, у якому результати прогнозування, людський зворотний зв’язок та повторне навчання моделі утворюють замкнений контур постійного вдосконалення системи.

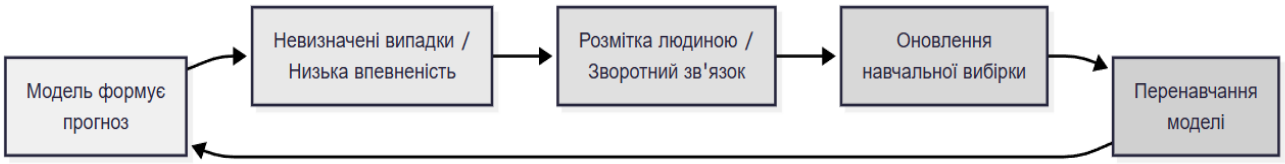


Рис. 1.6. Цикл активного навчання (Active Learning)

1.7.5 Великі мовні моделі в сервіс-деск системах

Останні роки характеризуються активним дослідженням можливостей великих мовних моделей у задачах аналізу звернень користувачів та підтримки прийняття рішень [11], [12]. LLM здатні обробляти складні та неструктуровані тексти, формувати альтернативні гіпотези та генерувати пояснення власних рішень.

Водночас, у науковій літературі підкреслюється, що використання LLM у повністю автономному режимі є ризикованим через нестабільність відповідей, складність формальної валідації та можливість генерації некоректних результатів [13]. Тому більшість дослідників розглядає LLM як допоміжний компонент, який має працювати у поєднанні з більш контрольованими ML-моделями.

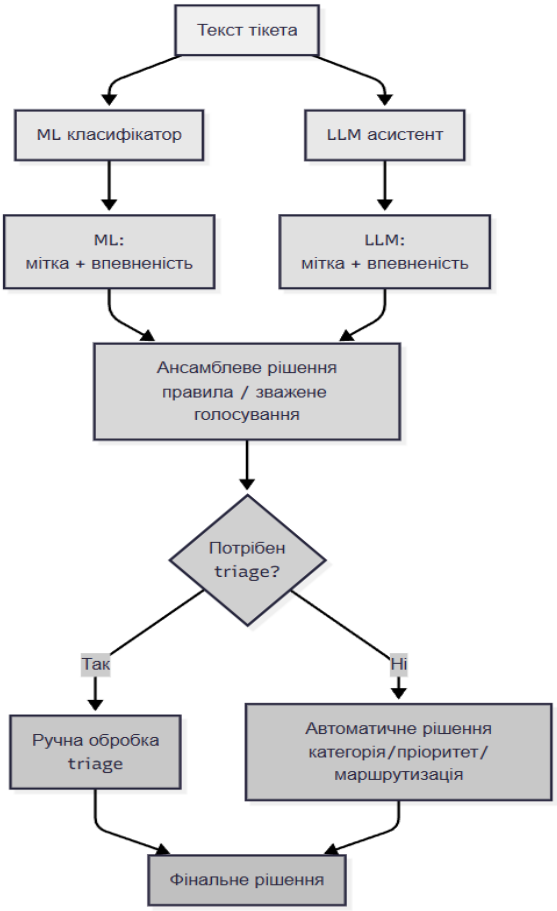


Рис. 1.7. Ансамбль ML + LLM з контрольованим triage

Найбільш перспективним підходом вважається використання ансамблевих систем, у яких результати ML та LLM комбінуються з урахуванням рівня впевненості та додаткових правил прийняття рішень. Рисунок 1.7 ілюструє узагальнену архітектуру такого ансамблю має працювати у поєднанні з більш контрольованими ML-моделями.

1.7.6 Висновки за результатами огляду досліджень

Проведений огляд наукових досліджень показує, що автоматизація сервіс-деск систем потребує комплексного підходу, який поєднує різні методи аналізу

тексту та прийняття рішень. Класичні ML-моделі забезпечують швидкодію та стабільність, але мають обмеження у складних випадках. Контекстні мовні моделі та великі мовні моделі демонструють високий потенціал, однак потребують механізмів контролю та triage.

Таким чином, поєднання машинного навчання, великих мовних моделей, механізмів triage та активного навчання є перспективним напрямом розвитку сервіс-деск систем, що безпосередньо узгоджується з тематикою та цілями даної роботи [14], [15].

На основі проведеного аналізу виявлено такі невирішені проблеми, на розв'язання яких спрямована дана робота:

1. Відсутність формалізованого механізму узгодження рішень ML та LLM. Існуючі дослідження розглядають ML-класифікатори та великі мовні моделі як окремі компоненти, проте не пропонують математично обґрунтованої моделі ансамблевого прийняття рішень із урахуванням рівнів впевненості обох систем. У даній роботі розроблено модель зваженого голосування з пороговими значеннями τ_{high} та τ_{low}для динамічного вибору джерела рішення.
2. Недостатня увага до управління невизначеністю при автоматичній класифікації. Більшість підходів орієнтовані на максимізацію точності без механізмів виявлення та обробки випадків низької впевненості моделей. У даній роботі запропоновано контрольований механізм triage, який автоматично ідентифікує критичні випадки та направляє їх на ручну обробку.
3. Обмеженість існуючих методів призначення виконавців. Традиційні системи використовують статичні правила маршрутизації або однофакторний розподіл навантаження. У даній роботі розроблено модель Smart Assignment з 5-факторним скорингом, що враховує спеціалізацію, доступність, продуктивність виконавця та ймовірнісну оцінку ML-моделі.
4. Відсутність механізмів адаптації моделей до змін у потоці інцидентів. Існуючі рішення потребують повного перенавчання при появі нових категорій або зміні характеру звернень.

2 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОТОТИПУ СЕРВІС-ДЕСК СИСТЕМИ

У даному розділі наведено опис реалізації програмного прототипу сервіс-деск системи, що забезпечує автоматизовану класифікацію (category) та пріоритизацію (priority) інцидентів, залучення великої мовної моделі як другого експертного агента за умов низької впевненості ML, механізм triage для контролю невизначених випадків, а також автоматизоване призначення виконавців за алгоритмом Smart Assignment. Архітектурні рішення прототипу узгоджуються з сучасними практиками ITSM та підходами до керованої автоматизації human-in-the-loop [1; 15].

Зеленим на Рисунок 2.1 позначено інтелектуальні модулі методики (формування векторних представлень, ML-класифікація, аналіз локальною LLM, ансамблеве узгодження та Smart Assignment), а сині блоки відображають процесні та інтеграційні кроки роботи сервіс-деск (реєстрація інциденту, попередня обробка, оновлення тікета та відображення результатів). Етап “збір фідбеку та перенавчання” реалізується ітераційно (періодично), що дозволяє адаптувати моделі на основі накопичених прикладів.

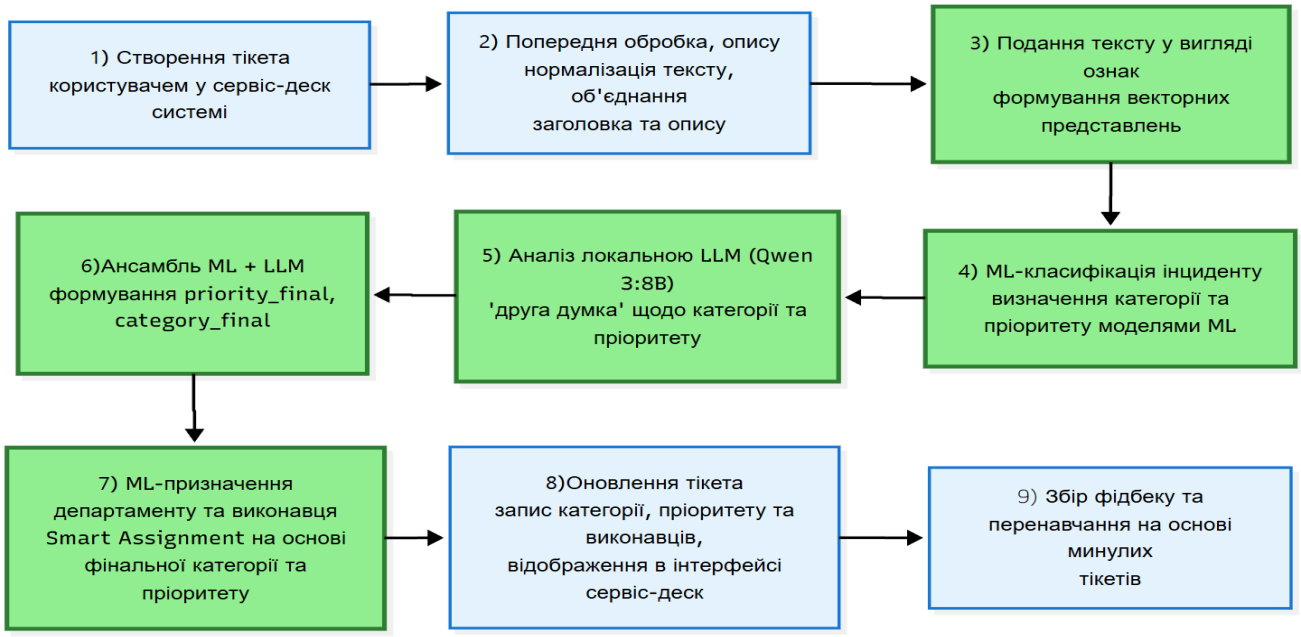


Рис. 2.1 Узагальнена схема етапів методики HYBRID v2.0

2.1 Архітектура програмного прототипу та технологічний стек

Прототип реалізовано у вигляді клієнт-серверної системи, де серверна частина представляє монолітний бекенд, а користувацький інтерфейс реалізовано як набір статичних HTML-сторінок із JavaScript. Серверна частина розроблена на Python 3.11+ з використанням фреймворку FastAPI. Доступ до даних забезпечується через SQLite-базу даних із застосуванням ORM SQLAlchemy. Така конфігурація є достатньою для локального прототипування та відтворюваних експериментів.

FastAPI надає REST/JSON API для операцій над інцидентами\тікетами, а також інтерфейс документації Swagger для верифікації доступних endpoint'ів. На рівні фронтенду статичні сторінки розміщуються у каталозі /frontend і взаємодіють із бекендом через HTTP-запити. Реалізовано створення, перегляд, редагування та отримання окремого тікета, що дозволяє інтегрувати модулі автоматизованої обробки безпосередньо в потік роботи користувача.

Таблиця 2.1

Ключові технології та їх роль у прототипі

Компонент	Технології	Призначення
Backend	FastAPI (Python 3.11+)	REST API, бізнес-логіка обробки інцидентів
Frontend	HTML + JavaScript	Веб-інтерфейс для тікетів та аналітики
База даних	SQLite + SQLAlchemy ORM	Збереження тікетів, прогнозів, метаданих моделей
ML	Sentence-Transformers embeddings + SGDClassifier	Автоматична класифікація category/priority
LLM	Локальна модель Qwen3:8b через Ollama	Другий експертний прогноз для складних випадків

2.1.1 Сценарна модель взаємодії компонентів (UML Sequence)

Для опису поведінки сервіс-деск системи недостатньо лише статичної архітектури, оскільки ключові властивості рішення проявляються у динаміці: який компонент ініціює обробку, у якій послідовності виконуються етапи класифікації та пріоритизації, де приймається фінальне рішення, що саме зберігається в базі даних і які події фіксуються для подальшого навчання. Тому для документування взаємодії компонентів у межах типового сценарію обробки інциденту використано UML-діаграму послідовностей, яка наочно відображає обмін повідомленнями між учасниками та відповідальність кожного модуля в ланцюжку обробки [41].

У роботі діаграму послідовностей подано у двох частинах, що дозволяє підвищити читабельність і чітко розмежувати рівні відповідальності. Перша частина описує зовнішній контур взаємодії користувача з системою: дії у клієнтському інтерфейсі, виклик REST API, первинну валідацію даних, створення запису тікета та формування відповіді клієнту. Друга частина відображає внутрішній контур інтелектуальної обробки, де після реєстрації звернення запускається пайплайн аналізу тексту, формується прогноз категорії та пріоритету, виконується узгодження рішень між компонентами, застосовуються triage-правила для керованої перевірки невизначених випадків і, за потреби, здійснюється автоматизоване призначення виконавця засобами Smart Assignment.

Учасниками сценарію на діаграмі виступають: користувач (ініціатор звернення), клієнтська частина (Web UI), серверна частина (REST API), база даних, модуль попередньої обробки тексту, модуль побудови контекстних представлень (ембеддингів), ML-класифікатор, LLM-компонент, модуль ансамблевого узгодження та triage, а також модуль Smart Assignment і журналювання результатів. Такий склад учасників відображає реальну логіку прототипу: частина операцій орієнтована на обслуговування користувацького сценарію (створення/оновлення тікета), інша — на формування інтелектуального рішення та накопичення даних для подальшого вдосконалення.

У межах зовнішнього контуру сценарій починається зі створення тікета: користувач заповнює форму, клієнтська частина надсилає запит до REST API, сервер виконує перевірку коректності вхідних даних і створює запис у базі даних. Після успішного збереження система ініціює внутрішню обробку тікета та повертає клієнту підтвердження реєстрації звернення. Важливо, що на цьому етапі фіксується мінімально необхідний набір полів (ідентифікатор тікета, текст звернення, службові атрибути), який надалі використовується для запуску класифікаційного контуру та для відстеження результатів у журналі прогнозів.

Внутрішній контур починається з підготовки тексту тікета: нормалізації та формування вхідного представлення для моделі. Далі обчислюються контекстні ембеддинги, на основі яких ML-класифікатор формує прогноз категорії та пріоритету разом із оцінкою впевненості. Паралельно або умовно (залежно від правил методики) може формуватися відповідь LLM-компонента, яка виступає додатковим джерелом сигналу для складних або неоднозначних звернень. На наступному кроці модуль ансамблевого узгодження порівнює прогнози, застосовує порогові правила впевненості та визначає один із двох результатів: або фінальне рішення може бути прийняте автоматично, або тікет переводиться у режим triage для керованої ручної перевірки. Якщо triage не потрібен, запускається Smart Assignment: кандидати на призначення ранжуються за багатофакторним скорингом (компетенції, команда, доступність, додаткові сигнали методики), після чого обирається виконавець і оновлюється стан тікета. На завершення внутрішнього контуру результати класифікації, підстави triage або призначення, а також службові атрибути виконання фіксуються у базі даних і журналі прогнозів, що забезпечує відтворюваність рішень та можливість подальшого аналізу й донавчання моделей.

Таким чином, UML Sequence-діаграма дозволяє формально описати наскрізний сценарій обробки інциденту та показати, як у межах єдиного контуру інтегруються ML-компонент, LLM-компонент, triage-контроль і Smart Assignment. Це, у свою чергу, спрощує верифікацію логіки прототипу, забезпечує прозорість взаємодії модулів і створює основу для подальшого розширення методики без порушення узгодженості архітектури.

2.1.2 Потік взаємодії користувача (ч.1): створення тікета та ініціація обробки

Сценарій починається з введення користувачем даних інциденту у веб-інтерфейсі - заголовок та опис. Після клієнтської валідації фронтенд формує запит на створення тікета через REST endpoint POST /tickets. Контракт запиту передбачає передачу основних полів інциденту, а також параметрів керування виконанням (зокрема можливість вимкнення LLM-обробки на рівні запиту через skip_llm). Наявність такого параметра забезпечує керованість експлуатаційних режимів та відтворюваність порівняльних експериментів у режимах “ML-only” та “ML+LLM”

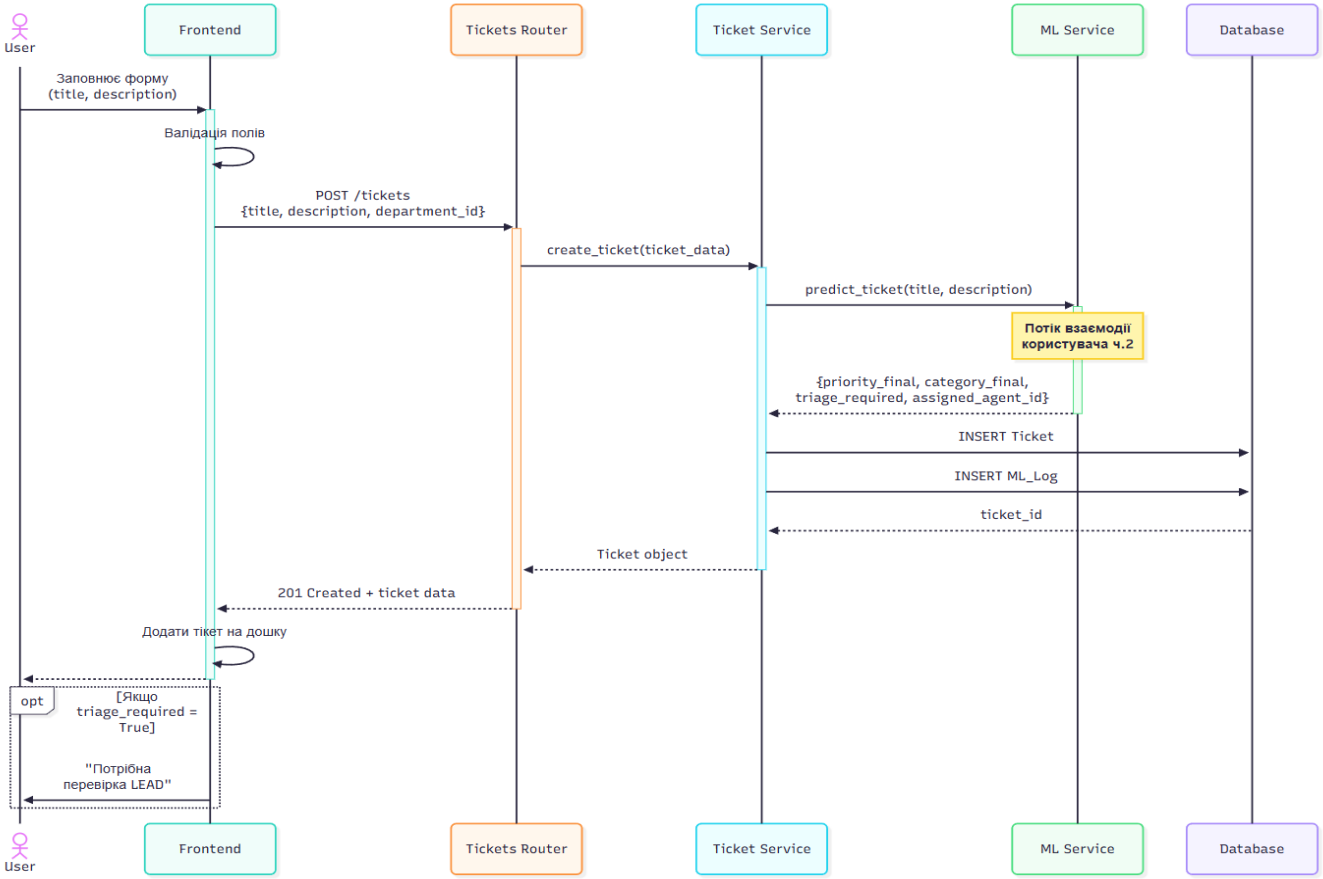


Рис. 2.2 Потік взаємодії користувача (ч.1): створення тікета та ініціація автоматизованої обробки

Після надходження запиту на бекенд модуль маршрутизації запитів Tickets Router делегує виконання бізнес-логіки сервісному шару Ticket Service. Ticket Service створює запис тікета в БД та ініціює виклик ML Service для отримання

первинного рішення щодо категорії, пріоритету та необхідності triage. Далі результати зберігаються у базі даних у двох площинах:

— у записі тікета (стан обробки, фінальні значення класифікації/пріоритизації, службові ознаки);
— у журналі прогнозів (ML/LLM-лог), що забезпечує простежуваність та можливість подальшого аналізу рішень.

Після завершення операції створення бекенд повертає 201 Created та об’єкт тікета, який містить основні атрибути інциденту і результати первинної обробки. Стан тікета задається значеннями NEW, TRIAGE, IN_PROGRESS, RESOLVED, CLOSED . Якщо встановлено ознаку triage, інтерфейс відображає необхідність експертної перевірки рішення роллю LEAD, після чого інцидент переходить у режим ручної валідації (без примусового автопризначення виконавця).

Підхід автоматичного ticket routing є практично релевантним у великих організаціях: у роботі Ackerman та співавт. описано впровадження NLP-рішення для маршрутизації понад одного мільйона тікетів на рік у межах підприємства [37]. У розробленій системі додатково закладено контрольовану гілку triage як запобіжник для випадків невизначеності, що є важливим для зменшення ризику дорогих помилок маршрутизації та пріоритизації.

2.1.3 Потік взаємодії користувача (ч.2): ML/LLM-пайплайн, ансамбль, triage та Smart Assignment

Друга частина UML-діаграми деталізує внутрішній контур інтелектуальної обробки, який запускається після створення тікета. На вході ML Service отримує текст інциденту та ініціює формування ознак через Embedding Service. Для отримання семантичного подання використовується sentence-transformers модель, що формує embedding фіксованої розмірності (384), після чого ML Models виконують прогноз категорії та прогноз пріоритету з оцінками впевненості

Паралельно (за умови активного LLM-режиму) LLM Router формує промпт і викликає локальну LLM, повертаючи альтернативні оцінки категорії та пріоритету. Далі Ensemble Service виконує узгодження результатів ML і LLM, визначає

фінальні значення та встановлює ознаку triage_required у випадках, коли автоматичне рішення потребує додаткової валідації (наприклад, при низькій впевненості або суттєвих розбіжностях між каналами). Детальні правила прийняття рішення та порогові значення узгодження наведено в підрозділі 3.3; у даному підпункті акцент зроблено саме на послідовності викликів і обміні даними між модулями.

Після узгодження, якщо triage не потрібен, запускається Smart Assignment, який виконує підбір виконавця з урахуванням категорії/пріоритету та організаційного контексту Якщо triage активований, автоматичне призначення виконавця обмежується до моменту підтвердження рішення експертом.

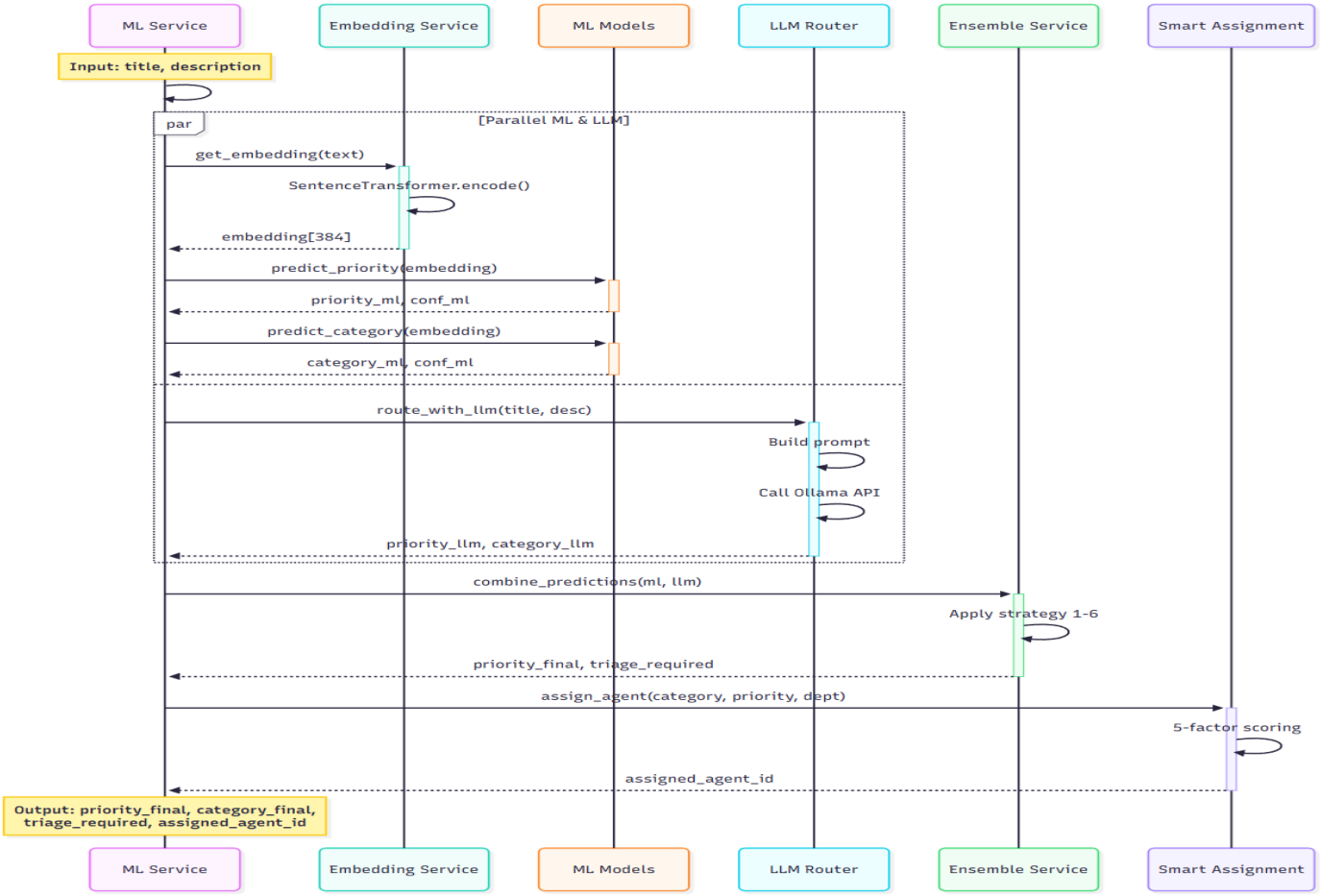


Рис. 2.3 Потік взаємодії користувача (ч.2): ML/LLM-пайплайн, ансамбль, triage та Smart Assignment

Необхідність поєднання інтелектуальних компонентів із контрольними механізмами підтверджується сучасними дослідженнями. Зокрема, огляд з

автоматизації обробки тікетів підкреслює різноманітність підходів та роль коректної організації пайплайна (представлення тексту, класифікація, багаторівневі сценарії) [38]. Також існують підходи, де призначення виконавця вирішується як end-to-end задача на трансформерах; прикладом є TaDaa, яка позиціонується як deep learning auto-advisor для призначення групи та резолвера [39]. У межах даної роботи застосовано декомпозицію на прогноз категорії/пріоритету та окремий блок призначення виконавця, що спрощує контроль якості, аудит і адаптацію в умовах обмежених даних.

Окремий напрям досліджень — LLM-орієнтоване triage на основі multi-agent/LLM-агентів. Робота Triangle описує end-to-end систему triage на основі Multi-LLM-Agent framework [40]. У запропонованому рішенні використано більш керований підхід: LLM застосовується як додатковий канал сигналу, а остаточне рішення формується ансамблевим узгодженням із можливістю переходу до triage та із журналюванням прийнятого рішення.

2.1.4 Порівняльне обґрунтування сценарного потоку та висновки

Запропонована сценарна організація потоку забезпечує керовану автоматизацію процесів Service Desk і має низку переваг у порівнянні з поширеними альтернативами.

1. Порівняння з “лише ML” (одноканальна класифікація). Одноканальна класифікація часто є швидкою та придатною для CPU-only середовищ, однак у практиці Service Desk виникають випадки семантичної неоднозначності (нетипові формулювання, брак контексту, змішані мови). Узагальнюючі огляди ticket automation підкреслюють, що результати суттєво залежать від представлення тексту та організації багаторівневих сценаріїв [38]. Додавання альтернативного каналу (LLM) та етапу узгодження підвищує стійкість рішення до нетипових формулювань, а triage забезпечує контроль у випадках невизначеності.
2. Порівняння з end-to-end deep assignment (одна модель “одразу призначає”). Існують підходи, орієнтовані на призначення групи/виконавця як єдину deep

learning задачу; прикладом є TaDaa, що використовує трансформери для рекомендації груп і резолверів [39]. У межах даної роботи застосовано декомпозицію задачі: спочатку формуються фінальні рішення щодо категорії та пріоритету, а призначення виконавця виконується окремим сервісом із багатофакторним скорингом і порогами прийнятності. Це спрощує валідацію, аудит та кероване вдосконалення компонентів незалежно.

3. Порівняння з LLM-agent triage підходами**.** Дослідження на кшталт Triangle демонструють потенціал multi-agent/LLM triage [40]. Для дипломної реалізації важливими є відтворюваність і контрольованість: саме тому в сценарному потоці передбачено формалізовані умови переходу до triage та структуроване логування результатів, що дозволяє пояснити кожне рішення й використовувати накопичені дані для подальшого вдосконалення.

Отже, двочастинна UML-діаграма послідовності фіксує наскрізний потік від створення інциденту до формування фінального рішення щодо категорії, пріоритету, ознаки triage та, за умов достатньої надійності рішення, автоматизованого призначення виконавця.

2.2 Модель даних та API для роботи з інцидентами

Базовою сутністю прототипу є тікет інциденту, який містить заголовок, опис та службові атрибути стану обробки. Після створення тікета виконується автоматизована класифікація категорії та пріоритету, а також (за умовами triage) може запускатися LLM для формування альтернативного прогнозу. Результати автоматизованої обробки зберігаються у полях тікета, що забезпечує прозорість рішення й можливість подальшого аналізу.

Для роботи з інцидентами реалізовано endpoint’и: створення тікета (POST /tickets), отримання списку (GET /tickets), отримання конкретного тікета (GET /tickets/{id}) та редагування (PUT /tickets/{id}). Таким чином, API покриває основні операції життєвого циклу інциденту і може використовуватись як фронтендом, так і зовнішніми клієнтами у тестовому середовищі.

Таблиця 2.2

Основні поля таблиці tickets та їх призначення

Група полів	Приклади полів	Призначення
Ідентифікаційні та описові	id, title, description	Зміст інциденту та ідентифікатор
Класифікація	category, priority	Результат автоматизованої класифікації
Стан обробки	status, department_id, assigned_to	Стан та маршрутизація інциденту
Часові мітки	created_at, updated_at	Відтворюваність та аудит
ML-оцінки	ml_category_confidence, ml_priority_confidence	Впевненість ML-класифікатора
LLM-оцінки	llm_category, llm_priority, llm_confidence	Альтернативний прогноз LLM
Ансамбль/triage	ensemble_strategy, needs_triage	Стратегія узгодження та прапорець triage

Для аудиту рішень та подальшого аналізу використовується таблиця ml_logs, у якій фіксуються результати кожного прогнозу з часовою міткою, значеннями confidence та інформацією про залучення LLM. Крім того, реалізовано версіонування моделей у таблиці ml_model_metadata, яка містить версію, timestamp та метрики якості. Такий підхід підвищує відтворюваність та керованість еволюції моделей [11; 15].

2.3 Реалізація ML/LLM-пайплайну та механізму triage

Для автоматизованої класифікації використовується підхід із «замороженими» ембеддингами на основі sentence-transformers моделі paraphrase-multilingual-MiniLM-L12-v2. Це дозволяє отримувати семантичне представлення тексту інциденту без тонкого донавчання трансформера у межах прототипу. Використання BERT-подібних моделей для формування подань тексту є поширеною практикою у сучасних NLP-системах [8; 9].

Поверх ембеддингів застосовано лінійні класифікатори SGDClassifier для двох окремих задач: визначення категорії та визначення пріоритету. Моделі зберігаються у файлах category_model.pkl та priority_model.pkl. Для кожного прогнозу обчислюється confidence, який записується у поля ml_category_confidence та ml_priority_confidence.

У прототипі альтернативний прогноз формується локальною великою мовною моделлю через Ollama модель Qwen3:8b паралельно з ML-класифікатором за умови, що skip_llm не активований.

Поріг високої впевненості τ_{high}= 0,75 використовується не як умова "виклику LLM", а як критерій для стратегій ансамблевого узгодження (домінування більш надійного прогнозу та контроль ризику помилки). Далі застосовується ансамблева логіка узгодження рішень, яка формує фінальний прогноз (з урахуванням можливих розбіжностей між ML та LLM) та ознаку потреби ручної перевірки (triage). Узгодження результатів кількох моделей відповідає принципам ансамблевих методів [10].

Для контролю невизначених випадків реалізовано механізм triage. У прототипі використовується поріг TRIAGE_THRESHOLD = 0,60: якщо фінальна впевненість нижча за цей поріг, встановлюється прапорець needs_triage = True, що означає необхідність ручної обробки. Крім того, у випадку суттєвого конфлікту між ML та LLM (різниця прогнозів за confidence більше 0,3) фінальна впевненість знижується, що збільшує ймовірність направлення інциденту на triage.

Такий підхід відповідає human-in-the-loop практикам [15].

Таблиця 2.3

Правила залучення LLM та triage у прототипі

Умова	Дія	Результат у даних
Одна модель високовпевнена (≥ 0,75), інша низьковпевнена (< 0,50)	Прийняти прогноз більш впевненої моделі	ensemble_strategy (відповідна стратегія), needs_triage = False
Обидві моделі високовпевнені (≥0,75) і прогноз різний	Вибір більш критичного пріоритету + обов'язковий triage	ensemble_strategy, needs_triage = True, triage_reason
skip_llm = False	Виклик LLM для альтернативного прогнозу (паралельно з ML)	llm_category / llm_priority, category_confidence / priority_confidence
Обидві моделі низьковпевнені (<0,50) або "близький матч" у weighted voting ($\lvert score_{ml} - score_{llm} \rvert$ < 0,1)	Направлення на ручну обробку triage	needs_triage = True, triage_reason, ensemble_strategy

2.3.1 Формалізація ансамблевого прийняття рішень ML + LLM

У прототипі сервіс-деск системи рішення щодо фінальної категорії та пріоритету інциденту формується на основі комбінації результатів ML-класифікатора та великої мовної моделі (LLM). На практиці LLM викликається паралельно з ML (за умови skip_llm = False) для отримання альтернативного прогнозу, після чого застосовується набір правил узгодження (ensemble_strategy) та визначається прапорець needs_triage.

Нехай x — текст інциденту, а Y — множина можливих класів (для категорії або пріоритету). ML-модель формує прогноз $y_{(ML)}$ та оцінку впевненості $conf_{ml} \in [0;1]$, а LLM формує прогноз $y_{(LLM)}$ та оцінку впевненості $conf_{llm} \in [0;1]$ (для LLM вона обчислюється окремо, евристично, на основі ознак тексту).

Формально рівень високої впевненості у прототипі визначається порогом: $\tau_{high} = 0{,}75$

Рівень низької впевненості для окремих сценаріїв визначається порогом: $\tau_{low} = 0{,}50$

Для випадків розбіжності прогнозів між ML та LLM в прототипі реалізовано зважене голосування (winner-takes-all), де кожна модель отримує фіксований ваговий коефіцієнт. Для поточної реалізації використовуються: α = 0,55 для ML та (1−α) = 0,45 для LLM.

Зважені бали для порівняння прогнозів визначаються як:

$score_{ml} = \text{conf}_{\text{ml}} \times \alpha$

$score_{llm} = \text{conf}_{\text{llm}} \times (1 - \alpha)$

Фінальний клас визначається як прогноз моделі з більшим зваженим балом:

$$yfinal = arg\ max_y\ p_{ens}(y \mid x) \quad (2.1)$$

Фінальна оцінка впевненості у прототипі, як правило, дорівнює впевненості «переможця» (conf_{ml}) або conf_{llm}), а не зваженій сумі score. Це відповідає реалізації, у якій зважування використовується для вибору джерела рішення, а не для побудови повного розподілу ймовірностей за всіма класами.

Активація triage у реалізації визначається правилом/стратегією. Зокрема, triage може активуватися у таких типових випадках: (1) обидві моделі погоджуються, але середня/комбінована впевненість є недостатньою (нижче τ_{high}); (2) обидві моделі дуже впевнені ($\geq \tau_{high}$), але не погоджуються між собою (критичний конфлікт); (3) обидві моделі мають низьку впевненість (нижче τ_{low}); (4) для зваженого голосування — «близький матч» між $score_{ml}$ − та $score_{llm}$ (наприклад, $|score_{ml} - score_{llm} < 0{,}1$).

Умова близького матчу: $|score_{ml} - score_{llm}| < 0{,}1$

Таким чином , формалізація відображає саме реалізований підхід: паралельне отримання двох прогнозів, правила узгодження з фіксованими вагами (α = 0,55) у сценаріях розбіжності, та керований механізм triage, який підключається при низькій надійності або конфлікті рішень.

2.3.2 Стратегії ансамблевого прийняття рішень

У прототипі реалізовано набір стратегій узгодження результатів ML та LLM (ensemble_strategy). Стратегія обирається на основі співпадіння/розбіжності прогнозів, рівнів $conf_{ml}$ та $conf_{llm}$ відносно порогів τ_{high}= 0,75 і τ_{low}= 0,50, а також додаткових правил для визначення needs_triage.

Нижче наведено опис стратегій у термінах, які відповідають логіці реалізації (збережено акцент на умовах спрацювання, результаті $y_{(final)}$, $conf_{(final)}$ та ознаці needs_triage).

Стратегія 1. High Confidence Agreement (обидві моделі згодні з високою впевненістю). Умова: $y_{(ML)} = y_{(LLM)}$ та $conf_{ml} \geq \tau_{high}$і $conf_{llm} \geq \tau_{high}$. Рішення: $y_{(final)} = y_{(ML)}$ (еквівалентно $y_{(LLM)}$). Фінальна впевненість визначається як combined_confidence (узгоджена оцінка). Triage: needs_triage = False.

Стратегія 2. Agreement with Moderate Confidence (згода з помірною впевненістю). Умова: $y_{(ML)} = y_{(LLM)}$, але принаймні одна з моделей має впевненість нижчу за τ_{high}. Рішення: $y_{(final)} = y_{(ML)}$. Фінальна впевненість обчислюється як середнє ($conf_{ml}$ + $conf_{llm}$)/2 або інша узгоджувальна оцінка. Triage активується, якщо combined_confidence < τ_{high}.

Стратегія 3. Disagreement – Use Highest Confidence (домінування найвпевненішої моделі). Умова: $y_{(ML)} \neq y_{(LLM)}$, при цьому одна модель має високу впевненість ($\geq \tau_{high}$), а інша — низьку ($< \tau_{low}$). Рішення: $y_{(final)}$ дорівнює прогнозу моделі з високою впевненістю (ML або LLM). Triage: needs_triage = False (за умови, що домінуюча впевненість є високою).

Стратегія 4. Weighted Voting (зважене голосування при розбіжності з помірною впевненістю). Умова: $y_{(ML)} \neq y_{(LLM)}$ та $conf_{ml} \geq \tau_{low}$і $conf_{llm} \geq \tau_{low}$. Рішення: обчислюються $score_{ml}$ = $conf_{llm}$ 0,55 та $score_{llm}$ = $conf_{llm}$ ×0,45; обирається прогноз із більшим score. Triage: активується для «близького матчу», коли $|score_{ml} - score_{llm}| < 0{,}1$.

Стратегія 5. High Confidence Disagreement (обидві дуже впевнені, але не згодні). Умова: $y_{(ML)} \neq y_{(LLM)}$ та $conf_{ml} \geq \tau_{high}$і $conf_{llm} \geq \tau_{high}$. Рішення: фінальний

клас вибирається за правилом пріоритетності (наприклад, вищий пріоритет у критичних випадках), а фінальна впевненість задається як узгоджена оцінка (наприклад, середнє). Triage: needs_triage = True (критичний конфлікт потребує ручної перевірки).

Стратегія 6. Both Very Uncertain (обидві моделі дуже невпевнені). Умова: $\text{conf}_{\text{ml}} < \tau_{low}$ та $\text{conf}_{\text{llm}} < \tau_{low}$. Рішення: обирається прогноз моделі з більшою впевненістю (max conf_{ml} , conf_{llm})). Triage: needs_triage = True (низька надійність автоматичного рішення).

Окрім наведених стратегій, у реалізації передбачені деградаційні сценарії, коли результат однієї з моделей відсутній (наприклад, LLM недоступна або виклик пропущено через skip_llm). У такому випадку рішення приймається на основі наявного прогнозу, а потреба triage визначається за порогами впевненості.

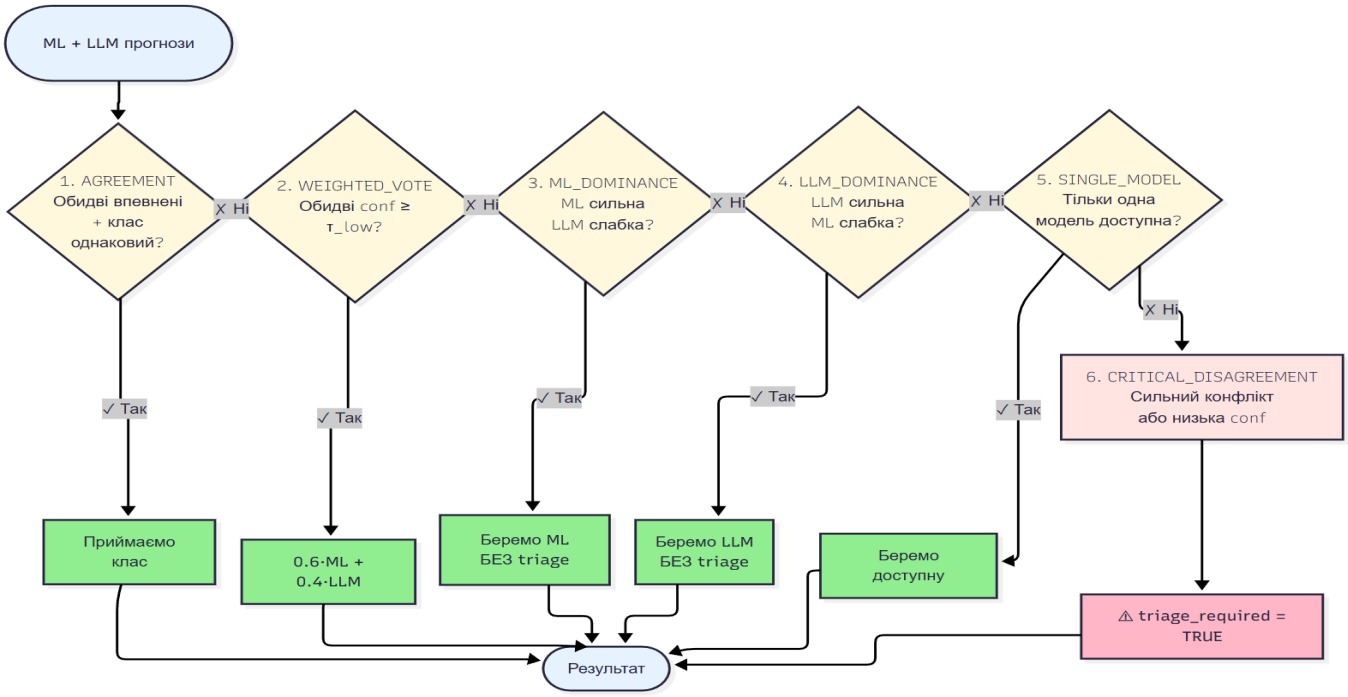


Рис. 2.4 Схема стратегії прийняття рішення

2.3.3 Структура запиту до LLM та формат відповіді

У розробленому прототипі велика мовна модель використовується як допоміжний експертний компонент, який працює паралельно з ML-класифікатором та надає альтернативний прогноз категорії та пріоритету

інциденту. Отримані результати застосовуються на етапі ансамблевого узгодження рішень, а також можуть впливати на активацію механізму triage у випадках низької фінальної впевненості.

Виклик LLM реалізовано локально через Ollama з використанням моделі Qwen3:8b у модулі llm_router_optimized.py. Такий підхід дозволяє інтегрувати LLM у конвеєр обробки інцидентів без залежності від зовнішніх сервісів.

Запит до LLM формується у вигляді структурованого інструктивного prompt англійською мовою, що обумовлено вимогами сумісності з мовною моделлю. Prompt містить текст інциденту та чіткі інструкції щодо формату вихідних даних, які повинні бути повернуті виключно у вигляді JSON-об’єкта.

Лістинг 3.1 – Узагальнена структура запиту (prompt) до LLM у прототипі

```
You are IT Service Desk routing AI. Analyze incident and return JSON:

{
  "category": "L1_TRIAGE|L1_PASSWORD|...|OTHER",
  "priority": "P1|P2|P3",
  "urgency": "HIGH|MEDIUM|LOW",
  "team": "ServiceDesk_L1|Network_Ops|...",
  "assignee": "user.id or null",
  "auto_assign": true,
  "reasoning": "brief explanation"
}

Incident text:
<title + description>
```

У відповіді LLM повертаються прогнозні значення категорії та пріоритету, а також додаткові службові поля (urgency, team, assignee, auto_assign, reasoning). Оцінка впевненості для LLM не генерується безпосередньо мовною моделлю, а обчислюється окремим евристичним модулем на основі аналізу ключових слів у тексті інциденту.

Лістинг 3.2

Приклад структурованої відповіді LLM у прототипі

```
{
  "category": "NETWORK_VPN",
  "priority": "P1",
  "urgency": "HIGH",
  "team": "Network_Ops",
  "assignee": null,
  "auto_assign": true,
  "priority_confidence": 0.95,
  "category_confidence": 0.85,
  "reasoning": "VPN connection outage detected"
}
```

Значення priority_confidence та category_confidence обчислюються евристично: для пріоритету враховується наявність термінів критичності (наприклад, production, outage, critical), а для категорії — кількість збігів ключових слів, характерних для відповідної категорії інцидентів.

Після отримання відповіді виконується валідація JSON-структури та допустимості значень. У разі помилки парсингу або невідповідності формату застосовується fallback-механізм, а результати LLM не використовуються для автоматичного прийняття рішення.

Усі результати роботи LLM (прогнозні значення та обчислені показники впевненості) зберігаються у таблиці ml_prediction_logs, що забезпечує аудитованість та можливість подальшого аналізу ефективності ансамблевого підходу.

2.4 Smart Assignment та активне навчання 4-Tier Active Learning

Після визначення категорії, пріоритету та рішення щодо потреби triage прототип виконує автоматизоване призначення виконавця (Smart Assignment), а також забезпечує адаптивне покращення моделей за рахунок активного навчання 4-Tier Active Learning. Обидва механізми реалізовано як частину серверної логіки FastAPI та інтегровано з базою даних SQLite через SQLAlchemy ORM.

2.4.1 Smart Assignment: 5-факторний скоринг виконавців

Мета механізму - обрати агента, який одночасно відповідає профілю інциденту, належить до релевантного департаменту, є доступним та має достатню історичну якість роботи. Результат Smart Assignment фіксується в полі assigned_to (ID виконавця), а також супроводжується значенням впевненості assignment_confidence та коротким поясненням assignment_reasoning.

Формалізація інтегрального скорингу:

$$S(agent \mid x) = w_1 \cdot spec + w_2 \cdot dept + w_3 \cdot avail + w_4 \cdot perf + w_5 \cdot p_{ml} \quad (2.2)$$

$$0 \le w_i \le 1, \quad w_1 + w_2 + w_3 + w_4 + w_5 = 1 \quad (2.3)$$

де x - текст інциденту та контекстні атрибути (категорія, пріоритет, департамент), а spec, dept, avail, perf, p_{ml} - нормалізовані фактори в діапазоні [0;1]. Фінальний виконавець визначається як агент з максимальним скорингом:

$$agent\_final = arg\ max\_agent\ \ S(agent \mid x) \quad (2.4)$$

Параметри $w_1 \ldots w_5$ є налаштовуваними вагами. На практиці вони підбираються так, щоб відображати пріоритетність компетенцій (spec), організаційної відповідності (dept), поточного стану агента (avail), історичної надійності (perf) та рекомендації ML-моделі призначення (p_{ml}).

Пояснення факторів скорингу (нормалізовані 0…1):

spec - збіг спеціалізації агента з інцидентом (категорія, ключові слова з опису, рівень підтримки);

dept - відповідність департаменту агента до департаменту, визначеного за категорією інциденту (Network, Workplace, Application, Security тощо);

avail - доступність агента: статус (available/busy/offline/leave) та поточне навантаження відносно workload_capacity;

perf - історична якість роботи (частка підтверджених автопризначень, успішні закриття,);

p_{ml} - ймовірність призначення агента, передбачена ML-моделлю p_assign(agent | x), навченою на базі історичних призначень.

Таблиця 2.4

Фактори Smart Assignment та приклад ваг у прототипі створених в системі

Фактор	Вага (приклад)	Інтерпретація в прототипі
spec	0,30	Keyword/категорійний збіг між specialty агента та описом інциденту
dept	0,15	Бінарна відповідність департаменту 1 -відповідний, 0 - невідповідний
avail	0,20	Композиція availability_status та workload_score активні тікети
perf	0,10	Історичний рейтинг агента assignment_score
p_{ml}	0,25	Ймовірність p_assign(agent|x) з ML-моделі призначення

Значення ваг у табл. 2.4 наведені як робочий приклад. У межах прототипу ваги можуть коригуватися для різних департаментів або залежно від критичності (P1–P3).

Правило прийняття рішення та fallback на ручну обробку:

Після обчислення S(agent | x) для всіх кандидатів система обирає agent_final з максимальним значенням. Величина assignment_confidence = S(agent_final | x) використовується як індикатор якості призначення. Якщо assignment_confidence

нижче порогового значення S_min, інцидент спрямовується на triage/ручне призначення.

Таблиця 2.5

Порівняння з схожими підходами :

Джерело	Що пропонує	Зв’язок з прототипом
Shao et al., 2008 (EasyTicket) [26]	Маршрутизацію тікетів на основі “шляху розв’язання” та історії виконання.	Підтверджує доцільність автоматичного routing/assignment на історичних даних.
Agarwal et al., 2020 [26]	End-to-end підхід до автоматичного призначення helpdesk-звернень (AI lifecycle).	Концептуально близько до Smart Assignment: прогноз виконавця + контроль якості/пояснюваність.
Devlin et al., 2018 (BERT) [27]	Контекстні ембеддинги для текстових задач.	Обґрунтовує використання семантичних представлень (embeddings) у p_{ml} та в AL-циклі.
Wei & Zou, 2019 (EDA) [28]	Просте текстове data augmentation для класифікації.	Підтримує підхід із генерацією/розширенням навчальних тікетів для стартового навчання.

2.5 Модель даних та фіксація рішень методики

2.5.1 Призначення моделі даних у контурі Service Desk

Реалізація методики автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі передбачає зберігання двох взаємопов’язаних класів даних: (1) операційних даних про інцидент як об’єкт процесної обробки та (2) даних, що забезпечують простежуваність прийнятих рішень у контурі ML/LLM і Smart Assignment. Вимога простежуваності є практично значущою для ITSM-середовища, оскільки дозволяє відтворювати причини призначення пріоритету, маршрутизації та вибору виконавця, а також виконувати аудит змін і контроль якості прийняття рішень [50].

У контексті пріоритизації інцидентів важливо зберігати не лише значення фінального пріоритету, але й підстави для його визначення. У практичних ITSM-підходах пріоритет часто формується як функція параметрів “вплив” (impact) і “терміновість” (urgency), що підкреслює необхідність відокремлення фінального рішення від контекстних пояснень і проміжних оцінок [51], [52]. В межах запропонованої методики це реалізується через поєднання атрибутів тікета, журналювання прогнозів і фіксації експертних корекцій - LEAD feedback.

2.5.2 Логічна структура сутностей і зв’язків

На рисунку 3.5 наведено логічну модель даних, яка відображає сутності DEPARTMENT, USER, TICKET, ML_PREDICTION_LOG, TICKET_ASSIGNEE та їх зв’язки. Центральною сутністю є TICKET, яка описує інцидент як об’єкт обробки та містить дані, необхідні для класифікації, пріоритизації, triage і призначення виконавця. До таких атрибутів належать: title, description, category, priority, status, організаційна прив’язка (department_id), а також поля керування triage (triage_required, triage_reason) і поля, пов’язані з механізмом призначення (assignment_method, assignment_confidence, assignment_reasoning, assignment_alternatives, auto_assigned, assignment_confirmed).

Сутність DEPARTMENT реалізує організаційну структуру системи: користувачі та інциденти прив’язані до підрозділів, а також підтримується керівна роль наприклад, lead_user_id як відповідальний за triage/контроль підрозділу. Сутність USER описує користувачів системи та їх ролі ADMIN/LEAD/AGENT/USER , а також параметри, які використовуються при Smart Assignment: спеціалізація (specialty), обмеження пропускної спроможності (workload_capacity), індикатор історичної ефективності (assignment_score) і поточний статус доступності (availability_status). Такий набір полів є необхідним для підтримки задачі “обрати виконавця” з урахуванням доменної компетенції та експлуатаційних обмежень.

Взаємозв’язки між сутностями забезпечують повний цикл життя тікета. Поле created_by_user_id у TICKET відображає ініціатора інциденту, тоді як

assigned_to_user_id фіксує поточного виконавця. Підхід із відокремленням “ядра тікета” (TICKET) від допоміжних сутностей, що відповідають за історію та аудит, узгоджується з практиками реалізації систем тікетингу, де основна інформація про звернення зберігається окремо від журналів, історії подій і додаткових артефактів [53].

На рисунку 2.5 наведено ERD-схему, що відображає основні сутності та зв’язки, необхідні для реалізації методики, включно з фіксацією ML/LLM-прогнозів у журналі та збереженням результатів Smart Assignment.

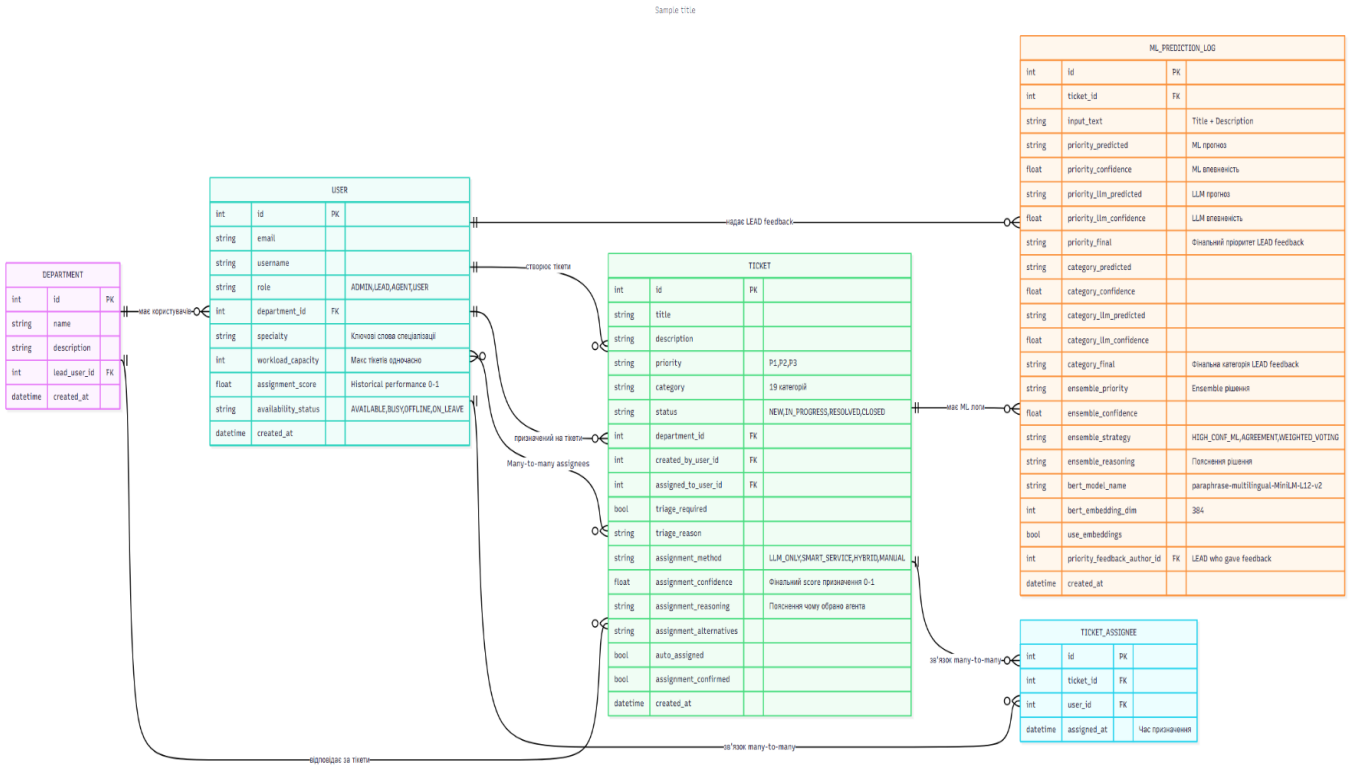


Рис. 2.5 Структура бази даних прототипу сервіс-деск системи

2.5.3 Протоколювання ML/LLM-прогнозів та фінального рішення

Для забезпечення відтворюваності результатів і можливості аналізу помилок у методиці використовується таблиця ML_PREDICTION_LOG, яка виконує роль журналу прогнозів і метаданих інференсу. У ній зберігаються: вхідний текст (input_text), результати прогнозування пріоритету та категорії з боку ML (priority_predicted, priority_confidence, category_predicted, category_confidence), результати з боку LLM (priority_llm_predicted, priority_llm_confidence, category_llm_predicted, category_llm_confidence), фінальні значення (priority_final,

category_final) та параметри ансамблювання (ensemble_priority, ensemble_confidence, ensemble_strategy, ensemble_reasoning). Окремо фіксуються метадані шару представлення тексту, що дозволяє порівнювати результати при зміні енкодера або конфігурації ознак без втрати історичної узгодженості.

Наявність такого журналу відповідає загальним практикам експеримент-трекінгу та забезпечення відтворюваності ML-результатів, де рекомендовано зберігати параметри, метрики, артефакти та контекст виконання для подальшого аналізу [54]. У межах роботи це дає можливість обґрунтовано порівнювати базові й удосконалені варіанти методики а також виконувати аналіз причин triage і випадків, де знадобилася експертна корекція.

Важливим елементом методики є підтримка експертного контролю. Для цього в ML_PREDICTION_LOG передбачено збереження інформації про автора експертної корекції (priority_feedback_author_id), що дозволяє аудитувати внесені зміни та відрізняти “автоматичні” рішення від рішень, скоригованих LEAD. Така постановка узгоджується з підходами, які підкреслюють необхідність прозорості та provenance tracking у системах із AI-компонентами як елементів управління ризиками та підвищення довіри до результатів [55]. Додатково, у роботах щодо аудитовності AI-інтегрованих систем наголошується на значенні логування та метаданих моделей для подальшої перевірки й контролю якості .

2.5.4 Зберігання результатів Smart Assignment та альтернатив

Для підтримки Smart Assignment у моделі даних використано два взаємодоповнювальні механізми.

По-перше, факт призначення (поточний відповідальний виконавець) фіксується безпосередньо в TICKET через поле assigned_to_user_id, а також супроводжується службовими полями, що пояснюють спосіб і якість призначення: assignment_method, assignment_confidence та assignment_reasoning. Це дозволяє забезпечити простежуваність логіки призначення та формувати аналітику щодо причин вибору агента.

По-друге, для зберігання альтернативних варіантів призначення використовується поле assignment_alternatives, яке містить JSON-структуру з переліком кандидатів/альтернатив і відповідними значеннями внутрішньої логіки ранжування (наприклад, топ-N кандидатів, їхні узагальнені фактори, коментарі тощо). Такий підхід дозволяє зберегти кандидатів без ускладнення схеми додатковими “кандидатними” таблицями і є зручним для експорту та пояснюваності на рівні UI/дашбордів.

Таблиця TICKET_ASSIGNEE використовується як many-to-many зв’язок між тікетом і користувачем для фіксації події призначення (зв’язок “хто і коли був призначений/зафіксований”), але не використовується для зберігання кандидатів або scoring-показників. Відповідно, в цій таблиці зберігаються лише ідентифікатори ticket_id, user_id і час призначення assigned_at. Це розділення відповідальностей між “журналом альтернатив у JSON” та “таблицею фактів призначень” дозволяє одночасно забезпечити простоту схеми та можливість аудиту прийнятих рішень.

3 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОЇ МЕТОДИКИ

3.1 Мета та завдання експериментальної перевірки

Метою експериментальної перевірки є кількісне підтвердження ефективності запропонованої методики автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі із застосуванням машинного навчання, зокрема HYBRID v2.0 є поєднання ML-класифікатора, контекстних ембеддингів, LLM-підсистеми, правил triage та механізму Smart Assignment.

Для досягнення мети розв’язано такі завдання:

— визначено протокол тестування та правила формування вибірок;

— обрано метрики якості класифікації категорії та пріоритету, а також показники автоматизації та triage; [10, 11]

— виконано порівняння з базовими підходами (TF–IDF + лінійний класифікатор; BERT/SBERT-ембеддинги + легкий класифікатор); [6, 28, 42]

— оцінено внесок 4-рівневого активного навчання в якість моделі;

— оцінено ефективність Smart Assignment за показниками точності призначення та скорочення переадресацій;

— проведено оцінку часових характеристик (латентність компонентів, поведінка при тайм-ауті LLM);

— виконано розрахункову оцінку економічного ефекту (ROI) як додатковий критерій практичної цінності.

3.2 Дані для експериментів та протокол тестування

Експериментальна перевірка виконана на сукупності інцидентів сервіс-деск системи, накопичених у базі даних та журналі прогнозів. Загальний обсяг корпусу

становить 1085 тікетів, при цьому формування підвибірок здійснювалося за схемою train/validation/test (відповідно 752/145/188 записів)

Такий розподіл дозволяє одночасно:

— навчати модель на репрезентативній частині звернень;

— контролювати перенавчання на валідації;

— отримати незалежну підсумкову оцінку на тестовій підвибірці.

Особливістю предметної області є обмеженість повністю ручної розмітки (еталонної підвибірки). У межах методики це компенсується використанням triage та активного навчання, де система фокусує ручну перевірку на найбільш “дорогих” з точки зору ризику помилки випадках - низька впевненість, розбіжність ML/LLM, нестандартні формулювання. Такий дизайн узгоджується з практикою побудови керованих людиною контурів контролю якості в інцидент-менеджменті (людина перевіряє не все, а те, що має найбільший ризик).

3.3 Метрики оцінювання

Для оцінки якості класифікації застосовано стандартні метрики багатокласової класифікації:

— Accuracy (точність) — частка правильних рішень;

— Precision/Recall/F1-score — для контролю якості по класах і для випадків дисбалансу (узагальнюється як macro або weighted). Визначення цих метрик та підхід до їхнього агрегування відповідають загальноприйнятим практикам оцінювання класифікаторів scikit-learn.sourceforge.net.

Для оцінки процесної ефективності методики (як інженерного рішення) додатково використано:

— частку автоматизованих рішень (без залучення ручної перевірки);

— частку triage та розподіл причин triage;

— часові характеристики: затримка окремих компонентів і повного циклу обробки, а також сценарій деградації при тайм-ауті LLM;

— якість Smart Assignment: точність призначення та зменшення переадресацій;
— розрахунковий економічний ефект (ROI, період окупності) як показник практичної доцільності для організації.

3.4 Порівняння базових підходів та запропонованої методики

Для забезпечення коректного порівняння всі три підходи (TF–IDF baseline, BERT/SBERT-ембеддинги та HYBRID v2.0) оцінювалися за єдиним протоколом на одному й тому самому корпусі з 1085 тікетів (19 категорій і 3 рівні пріоритету) та на однаковому розбитті train/validation/test = 752/145/188. Розбиття було зафіксоване (із фіксацією випадковості), щоб виключити вплив “вдалої/невдалої” тестової вибірки і забезпечити відтворюваність результатів. Підбір параметрів моделей та будь-які налаштування виконувалися на train/validation, тоді як тестова підвибірка використовувалася виключно для фінального вимірювання метрик.

Перед навчанням застосовано однакову базову підготовку текстів інцидентів), що гарантує: різниця в якості пояснюється саме різними підходами до представлення тексту і прийняття рішення, а не різними правилами препроцесингу. Далі кожен підхід навчався як два окремі класифікатори: для категоризації та для пріоритизації. У baseline варіанті тексти перетворювалися на TF–IDF-вектори, після чого навчався лінійний класифікатор. У варіанті з ембеддингами для кожного тікета обчислювалося контекстне векторне подання sentence embedding, і вже на цих векторах навчався “легкий” класифікатор. У HYBRID v2.0 формувалося фінальне рішення як результат поєднання сигналів (прогноз ML-ядра, контекстні ознаки та правила прийняття рішення, які закладені в методику).

Після завершення навчання кожен підхід генерував прогнози на тестовій підвибірці, тобто для кожного прикладу з test отримувався прогнозований клас, який безпосередньо порівнювався з еталонною міткою цього прикладу. Для HYBRID v2.0 в оцінюванні використовувалася саме фінальна мітка, яку повертає методика після комбінування компонентів. Окремо важливо розділяти “якість

класифікатора” і “процесний контур triage”: triage у системі виконує роль механізму контролю ризику (переводить частину випадків на ручну перевірку), але не змінює еталонні мітки в датасеті. Тому для рис. 3.1 фіксується якість саме класифікаційного рішення, а показники автоматизації та частка triage/причини triage аналізуються окремо у п. 3.6.

Метрики на рисунку. 3.1 розраховувалися стандартним способом: Accuracy як частка правильних передбачень на тесті та зважений F1-score як узагальнення якості по класах із вагами, пропорційними представленості класів у тестовій підвибірці. Зважений F1 використано додатково до Accuracy, оскільки в задачі категоризації (19 класів) розподіл звернень за класами є нерівномірним, і одна лише Accuracy може маскувати слабшу якість на рідкісних категоріях. Обчислення виконувалося типовими функціями для оцінювання класифікації на кшталт реалізацій у scikit-learn на одних і тих самих парах “еталонна мітка — прогноз моделі” для кожного з підходів.

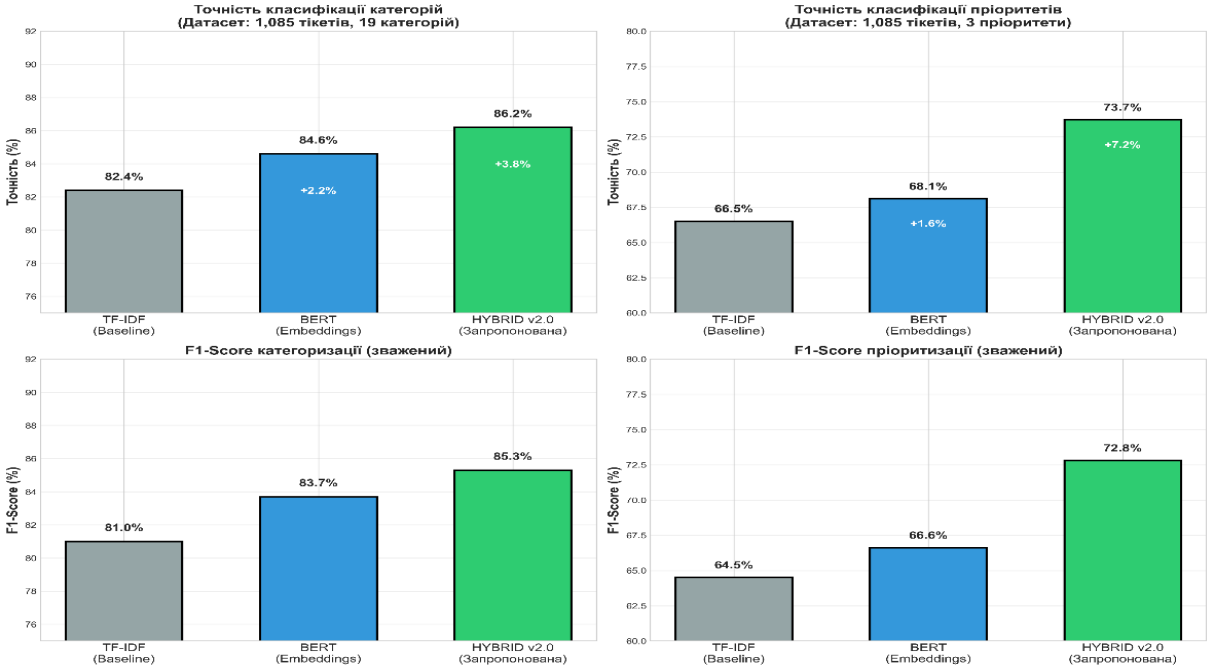


Рис3.1 Порівняння якості TF–IDF, BERT та HYBRID v2.0

Показані на діаграмі прирости “+x%” є різницею у відсоткових пунктах відносно baseline TF–IDF. Наприклад, для категоризації значення 86,2% для HYBRID v2.0 порівняно з 82,4% для TF–IDF відповідає приросту +3,8 в.п., а для пріоритизації 73,7% порівняно з 66,5% відповідає +7,2 в.п. Такий формат подачі

дозволяє інтерпретувати ефект методики як практично приріст якості за фіксованих даних.

3.5 Динаміка якості та внесок активного навчання

Окремий фокус запропонованої методики — показати, як змінюється якість моделей у процесі накопичення даних та повторного перенавчання. Для цього проведено ітераційний експеримент із активним навчанням.

Початково корпус тікетів було один раз поділено на навчальну, валідаційну та тестову підвибірки. Навчальна частина використовувалася для ітераційного перенавчання та поповнення даними в контурі активного навчання. Валідаційна застосовувалася для контролю налаштувань у процесі розробки. Тестова використовувалася виключно для незалежного вимірювання метрик після кожної ітерації. Підвибірки є взаємовиключними, а тестова частина не використовувалася для навчання, що забезпечує коректність порівняння результатів між ітераціями.

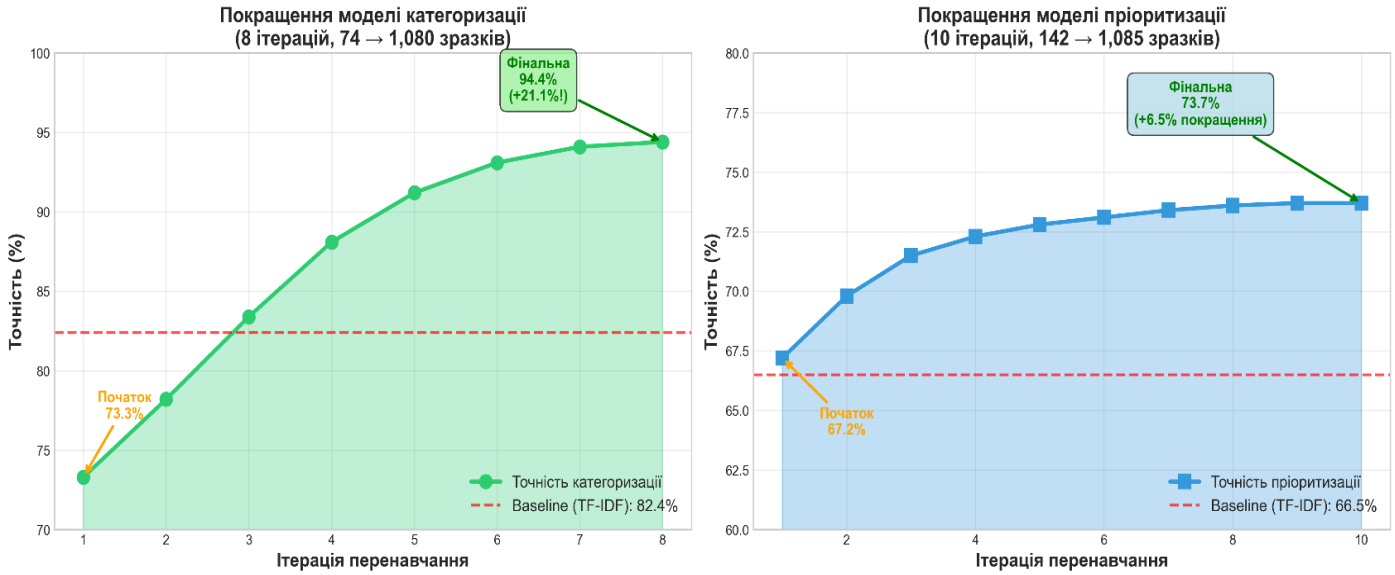


Рис 3.2 Динаміка якості моделі під час ітерацій перенавчання

Ітераційний цикл мав однакову структуру на кожному кроці. Спочатку модель навчалася на поточному навчальному наборі. Далі після завершення навчання виконувалося оцінювання: модель формувала прогнози на тестовій підвибірці, після чого точність обчислювалася як частка правильних передбачень

відносно еталонних міток. Саме ці значення точності зафіксовані на рис. 3.2 для кожної ітерації перенавчання. Після фіксації метрик частина нових прикладів відбиралася до контуру активного навчання та додавалася до навчального набору наступної ітерації, що забезпечувало поступове розширення навчальної бази.

Рис 3.3 Внесок 4-рівневого активного навчання у відбір зразків реальних тікетів

На рисунку 3.2 наведено динаміку точності для задач категоризації та пріоритизації. Для категоризації спостерігається зростання точності від стартового значення 73,3% до фінального 94,4% у процесі послідовних ітерацій перенавчання, що супроводжується збільшенням обсягу навчального корпусу. Для пріоритизації відображено зростання точності від 67,2% до 73,7% упродовж ітерацій з аналогічним нарощуванням навчального набору. Вказані на графіках прирости отримані як різниця між фінальним та стартовим значеннями точності в межах одного й того самого протоколу оцінювання.

Рисунок 3.3 пояснює, за рахунок яких типів прикладів формувався контур активного навчання. Під час відбору кожному відібраному тікету призначався рівень 1–4 відповідно до причини включення у контур активного навчання, а ця інформація фіксувалася в журналі експериментів. Загалом було відібрано 189

зразків, після чого підраховано кількість прикладів у кожному рівні та їхню частку від загального обсягу. На цій основі побудовано діаграму розподілу, яка відображає структуру відібраних прикладів за рівнями у відсотковому співвідношенні.

Для порівняння з класичним підходом використано baseline-модель TF–IDF з лінійним класифікатором. Вона оцінювалася на тій самій тестовій підвибірці та за тим самим правилом обчислення точності, а її значення нанесено на рис. 3.2 як контрольний орієнтир для зіставлення з результатами ітераційного перенавчання.

Рисунок 3.3 показує, що внесок активного навчання не зводиться до “збільшення датасету”, а полягає у пріоритизації найбільш інформативних випадків. Це критично для сервіс-деск, де значна частина звернень є шаблонною, а основні помилки зосереджені у нестандартних описах або при зміні контексту або інцидентів нового типу.

3.6 Рівень автоматизації та triage-контур

Практична ефективність методики в сервіс-деск визначається не лише середніми метриками класифікації, а тим, яку частку звернень система здатна обробити повністю автоматично без втрати керованості ризику. Саме тому окремо оцінено triage-контур як механізм “людина-в-контурі”, який спрямовує ручну перевірку не на весь потік тікетів, а на випадки з підвищеною ймовірністю помилки (низька впевненість моделі, розбіжність між компонентами, потреба уточнення еталонної мітки).

Як показано на рисунку 3.4, загальна структура датасету для експериментів становить 1085 тікетів, з яких 183 (16,9%) — реальні журнальні звернення, а 902 (83,1%) — тестові тікети, згенеровані скриптом для забезпечення достатнього покриття класів і сценаріїв. Для потоку тестових тікетів (902 шт.) triage застосовано до 200 звернень, що відповідає 22,2% від потоку, тоді як 702 звернення (77,8%) були оброблені автоматично без залучення ручної перевірки. Таким чином, triage-контур у дослідженні виступає не як “виняток”, а як контрольований інструмент

управління ризиком: система автоматизує більшість звернень, але цілеспрямовано “вилучає” з автоматизації частину кейсів із найбільшою невпевненістю . [1, 51, 52]

Аналіз причин направлення на triage рисунок. 3.4демонструє, що домінуючим чинником є ручна перевірка/уточнення мітки — 120 випадків із 200 (60%). Це важливий практичний сигнал: у реальному процесі сервіс-деск частина тлумачень категорій/пріоритетів може еволюціонувати, а також можливі неоднозначності в “еталоні”, тому контроль якості міток є критичним для стабільності моделі. Друга причина — низька впевненість моделі (50 випадків; 25%), яка відображає ситуації, де текст звернення не дає достатньо інформації або містить нетипові формулювання. Третя причина — розбіжність ML/LLM (30 випадків; 15%), тобто прикордонні кейси, де компоненти приймають різні рішення; саме такі випадки є найбільш “ризиковими” з точки зору помилкової автоматизації.

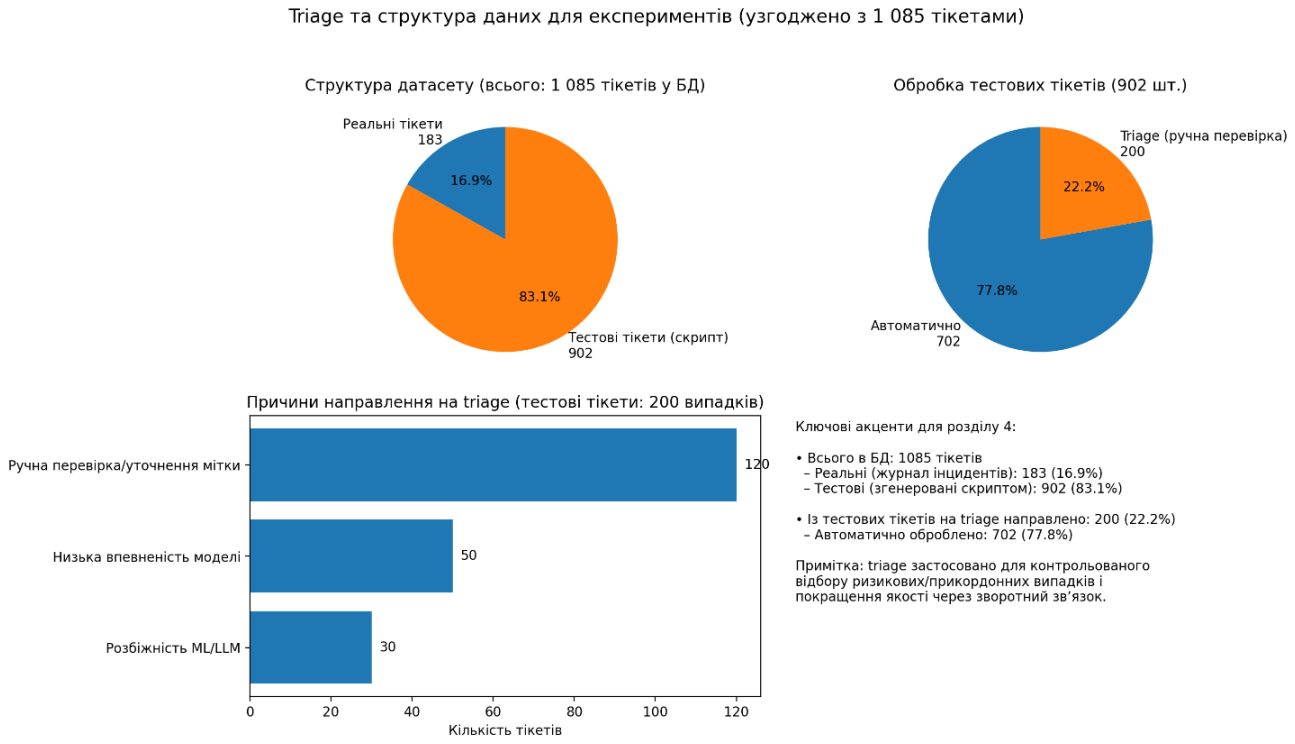


Рис. 3.4 Розподіл triage та причин перевірки.

У підсумку triage-контур інтерпретується як “запобіжник” методики: система не намагається максимізувати автоматизацію будь-якою ціною, а підтримує керований компроміс між швидкістю та якістю. Це особливо важливо для сервіс-деск, де ціна помилки для високопріоритетних інцидентів може бути

непропорційно великою; тому перенесення частини неоднозначних випадків у режим експертної перевірки є інженерно обґрунтованим підходом до забезпечення надійності процесу. [38, 39, 40]

3.7 Оцінка ефективності Smart Assignment

Наступним рівнем автоматизації після визначення категорії та пріоритету є призначення виконавця. У межах методики реалізовано Smart Assignment як механізм ранжування кандидатів, який формує підсумковий бал для кожного потенційного виконавця та обирає найкращий варіант. При цьому враховуються кілька практично значущих сигналів: відповідність спеціалізації до типу інциденту, належність до потрібного департаменту/команди, поточна доступність і завантаженість, історичні показники результативності за наявності достатніх даних, а також узгодженість призначення з прогнозом класифікатора.

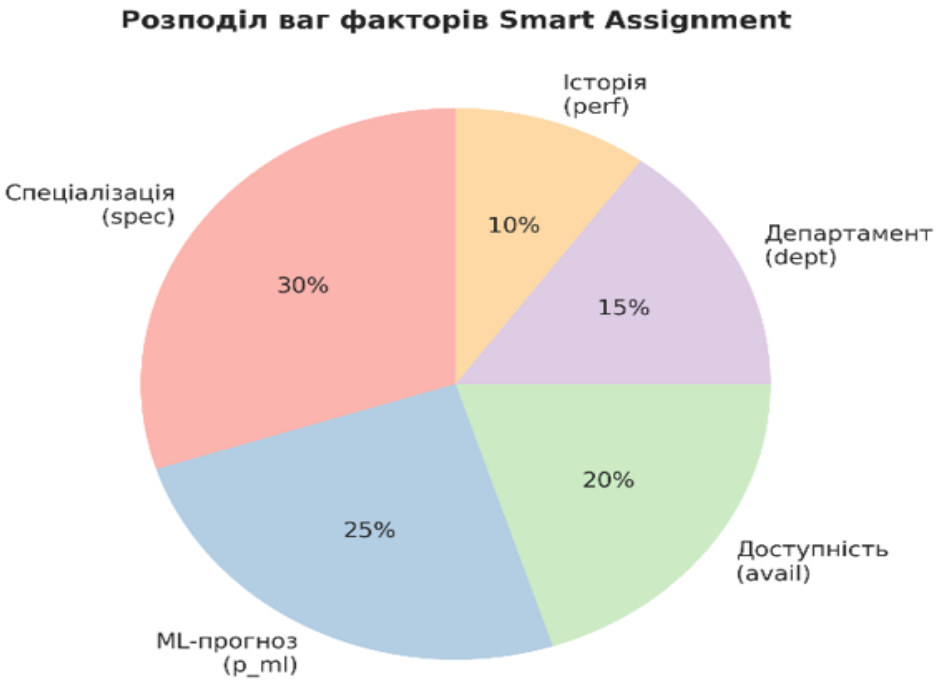


Рис. 3.5 Вагові коефіцієнти факторів Smart Assignment для обчислення підсумкового балу кандидата..

На рисунку. 3.5 наведено розподіл ваг факторів у формулі Smart Assignment (тобто налаштування, за яким система обчислює підсумковий бал кандидата). Найбільшу частку мають спеціалізація (30%) та ML-прогноз (25%), оскільки саме ці фактори найбільше впливають на коректність первинної маршрутизації: якщо інцидент правильно віднесено до типу робіт і підібрано виконавця відповідної компетенції, зменшується кількість помилкових призначень і переадресацій. Далі враховується доступність (20%), що забезпечує практичну здійсненність призначення з урахуванням поточного навантаження та статусу виконавця. Департамент (15%) використовується як організаційне обмеження, щоб призначення відповідало структурі команд та зоні відповідальності. Історія результативності (10%) має найменшу вагу, оскільки на етапі прототипу обсяг стабільних історичних даних зазвичай обмежений; відповідно цей сигнал застосовується як допоміжний, без домінування над компетентнісними та операційними факторами.

Такий розподіл ваг забезпечує керований і прозорий механізм призначення: рішення формується на основі інтерпретованих критеріїв, які можна адаптувати під правила організації та цільові показники наприклад, мінімізацію переадресацій або підвищення швидкості обробки. У подальшому, зі зростанням обсягу історичних даних, ваги можуть уточнюватися, зокрема за рахунок підсилення факторів, що відображають реальну ефективність виконавців у конкретних категоріях інцидентів.

3.8 Часові характеристики та поведінка при тайм-ауті LLM

Оскільки LLM-компонент є однією з ключових складових запропонованої методики та відповідає за підсилення якості в неоднозначних випадках, у роботі окремо оцінено часові характеристики обробки тікетів і поведінку системи за умов повільної або нестабільної відповіді LLM. Вимірювання виконувалися шляхом фіксації тривалості основних етапів обробки під час тестування з подальшим

узагальненням значень за перцентилями, що дозволяє оцінити як типовий час відповіді, так і гірші сценарії, важливі для реального сервіс-деск потоку.

Результати показують, що основні швидкодіючі компоненти методики забезпечує малу затримку, тоді як найбільшу варіативність у часі дає саме LLM-етап. Це не применшує роль LLM у методиці: навпаки, LLM використовується там, де її внесок у якість є найбільш відчутним — для складних, нетипових або прикордонних звернень, де одних лише статистичних ознак базової моделі часто недостатньо.

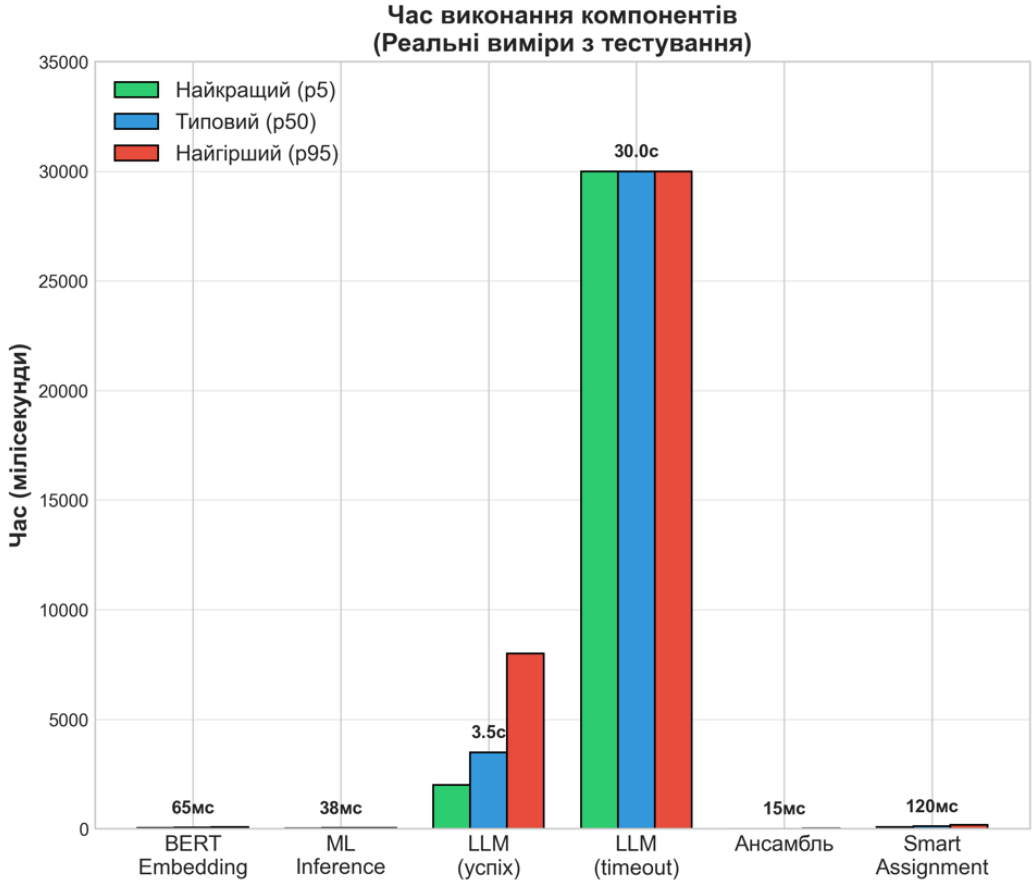


Рис 3.6 – Розподіл часу виконання етапів обробки тікета

Щоб поєднати приріст якості від LLM із вимогами до передбачуваного часу обробки, у методиці передбачено механізм тайм-ауту та сценарій деградації. Якщо LLM не повертає результат у межах заданого порогу, система не блокує обробку тікета, а формує фінальне рішення на основі доступних сигналів ML-компонентів і правил triage. Таким чином забезпечується керований компроміс “якість/час”: у штатному режимі LLM є важливою частиною рішення та підсилює точність на

складних кейсах, а в гірших часових сценаріях методика зберігає стабільність і прогнозовану тривалість обробки, не втрачаючи працездатності.

3.9 Узагальнення результатів розділу

Отримані результати підтверджують доцільність застосування запропонованої методики в задачах сервіс-деск: поєднання ML, контекстних ембеддингів, LLM-підсистеми, triage-контролю та Smart Assignment забезпечує покращення якості рішень у порівнянні з базовими підходами, а також формує керований процес автоматизації з можливістю ручної валідації ризикових випадків. На відміну від “суто ML” сценарію, запропонований підхід орієнтований на еволюцію якості в умовах обмеженої ручної розмітки та на контроль помилок у прикордонних ситуаціях, що є критичним для інцидент-менеджменту.

ВИСНОВКИ

1. Досліджено сучасні підходи до автоматизації сервіс-деск процесів, зокрема правила та експертні системи, класичні ML-моделі для текстової класифікації, підходи до маршрутизації та triage інцидентів, а також human-in-the-loop та активне навчання. На основі аналізу переваг і обмежень існуючих рішень сформовано практично орієнтовані вимоги до методики автоматизованої класифікації та пріоритизації інцидентів, що відповідає тематиці роботи.

2.Реалізовано програмний прототип сервіс-деск системи, який забезпечує повний цикл обробки звернення: реєстрацію інциденту, автоматизоване визначення категорії та пріоритету, ухвалення рішення щодо необхідності triage, а також подальшу підтримку керованої обробки складних тікетів. Для підвищення надійності прийняття рішень застосовано комбінування інструментів машинного навчання та LLM-компонента як "другої думки" з обов'язковою структурною валідацією відповіді, що знижує ризик некоректних або неповних результатів і підвищує відтворюваність роботи системи.

Окрему увагу приділено автоматизованому призначенню виконавця. У прототипі реалізовано механізм Smart Assignment на основі багатофакторного скорингу, де фінальне рішення формується як зважена сума нормалізованих факторів. На відміну від спрощених правил "за категорією", використаний підхід враховує відповідність спеціалізації агента, належність до департаменту, доступність, історичну ефективність і рекомендацію ML-моделі призначення. Додатково передбачено fallback-логіку: якщо впевненість скорингу недостатня або кандидатів немає, інцидент перенаправляється на triage/ручне призначення, що дозволяє контролювати якість у нестандартних ситуаціях.

Для підтримки адаптивного покращення якості рішень у прототипі закладено підхід активного навчання 4-Tier Active Learning із накопиченням прикладів різної надійності та можливістю донавчання з БД. Така організація зворотного зв'язку дозволяє поступово розширювати навчальну вибірку та підвищувати якість

моделей без пропорційного зростання ручної розмітки, що є критичним для сервіс-деск середовища з обмеженою кількістю якісно промаркованих інцидентів.

3. Отримані результати підтверджують працездатність запропонованих рішень у межах локального стенду та демонструють практичну доцільність поєднання ML-класифікації/пріоритизації, контрольованого triage, багатофакторного Smart Assignment і активного навчання як комплексного підходу до автоматизації сервіс-деск систем. Подальший розвиток роботи доцільно спрямувати на розширення набору реальних даних, поглиблене порівняння з альтернативними підходами на уніфікованих метриках, а також на формалізацію SLA-орієнтованих правил і підвищення якості даних.

Отримані результати свідчать про успішне досягнення поставленої мети та виконання всіх завдань дослідження. Запропонована гібридна методика HYBRID v2.0 демонструє покращення якості автоматизованої обробки інцидентів порівняно з існуючими підходами, забезпечуючи баланс між рівнем автоматизації та контролем якості критичних рішень. Розроблений програмний прототип підтверджує практичну реалізованість методики та її готовність до впровадження.

Результати дослідження апробовано та опубліковано у наступних статті та тезах:

1. Золотухіна О.А. Щербаченко О.С. Порівняльний аналіз хмарних рішень для створення ІМ-системи : Матеріали VI Всеукраїнської науково-технічна конференція « Застосування програмного забезпечення в інформаційно комунікаційних технологіях ».15.05.2025, ДУІКТ, м. Київ. Збірник тез. К.: ДУІКТ,2025.С.495-498.

2. Золотухіна О.А. Щербаченко О.С. Методологічні аспекти розробки системи Incident Management з використанням хмарних технологій : Матеріали VI Всеукраїнської науково-технічна конференція « Застосування програмного забезпечення в інформаційно комунікаційних технологіях ».15.05.2025, ДУІКТ, м. Київ. Збірник тез. К.: ДУІКТ,2025.С.532-535.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO/IEC 20000-1 Mapping Guide. BSI Group. [Електронний ресурс] – Режим доступу: https://www.bsigroup.com/globalassets/localfiles/fr-fr/isoiec-20000/ressources/isoiec_20000-1_mapping_guide_final-en.pdf (дата звернення: 28.11.2024).
2. Revina A. et al. IT Ticket Classification: The Simpler the Better. 2020. URL: https://depositonce.tu-berlin.de/items/93f30592-198d-48d3-9b05-e36bfc914f6d (дата звернення: 28.11.2024).
3. Remil O. Solutions and AI Techniques for Incident Management. arXiv, 2024. URL: https://arxiv.org/pdf/2404.01363.pdf (дата звернення: 28.11.2024).
4. Bogatinovski J. et al. A Survey of Research and Practices in AIOps. arXiv, 2021. URL: https://arxiv.org/pdf/2101.06054.pdf (дата звернення: 28.11.2024).
5. Dang Y. et al. AIOps: Real-World Challenges and Research Innovations. 2019. URL: https://orderlab.io/paper/aiops-icse19-briefing.pdf (дата звернення: 28.11.2024).
6. Salton G., Buckley C. Term-weighting approaches in automatic text retrieval. 1988. URL: https://ecommons.cornell.edu/bitstream/1813/6721/1/87-881.pdf (дата звернення: 28.11.2024).
7. Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. 1998. URL: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf (дата звернення: 28.11.2024).
8. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, 2018. URL: https://arxiv.org/pdf/1810.04805.pdf (дата звернення: 28.11.2024).
9. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv, 2019. URL: https://arxiv.org/pdf/1908.10084.pdf (дата звернення: 28.11.2024).
10. Mandal A. et al. Helpdesk Email Ticket Classification: A Deep Learning Approach. arXiv, 2018. URL: https://arxiv.org/pdf/1808.02636.pdf (дата звернення: 28.11.2024).

11. Cui Z. et al. Automated Labeling of Bugs and Tickets Using Attention-Based Mechanisms in Recurrent Neural Networks. arXiv, 2018. URL: https://arxiv.org/pdf/1807.02892.pdf (дата звернення: 28.11.2024).
12. Paramesh P. M. et al. Classifying the Unstructured IT Service Desk Tickets Using Ensemble of Classifiers. arXiv, 2021. URL: https://arxiv.org/pdf/2103.15822.pdf (дата звернення: 28.11.2024).
13. He M. et al. Ticket-BERT: A Transformer-Based Approach for Automatic Ticket Assignment. arXiv, 2023. URL: https://arxiv.org/pdf/2307.00108.pdf (дата звернення: 28.11.2024).
14. Microsoft Research. Large Language Models Can Provide Accurate and Interpretable Incident Triage (COMET). 2024. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2024/08/ISSRE24_LLM4triage.pdf (дата звернення: 28.11.2024).
15. Settles B. Active Learning Literature Survey. 2010. URL: https://burrsettles.com/pub/settles.activelearning.pdf (дата звернення: 28.11.2024).
16. Amershi S. et al. Guidelines for Human-AI Interaction. 2019. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2019/01/Guidelines-for-Human-AI-Interaction-camera-ready.pdf (дата звернення: 28.11.2024).
17. Lee D.-H. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. 2013. URL: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45594.pdf (дата звернення: 28.11.2024).
18. Ribeiro M. T., Singh S., Guestrin C. "Why Should I Trust You?" Explaining the Predictions of Any Classifier. 2016. URL: https://arxiv.org/pdf/1602.04938.pdf (дата звернення: 28.11.2024).
19. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. arXiv, 2017. URL: https://arxiv.org/pdf/1705.07874.pdf (дата звернення: 28.11.2024).
20. Guo C. et al. On Calibration of Modern Neural Networks. arXiv, 2017. URL: https://arxiv.org/pdf/1706.04599.pdf (дата звернення: 28.11.2024).
21. Sculley D. et al. Hidden Technical Debt in Machine Learning Systems. 2015. URL: https://papers.nips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf (дата звернення: 28.11.2024).

22. Bender E. M. et al. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 2021. URL: https://s10251.pcdn.co/pdf/2021_BenderEtAl_StochasticParrots.pdf (дата звернення: 28.11.2024).
23. Brown T. et al. Language Models are Few-Shot Learners. arXiv, 2020. URL: https://arxiv.org/pdf/2005.14165.pdf (дата звернення: 28.11.2024).
24. Zhao W. X. et al. A Survey of Large Language Models. arXiv, 2023. URL: https://arxiv.org/pdf/2303.18223.pdf (дата звернення: 28.11.2024).
25. Kramer S. et al. Integrating Large Language Models into Security Incident Response Workflows. USENIX, 2025. URL: https://www.usenix.org/system/files/soups2025-kramer.pdf (дата звернення: 28.11.2024)
26. Shao Q., Chen Y., Tao S., Yan X., Anvik J., Murphy G. C. EasyTicket: A Ticket Routing Recommendation Engine for Enterprise Problem Resolution. PVLDB, 2008. DOI: https://doi.org/10.14778/1454159.1454193 (дата звернення: 28.11.2024).
27. Agarwal S., Bandlamudi J., Mandal A., Ray A., Sridhara G. Automated Assignment of Helpdesk Email Tickets: An AI Lifecycle Case Study. AI Magazine, 2020. DOI: https://doi.org/10.1609/aimag.v41i3.5321 (дата звернення: 28.11.2024).
28. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv, 2018. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 28.11.2024).
29. Wei J., Zou K. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. arXiv, 2019. URL: https://arxiv.org/abs/1901.11196 (дата звернення: 28.11.2024).
30. Sharma S., Sharma B. Helpdesk Ticket Classification using Machine Learning and Deep Learning Techniques. Journal of Theoretical and Applied Information Technology, 2020. URL: https://www.jatit.org/volumes/Vol98No1/15Vol98No1.pdf (дата звернення: 28.11.2024).
31. Zangari A., Marcuzzo M., Schiavinato M., Gasparetto A., Albarelli A. A systematic literature review on automatic ticket routing / assignment in IT service management (та

суміжних доменах). Expert Systems with Applications, 2023. DOI/URL: https://doi.org/10.1016/j.eswa.2023.119984 (дата звернення: 1.12.2025).

32. Xu J., He R. Expert recommendation for trouble ticket routing. Data & Knowledge Engineering, 2018. URL: https://www.sciencedirect.com/science/article/abs/pii/S0169023X17305165 (дата звернення: 1.12.2025).
33. Han J., Sun A. DeepRouting: A Deep Neural Network Approach for Trouble Ticket Routing. (SCC/конф. матеріали), 2020. URL: https://personal.ntu.edu.sg/axsun/paper/SCC20Draft.pdf (дата звернення: 1.12.2025).
34. [Feng L., Senapati J., Liu B. TaDaa: Real Time Ticket Assignment using Deep Learning. arXiv, 2022. URL: https://arxiv.org/pdf/2207.11187.pdf (дата звернення: 1.12.2025).
35. Ackerman S., Alexander L., Bennett M., Chen D., Farchi E., House R. та ін. Deploying a ticket router across the enterprise (IBM/industry case). 2023. URL: https://research.ibm.com/blog/deploying-a-ticket-router-across-the-enterprise (дата звернення: 1.12.2025).
36. Tian Y., Wijedasa D., Lo D., Le Goues C. Learning to Rank for Bug/Ticket Recommendation (Assignee Recommendation). 2016. URL: https://www.comp.nus.edu.sg/~abhik/pdf/Materials/TianRecommendation2016.pdf (дата звернення: 1.12.2025).
37. Object Management Group. UML® — Unified Modeling Language, Version 2.5.1. 2017. [Електронний ресурс] – Режим доступу: https://www.omg.org/spec/UML/2.5.1/PDF (дата звернення: 2.12.2025). OMG
38. Yu Z. et al. Triangle: Empowering Incident Triage with Multi-Agent. Microsoft Research / ASE 2025 (PDF), 2025. [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/research/wp-content/uploads/2025/02/TRIANGLE_ASE25.pdf (дата звернення: 2.12.2025). Microsoft

39. Zhang L. et al. A Survey of AIOps for Failure Management in the Era of Large Language Models. arXiv, 2024. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/2406.11213.pdf (дата звернення: 2.12.2025). arXiv
40. Zhang L. et al. A Survey of AIOps in the Era of Large Language Models (LLM4AIOps). arXiv, 2025. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/2507.12472.pdf (дата звернення: 2.12.2025)
41. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ACL Anthology, 2019. [Електронний ресурс] – Режим доступу: https://aclanthology.org/N19-1423/ (дата звернення: 2.12.2025).
42. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. ACL Anthology, 2019. [Електронний ресурс] – Режим доступу: https://aclanthology.org/D19-1410.pdf (дата звернення: 2.12.2025).
43. Wang L. et al. Text Embeddings by Weakly-Supervised Contrastive Pre-training (E5). arXiv, 2022. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/2212.03533.pdf (дата звернення: 2.12.2025).
44. Wang W. et al. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. NeurIPS, 2020. [Електронний ресурс] – Режим доступу: https://proceedings.neurips.cc/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (дата звернення: 2.12.2025).
45. Kuratov Y., Arkhipov M. Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language. arXiv, 2019. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1905.07213.pdf (дата звернення: 2.12.2025).
46. Hugging Face. sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 (model card). [Електронний ресурс] – Режим доступу: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 (дата звернення: 2.12.2025).
47. Sentence-Transformers Documentation. Pretrained Models (Multilingual / paraphrase-multilingual-MiniLM-L12-v2). [Електронний ресурс] – Режим доступу:

https://www.sbert.net/docs/sentence_transformer/pretrained_models.html (дата звернення: 2.12.2025).

48. Hugging Face. intfloat/multilingual-e5-base (model card). [Електронний ресурс] – Режим доступу: https://huggingface.co/intfloat/multilingual-e5-base (дата звернення: 2.12.2025).
49. TensorFlow. Universal Sentence Encoder (README). [Електронний ресурс] – Режим доступу: https://github.com/tensorflow/tfjs-models/blob/master/universal-sentence-encoder/README.md (дата звернення: 2.12.2025).
50. BSI Group. ISO/IEC 20000-1 Implementation Guide (PDF). [Електронний ресурс] – Режим доступу: https://www.bsigroup.com/globalassets/localfiles/en-hk/iso-20000/isoiec-20000-1_implementation_guide-final.pdf (дата звернення: 2.12.2025).
51. Atlassian. How impact and urgency are used to calculate priority (Jira Service Management Cloud). [Електронний ресурс] – Режим доступу: https://support.atlassian.com/jira-service-management-cloud/docs/how-impact-and-urgency-are-used-to-calculate-priority/ (дата звернення: 2.12.2025).
52. Atlassian. Create an impact urgency priority matrix (Jira Service Management Cloud). [Електронний ресурс] – Режим доступу: https://support.atlassian.com/jira-service-management-cloud/docs/how-do-i-create-a-matrix-using-impact-and-urgency-values/ (дата звернення: 2.12.2025).
53. Best Practical. DatabaseAdmin – Request Tracker Wiki (overview of RT schema). [Електронний ресурс] – Режим доступу: https://rt-wiki.bestpractical.com/wiki/DatabaseAdmin (дата звернення: 2.12.2025).
54. MLflow. MLflow Tracking (logging params, code versions, metrics, artifacts). [Електронний ресурс] – Режим доступу: https://mlflow.org/docs/latest/ml/tracking/ (дата звернення: 2.12.2025).
55. NIST. Artificial Intelligence Risk Management Framework 1.0 (NIST.AI.600-1). 2024. [Електронний ресурс] – Режим доступу: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf (дата звернення: 2.12.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА

«Методика автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі із застосуванням машинного навчання»

Виконав: студент групи ПДМ-62 Олександр ЩЕРБАЧЕНКО

Керівник: канд. техн. наук, завідувач кафедри ІТ В'ячеслав Трейтяк

Київ - 2025

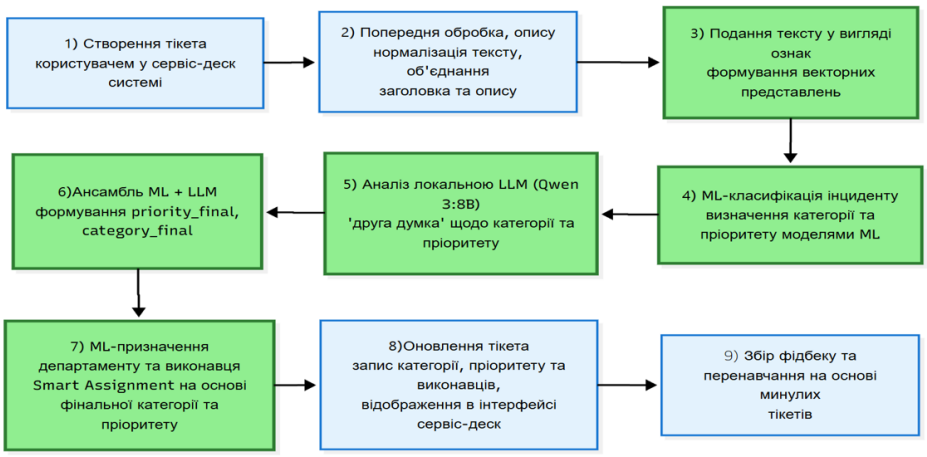
МЕТА, ОБ’ЄКТ, ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: удосконалення процесу автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі гібридної моделі з узгодженням рішень ML та LLM компонентів.

Об’єкт дослідження: процеси обробки інцидентів у сервіс-деск системі, зокрема етапи їхньої класифікації, пріоритизації та призначення виконавців.

Предмет дослідження: моделі та методи автоматизованої класифікації та пріоритизації інцидентів у сервіс-деск системі на основі аналізу історичних тікетів і зворотного зв’язку

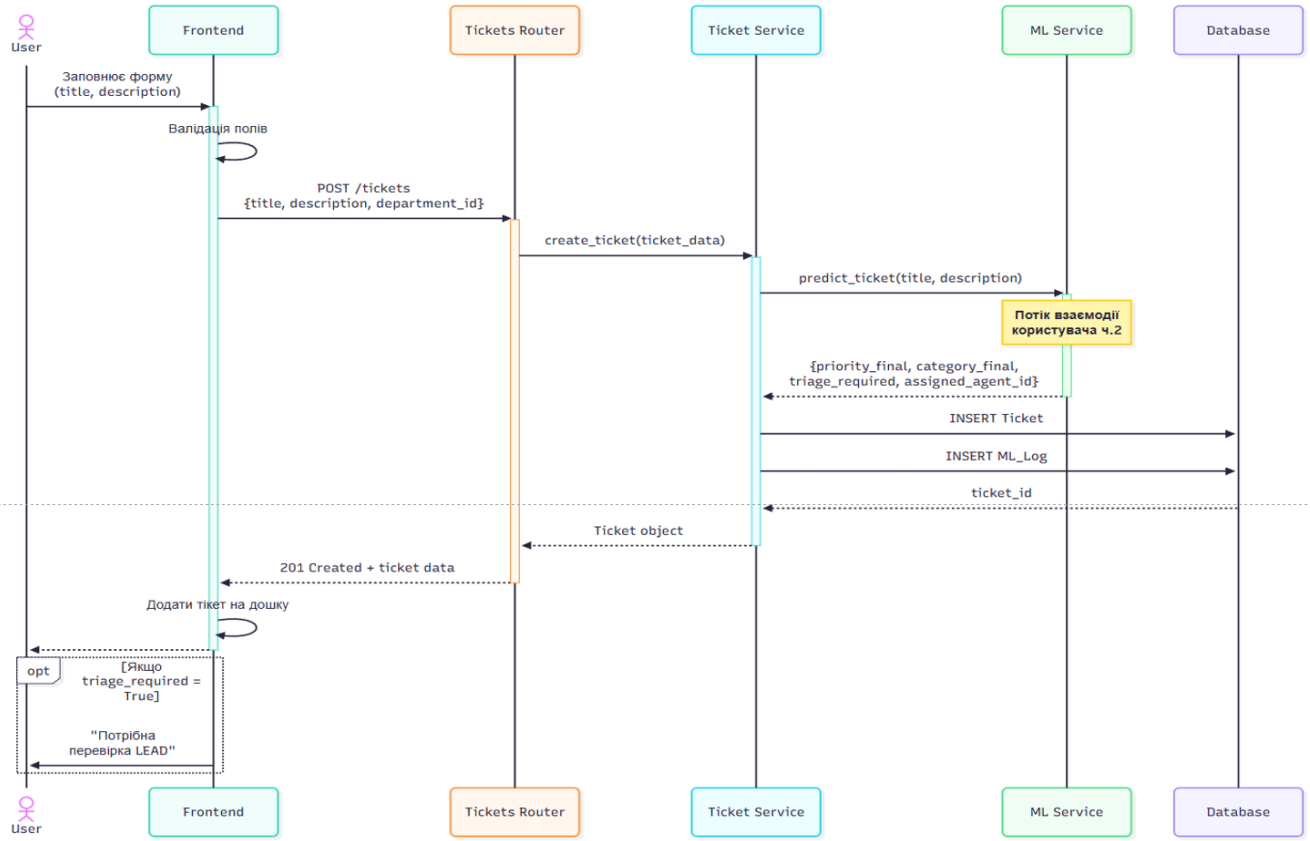
ЕТАПИ АВТОМАТИЗОВАНОЇ ОБРОБКИ ІНЦИДЕНТІВ У СЕРВІС-ДЕСК СИСТЕМІ



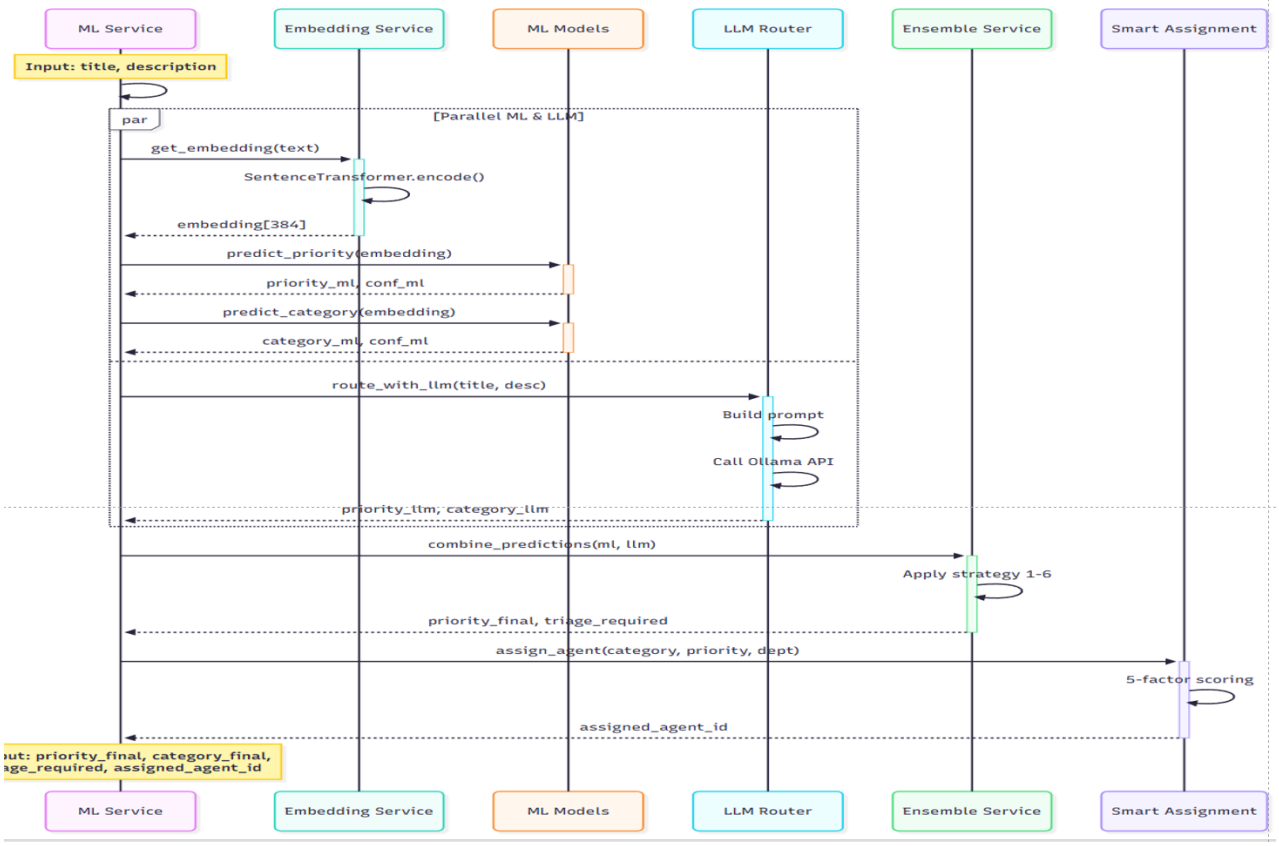
Особливості розробленої системи

•використання моделей машинного навчання для автоматизованої класифікації та пріоритизації інцидентів;
•паралельний аналіз локальною мовною моделлю Qwen 3:8B;
•прийняття остаточного рішення за допомогою ансамблю ML + LLM з можливістю відправлення інциденту на triage;
•збереження історії рішень та ручних коригувань у базі тікетів;
•адаптація моделей на основі попередніх інцидентів для поступового підвищення якості класифікації та пріоритизації.

Потік взаємодії користувача ч.1



Потік взаємодії користувача ч.2



Математична модель ансамблю ML + LLM

$$p_{ens}(y \mid x) = \alpha \cdot p_{ml}(y \mid x) + (1 - \alpha) \cdot p_{llm}(y \mid x), \quad \alpha = 0.6$$

Вхідні дані:

- $p_{ml}(y \mid x)$– ймовірності класів від ML;
- $p_{llm}(y \mid x)$,– ймовірності від LLM;
- $conf_{ml} = \max_y p_{ml}(y \mid x)$ рівень впевненості ML;
- $conf_{llm} = \max_y p_{llm}(y \mid x)$рівень впевненості LLM;
- τ_{high}– поріг високої впевненостіце , число від 0 до 1, вище якого ми вважаємо модель "досить впевненою, щоб їй довіряти без людини"
- τ_{low}– поріг низької впевненості.

$\alpha = 0{,}6$ — коефіцієнт, який каже: 60 % довіри до ML, 40 % – до LLM.

$$y_{final} = \arg\max_y p_{ens}(y \mid x)$$

y_{final} – клас, що максимізує ймовірність інциденту

Вихід ансамблю:

- priority_final – обраний пріоритет інциденту ;
- category_final – кінцева категорія інциденту;
- ensemble_strategy – яка стратегія застосована;
- ensemble_confidence – підсумкова впевненість ансамблю;
- triage_required – чи потрібна ручна перевірка ;
- triage_reason – чому потрібен triage .

Стратегії прийняття рішення:

1) **AGREEMENT** – ML і LLM дають однаковий клас, $conf_{ml} \geq \tau_{high}$, $conf_{llm} \geq \tau_{high}$ → приймаємо клас, без triage.

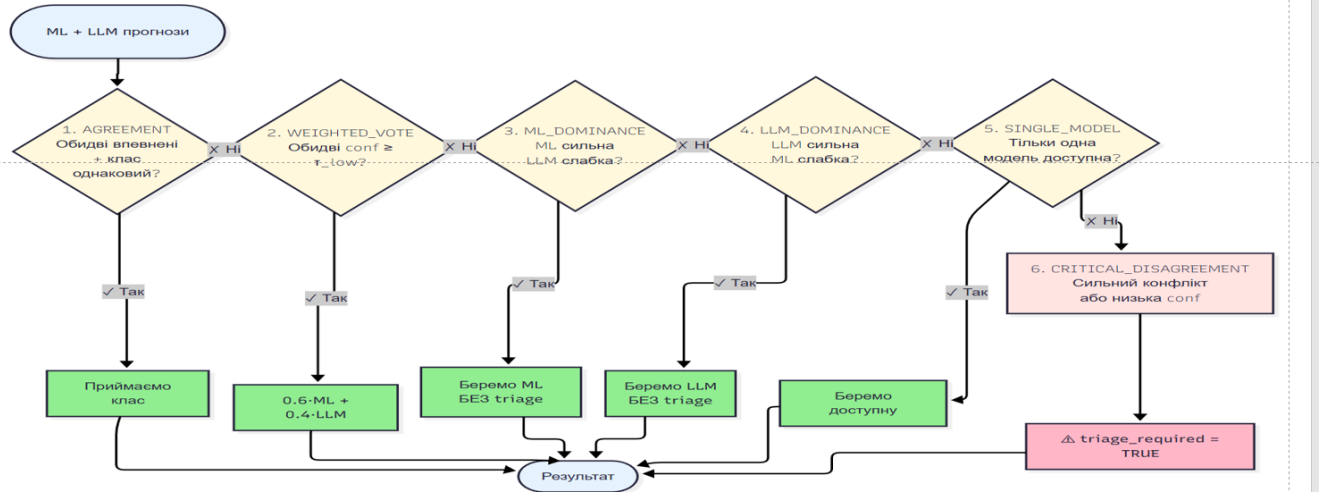
2) **WEIGHTEDVOTE**– класи різні, але обидві моделі достатньо впевнені
$(conf \geq \tau_{low}) \rightarrow p_{ens} = 0.6 \cdot p_{ml} + 0.4 \cdot p_{llm}$
$y_{final} = \text{argmax}_y\, p_{ens}(y \mid x)$

3) **MLDOMINANCE**– $conf_{ml} \geq \tau_{low}$,
LLM слабкий / відсутній → беремо прогноз ML.

4) **LLMDOMINANCE**– $conf_{llm} \geq \tau_{low}$,
ML слабкий / відсутній →тимчасово беремо прогноз LLM

5) **SINGLE_MODEL** – доступна лише одна модель → використовуємо її результат.

6) **CRITICAL_DISAGREEMENT**–
сильний конфлікт при високій впевненості
або $\max(conf_{ml}, conf_{llm}) < \tau_{low}$ →
$triage\_required$ = TRUE (ручна перевірка).



Математична модель Smart Assignment: 5-факторний скоринг виконавців

$$S(agent \mid x) = w_1 \cdot spec + w_2 \cdot dept + w_3 \cdot avail + w_4 \cdot perf + w_5 \cdot p_{ml},$$

$$0 \leq w_i \leq 1 \qquad w_1 + w_2 + w_3 + w_4 + w_5 = 1$$

Фактори скорингу (нормалізовані 0…1):

- spec – збіг спеціалізації агента з тікетом (категорія, ключові слова, рівень підтримки);
- dept – відповідність департаменту (Network, Workplace, Application, Security тощо);
- avail – доступність агента (менше активних тікетів = вищий бал);
- perf – історична якість роботи (прийняті автопризначення, успішно закриті тікети);
- p_{ml}– ймовірність призначення агента з ML-моделі $p\_assign(agent \mid x)$.
- $w_1 \ldots w_5$– частки впливу п'яти факторів $(spec, dept, avail, perf, p\_ML)$;
- w_i – налаштовувані ваги,у сумі дають 1.
- чим більший w_i ,тим сильніше цей фактор впливає на вибір виконавця.

Логіка Smart Assignment:

- Для кожного агента обчислюємо S(agent | x) за формулою.
- Обираємо агента з максимальним скорингом $S(agent \mid x)$ - підсумковий бал агента для тікета
- Якщо всі скоринги нижче порога S_min → triage_required = TRUE, виконавця обирає людина.

$$agent_{final} = \arg\max_{agent} S(agent \mid x)$$

Вихід Smart Assignment:

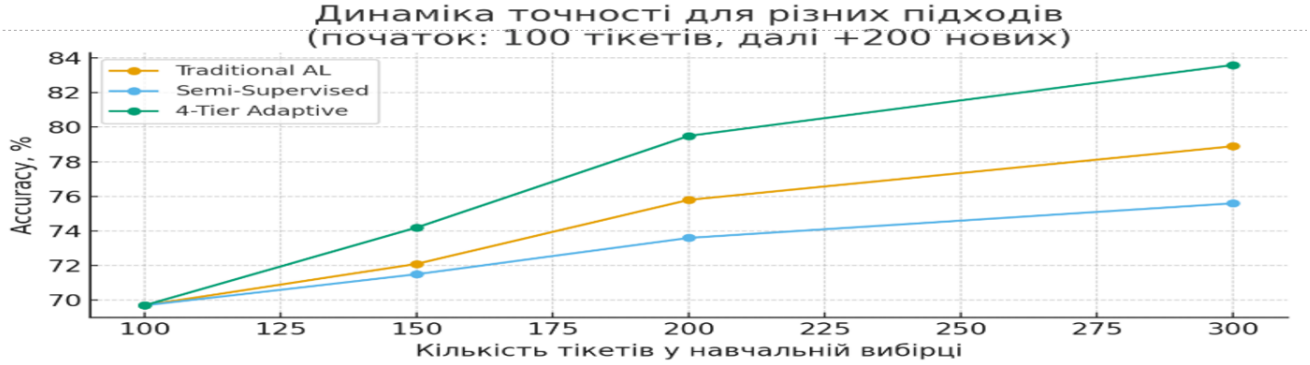
- assigned_agent_id – обраний виконавець;
- assignment_confidence – значення S(agent_final | x);
- assignment_reasoning – коротке пояснення (наприклад: "високий spec, dept і perf").

Порівняння 4-рівневої методики з існуючими підходами

Властивість / Підхід	Традиційне Active Learning Settles	Напівконтрольоване Навчання Semi-Supervised	4-рівнева адаптивна методика
Джерела міток	Лише експертні мітки (усі тікети через LEAD)	Експертні мітки + псевдомітки	Експертні корекції LEAD + ML+LLM узгодження + LLM + критичні помилки
Стратегія зважування	Усі приклади: w = 1,0	Експерт: w = 1,0 Псевдомітки: w = 0,5 (фіксована вага)	Адаптивні ваги Tier 1–4: w ∈ [0,51; 1,2] залежно від джерела й довіри
Цільове навчання на критичних помилках	Ні	Ні	Так: критичні помилки ML мають вагу w = 1,2
Інтеграція LLM	Немає	Може використовуватись лише для псевдоміток	LLM як частина ансамблю: ML+LLM узгодження
Безперервне донавчання на живому трафіку	Потрібен ручний retrain	Обмежено, без врахування ваг Tier	Донавчання кожні ~40 нових тікетів з урахуванням ваг Tier 1–4
Пояснюваність рішень	Лише історія ручних рішень	Частково	Для кожного тікета зберігаються Tier, джерело мітки, reasoning LLM, triage reason
Автоматизація тікетів	≈0% усе через експерта	Обмежена (псевдомітки з фіксованою вагою)	≈70% тікетів обробляються автоматично, експерт працює з ≈30%
Точність	78,9%	75,6%	83,6%

Методика 4-рівневої адаптивної системи зважування Tier 1–4 для навчання

Рівень	Джерело	Вага	Умова віднесення до рівня
1	Корекції LEAD	1,0	Експертна зміна в triage = максимальна довіра
2	Узгодження ML+LLM	0,68–0,85	ML і LLM дають однакову мітку, обидві conf ≥ 0,6 w = min(conf_ML, conf_LLM) × 0,85
3	Тільки LLM	0,51–0,60	LLM пропонує мітку з обґрунтуванням
4	Критичні помилки	1,2	Критична помилка. ML був впевнений (conf ≥ 0.80), але помилився й LEAD виправив вручну



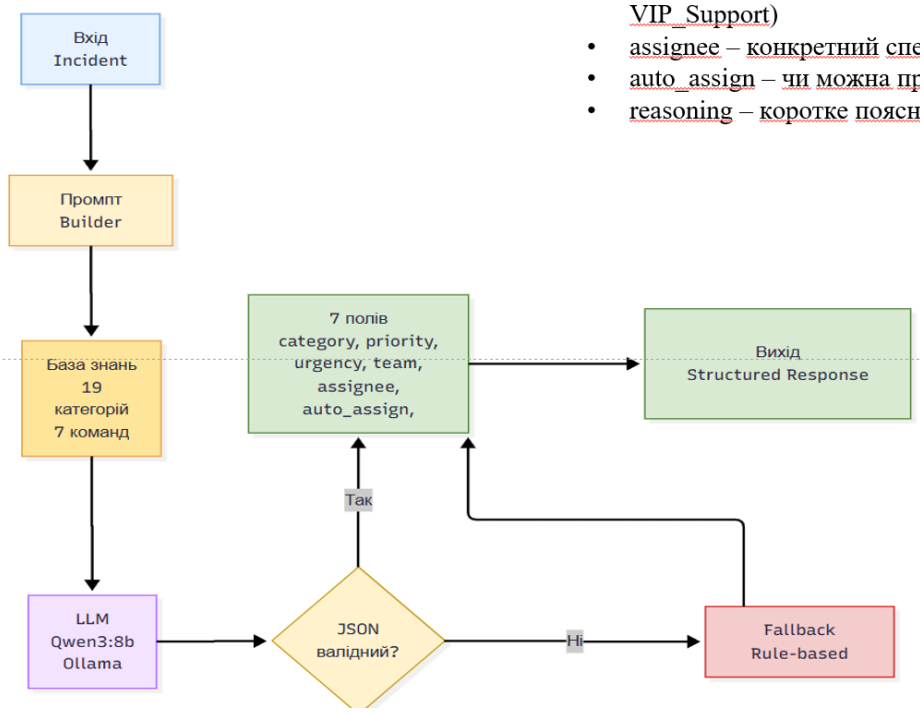
Порівняння функціональності прототипу із типовими підходами

Функція / Ознака	Ручний процес / базові системи	Сингл-ML класифікація	Комерційна AI-ITSM платформа ServiceNow	Прототип
Авто-класифікація пріоритету/категорії	Ні ,все вручну	Так ,ML (TF-IDF/BERT)	Так, AI-класифікація	ТакML (BERT) + “друга думка” локальної LLM
Ансамбль ML+LLM	Ні	Ні	Частково, як чорний ящик	Так, AGREEMENT, WEIGHTED_VOTE, CRITICAL_DISAGREEMENT
Triage за впевненістю/конфліктом	Ні	Обмежено, простий поріг	Так, але без прозорих правил	Так, triage_required + triage_reason
Автопризначення (Smart Assignment)	Ні або проста черга	Частково, rule-based	Такрозподіл навантаження	Так, 5-факторний скоринг + ML p_assign
Пояснюваність рішень	Ні	Ні	Частково (AI-коментарі)	Так, друга думка від локальної llm_reasoning ensemble_reasoning, LIME пояснення
Активне перевчання з фідбеку	Ні	Рідко, ручні retrain	Так, авто-оновлення моделей	Так, feedback CSV/БД та retrain з новими тікетами
Логи та аналітика ML vs LLM	Ні	Обмежено	Агреговані звіти	Так, детальний ML_PREDICTION_LOG зі змінами тікету та triage
Ручний оверрайд у UI	Так ,але без навчання системи	Так	Так	Так, кожен оверрайд іде у feedback для моделей
Використання історії тікетів	Ні	Частково, разові перенавчання	Так, Непрозоро	Так, історія/feedback → нові версії моделей
Локальна LLM	Ні	Ні	Зазвичай хмарні AI-модулі	Так, Qwen 3:8B локально, дані не виходять за межі системи

LLM-маршрутизація тікетів та структура промту:

Структура промту до LLM-асистента маршрутизації інцидентів складається з :

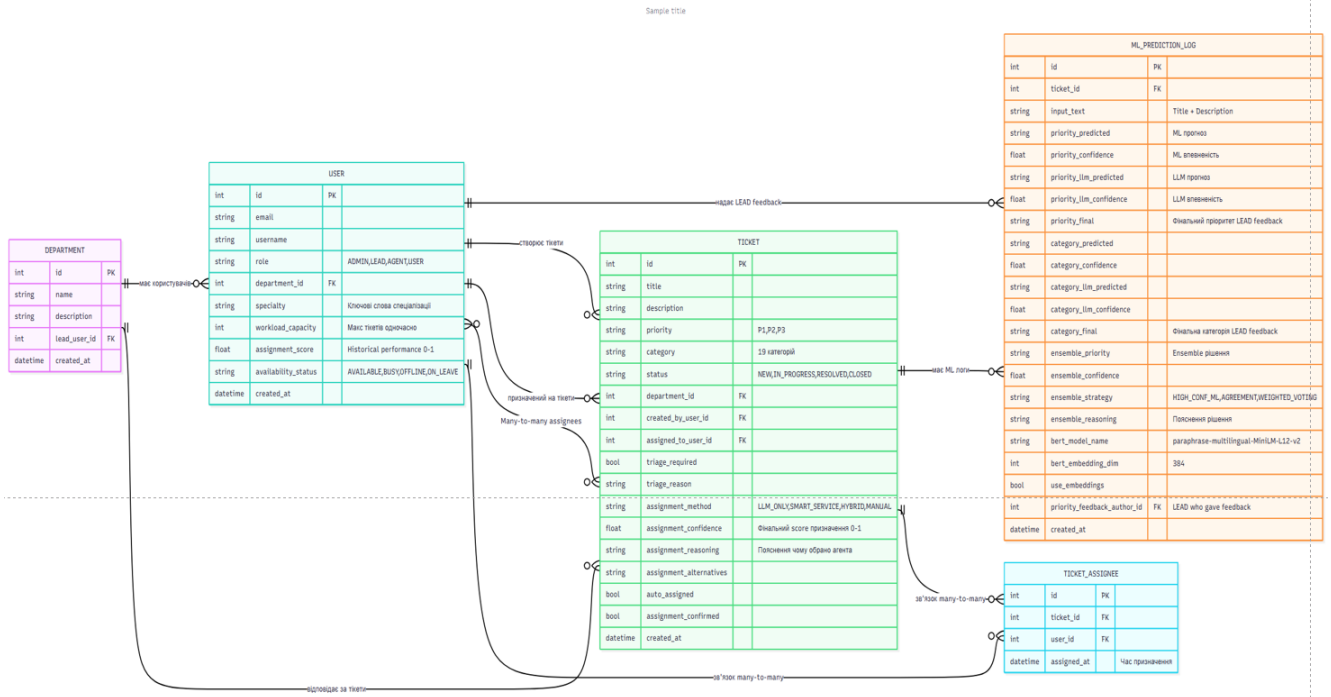
•Роль системи
•Результат LLM
•Категорії/правила в промті
•Вхідні дані тікету



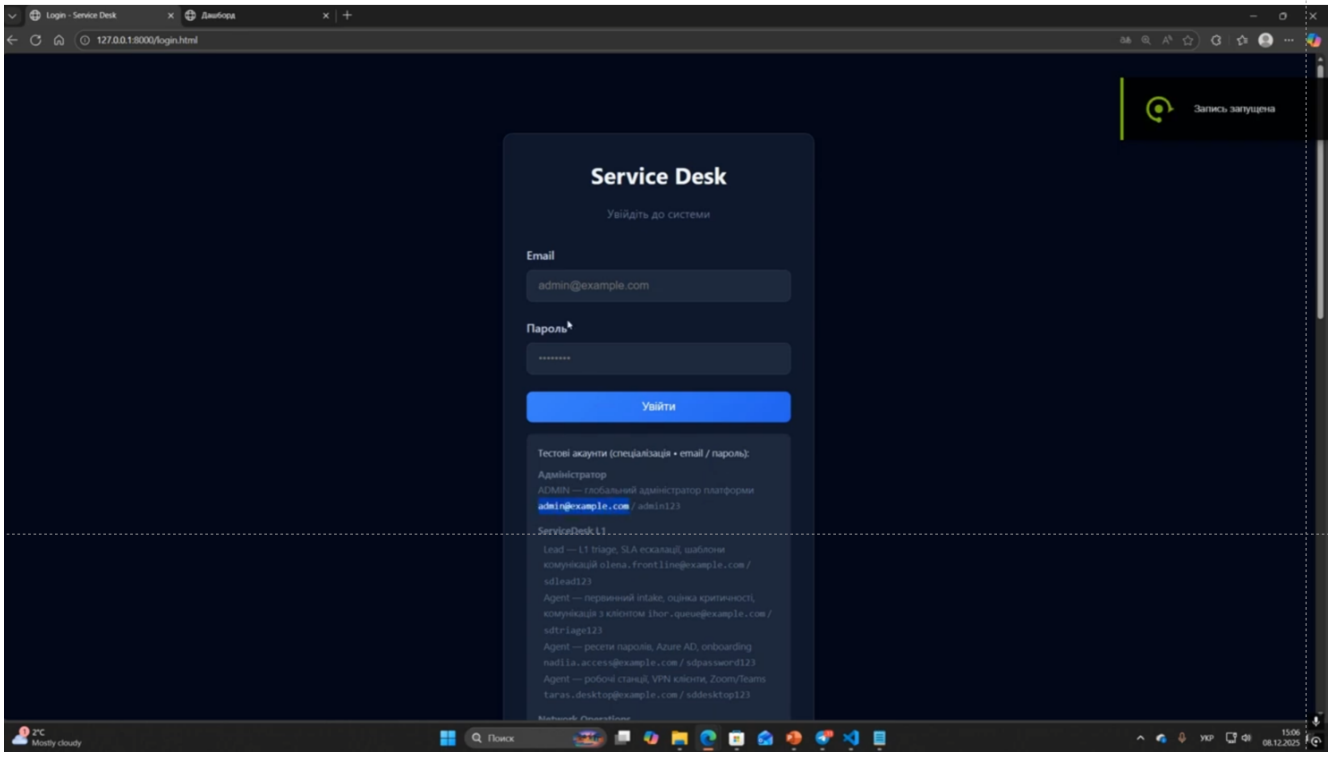
Результат LLM повертає:

- category – 19 спеціалізованих категорій інцидентів
- priority – бізнес-вплив: P1 / P2 / P3
- team – одна з 7 команд підтримки (ServiceDesk_L1, Network_Ops, Workplace_Eng, Application_Support, Security_Access, Billing_Finance, VIP_Support)
- assignee – конкретний спеціаліст або null
- auto_assign – чи можна призначити автоматично (true/false)
- reasoning – коротке пояснення вибору (explainability)

Схема даних: тікети, користувачі та ML-логи



Робота прототипу



ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

1. Золотухіна О.А. Щербаченко О.С. Порівняльний аналіз хмарних рішень для створення ІМ-системи : Матеріали VI Всеукраїнської науково-технічна конференція « Застосування програмного забезпечення в інформаційно комунікаційних технологіях ».15.05.2025, ДУІКТ, м. Київ. Збірник тез. К.: ДУІКТ,2025.С.495-498.
2. Золотухіна О.А. Щербаченко О.С. Методологічні аспекти розробки системи Incident Management з використанням хмарних технологій : Матеріали VI Всеукраїнської науково-технічна конференція « Застосування програмного забезпечення в інформаційно комунікаційних технологіях ».15.05.2025, ДУІКТ, м. Київ. Збірник тез. К.: ДУІКТ,2025.С.532-535.