МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Інформаційна система обліку успішності студентів з природномовним інтерфейсом**

Виконав: студент 4 курсу, групи 6.1212-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
С.С. Сидоренко
(ініціали та прізвище)
Керівник доцент, А.І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Діскус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ім. Ю.М. Потебні ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Тетяна КРИТСЬКА

“ 01 ” березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Сидоренку Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інформаційна система обліку успішності студентів з природномовним інтерфейсом

керівник роботи Безверхий Анатолій Ігорович, доцент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджена наказом ЗНУ від 23.12.2025 р. № 1830-с

2. Строк подання студентом кваліфікаційної роботи 02.06.2026
3. Вихідні дані бакалаврської роботи
   - комплект нормативних документів ;
   - технічне завдання до роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   - огляд та збір літератури стосовно теми кваліфікаційної роботи;
   - огляд та аналіз існуючих інформаційних систем ЗВО та аналогів;
   - дослідження технологій обробки природної мови та методів трансляції запитів у структурований SQL-код (Text-to-SQL);
   - проєктування архітектури інтелектуальної системи, бази даних та логіки безпеки;
   - створення програмного продукту (AI-системи деканату) та опис його компонентів;
   - визначення вимог для роботи програми та конфігурації апаратного забезпечення;
   - розробка висновків та пропозицій.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 01.03.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів магістерської роботи	Примітка
1.	Одержання завдання на кваліфікаційну роботу, збір та аналіз науково-технічної літератури за темою дослідження.	02.03-4.03	виконано
2.	Аналіз предметної області та дослідження процесів управління інформацією в деканатах ЗВО.	05.03-06.03	виконано
3.	Огляд систем-аналогів та дослідження технологій трансляції природної мови у бази даних (Text-to-SQL).	07.03-09.03	виконано
4.	Обґрунтування вибору технологічного стека (Node.js, PostgreSQL) та архітектури локальних мовних моделей (Ollama).	11.03-20.03	виконано
5.	Проєктування концептуальної, логічної та фізичної схеми реляційної бази даних.	20.03-27.03	виконано
6.	Створення аналітичних представлень (VIEW) у БД для оптимізації роботи нейромережевих алгоритмів.	27.03-04.04	виконано
7.	Розробка алгоритму інтелектуального маршрутизатора (Router LLM) для семантичної класифікації запитів.	05.04-12.04	виконано
8.	Програмна реалізація серверної частини застосунку (Backend) та налаштування середовища виконання.	13.04-20.04	виконано
9.	Інтеграція підсистеми семантичної безпеки на основі рольової моделі доступу (RBAC).	21.04-25.04	виконано
10.	Проєктування та програмна реалізація клієнтського веб-інтерфейсу (Frontend).	26.04-01.05	виконано
11.	Розробка модулів крос-платформної звітності та експорту даних у формати PDF, Excel, DOCX.	02.05-10.05	виконано
12.	Калібрування промпт-інструкцій та оптимізація взаємодії з моделями qwen2.5-coder та deepseek-r1.	11.05-16.05	виконано
13.	Проведення системного тестування роботи застосунку та оцінка точності генерації SQL-коду.	16.05-22.05	виконано
14.	Рефакторинг програмного коду за результатами тестування та фінальне налаштування системи.	23.05-28.05	виконано
15.	Оформлення пояснювальної записки, підготовка ілюстративного матеріалу та лістингів коду.	29.05-06.05	виконано
16.	Оформлення презентації, отримання відгуку, рецензії та подання роботи на захист.	07.05-13.06	в процессі

Студент _____ _____С.С. Сидоренко

Керівник роботи _____ _____А. І. Безверхий

Нормоконтроль пройдено

Нормоконтролер _____ _____І.А. Скрипник

АНОТАЦІЯ

Сторінок — 70,

Рисунків — 6,

Джерел — 15.

Сидоренко С.С. Інформаційна система обліку успішності студентів з природномовним інтерфейсом: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник А. І. Безверхий. Запоріжжя : ЗНУ, 2026. 70 с.

Метою роботи є розробка інформаційно-аналітичної системи деканату з природномовним інтерфейсом Text-to-SQL на основі гібридної архітектури великих мовних моделей. У ході дослідження створено реляційну базу даних PostgreSQL із шаром аналітичних представлень (VIEW) та серверну частину на Node.js. Реалізовано алгоритм триконтурної маршрутизації лінгвістичних потоків між локальними моделями Qwen2.5-Coder, DeepSeek-R1 та хмарними API. Забезпечено ешелонований захист сховища через динамічну ін'єкцію правил доступу (RBAC), лексичний фаєрвол isSafeSQL та інфраструктурну ізоляцію Read-Only пулу з’єднань СУБД. Практичне значення результатів полягає в автоматизації формування багатовимірних аналітичних зрізів успішності студентів та детермінованому експорті регламентованої звітності у формати PDF, Excel, Word.

Ключові слова: *електронний деканат, Text-to-SQL, гібридний інференс, PostgreSQL, Node.js, управління доступом, RBAC, інформаційна безпека.*

ABSTRACT

Pages — 70,

Figures — 6,

Sources — 15.

Sidorenko S.S. Information System for Tracking Student Performance with a Natural Language Interface: qualification thesis of the bachelor of specialty 121 "Software Engineering" / Supervisor A. I. Bezverkhyi. Zaporizhzhia: ZNU, 2026. 70 p.

The objective of the thesis is to develop an information-analytical system for the dean's office with a natural language Text-to-SQL interface based on a hybrid architecture of large language models. A PostgreSQL relational database with a layer of analytical views (VIEW) and a Node.js backend were developed. A three-tier routing algorithm for linguistic streams between local models (Qwen2.5-Coder, DeepSeek-R1) and cloud APIs was implemented. A defense-in-depth security framework was provided through dynamic access rule injection (RBAC), the isSafeSQL lexical firewall, and infrastructural isolation of a Read-Only connection pool. The practical significance lies in automating multi-dimensional student performance analytics and deterministic report exporting to PDF, Excel, and Word formats.

Keywords: *electronic dean's office, Text-to-SQL, hybrid inference, PostgreSQL, Node.js, access management, RBAC, information security.*

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 15
1.1 Сутнісна характеристика та еволюція інформаційно-аналітичних систем деканату 15
1.2 Теоретичні засади функціонування великих мовних моделей та механіка трансляції Text-to-SQL 17
1.3 Архітектури розгортання штучного інтелекту в освітніх системах 21
1.4 Оптимізація контекстного навчання мовних моделей та шару аналітичної логіки 25
1.5 Механізми забезпечення інформаційної безпеки та управління доступом 27
1.6 Огляд та критичний аналіз існуючих програмних продуктів-аналогів 31
1.7 Постановка завдання 33
1.8 Висновки до розділу 1 35
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ 36
2.1 Аналіз функціональних та нефункціональних вимог 36
2.1.1 Декомпозиція та специфікація функціональних вимог за рольовою моделлю 36
2.1.2 Формалізація нефункціональних вимог до архітектури та безпеки комплексу 38
2.2 Обґрунтування вибору технологічного стеку та архітектури розгортання 40
2.3 Проєктування реляційної моделі бази даних 43
2.3.1 Логічна структура та граф зв'язків сутностей сховища даних 43
2.3.2 Фізична специфікація таблиць та забезпечення реляційної цілісності 46
2.4 Проєктування аналітичного шару та оптимізація когнітивного навантаження на ШІ 48

2.4.1 Концепція перенесення бізнес-логіки на рівень реляційного ядра СУБД 48
2.4.2 Специфікація аналітичних представлень для обчислення рейтингів та успішності 49
2.5 Проєктування механізмів семантичної безпеки та управління доступом (RBAC) 52
2.5.1 Семантичне екранування та динамічна ін'єкція правил рольового доступу 52
2.5.2 Лексична валідація та інфраструктурний захист реляційного ядра 53
2.6 Висновки до розділу 2 54
3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО КОМПЛЕКСУ 55
3.1 Реалізація серверної частини та інтеграція з базою даних 55
3.1.1 Модульна структура та конфігурація середовища виконання 55
3.1.2 Організація пулів з'єднань СУБД та забезпечення транзакційної стійкості 56
3.2 Реалізація інтелектуального шару та триконтурної маршрутизації запитів 58
3.2.1 Алгоритм класифікації інтенту та диспетчеризації моделей 58
3.2.2 Динамічна генерація контексту та інтеграція правил безпеки 60
3.3 Розробка клієнтського застосунку та інтеграція модулів експорту звітності 64
3.3.1 Архітектура клієнтського інтерфейсу та обробка асинхронних запитів 64
3.3.2 Реалізація підсистеми експорту аналітичної звітності 65
3.4 Тестування системи 66
3.5 Висновки до розділу 3 67
ВИСНОВКИ 69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 70

ВСТУП

Актуальність теми

Сучасний етап розвитку вищої освіти в Україні характеризується стрімкою цифровізацією та накопиченням значних масивів структурованої інформації про навчальний процес на рівнях кафедр та факультетів. Ефективне управління цими активами вимагає від керівного складу освітніх установ оперативного отримання багатовимірної аналітичної звітності щодо успішності студентів, кадрового навантаження та динаміки академічних показників [1]. Проте традиційні інформаційні системи електронних деканатів мають жорстко обмежену функціональність, яка базується на статичних, наперед запрограмованих шаблонах звітів. Будь-який нестандартний аналітичний запит вимагає від співробітників або прямого володіння мовою структурованих запитів SQL, або ручного вивантаження «сирих» даних у сторонні табличні процесори для подальшої обробки, що створює високий технічний бар'єр і знижує швидкість прийняття управлінських рішень. Проєктування таких систем потребує впровадження надійних, масштабованих архітектур обробки інформації [2] та використання ефективних платформ менеджменту великих даних [3].

Новітні досягнення в галузі штучного інтелекту, зокрема поява великих мовних моделей архітектури Transformer, відкривають інноваційні можливості для подолання цієї проблеми за допомогою технології Text-to-SQL, яка дозволяє здійснювати автоматичну трансляцію природномовних запитів користувачів у валідний машинний код. Фундаментом для таких рішень виступають сучасні методи обробки природної мови [4] та готові програмні фреймворки інтеграції лінгвістичних моделей із реляційними базами даних [5]. Водночас розгортання таких інтелектуальних інтерфейсів в освітньому середовищі супроводжується жорсткими інфраструктурними викликами. Використання виключно комерційних хмарних сервісів штучного інтелекту створює прямі ризики порушення законодавства про захист

персональних даних через передачу конфіденційної інформації про студентів на сторонні сервери. З іншого боку, повна автономізація обчислень на базі локальних мовних моделей малої розмірності часто призводить до деградації точності та появи логічних галюцинацій при обробці складних аналітичних інтентів. Таким чином, науково-практична актуальність роботи зумовлена необхідністю проєктування та реалізації гібридної, відмовостійкої архітектури інформаційної системи, яка органічно поєднує безпеку локального інференсу штучного інтелекту із гнучкістю хмарного масштабування та багаторівневим ешелонованим захистом реляційного сховища.

Мета і завдання дослідження

Метою кваліфікаційної роботи є проєктування, програмна реалізація та комплексне тестування інтелектуальної інформаційно-аналітичної системи електронного деканату, яка забезпечує безпечну, стабільну та інтуїтивно зрозумілу взаємодію користувачів із нормалізованим сховищем даних через природномовний інтерфейс Text-to-SQL на основі гібридної топології великих мовних моделей.

Для досягнення поставленої мети в межах дослідження необхідно послідовно вирішити комплекс взаємопов'язаних інженерних та науково-практичних завдань. По-перше, вимагається провести детальний системний аналіз предметної області та критичний огляд існуючих програмних продуктів-аналогів для виявлення архітектурних прогалин у сучасних освітніх платформах. По-друге, необхідно спроєктувати суворо нормалізовану реляційну модель бази даних у середовищі СУБД PostgreSQL [6], яка б повністю виключала виникнення аномалій модифікації інформації, та розробити шар аналітичних представлень на основі офіційних специфікацій системи керування [7]. По-третє, завдання передбачає програмну реалізацію асинхронного серверного ядра на базі платформи Node.js [8] та фреймворку Express [9] із впровадженням механізму розділених пулів з'єднань СУБД. По-четверте, необхідно розробити та інтегрувати алгоритм триконтурної

інтелектуальної маршрутизації лінгвістичних потоків між локальними середовищами Ollama [10], оптимізованими програмістськими моделями Qwen-Coder [11], важкими міркувальними контурами DeepSeek-R1 та захищеними хмарними інтерфейсами прикладного програмування. По-п'яте, вимагається спроєктувати та впровадити дворівневу систему семантичної та інфраструктурної безпеки, що поєднує правила рольового доступу RBAC із лексичним фаєрволом згенерованого коду, враховуючи стандарти захисту від веб-уразливостей [12]. По-шосте, необхідно створити адаптивний клієнтський односторінковий вебзастосунок на базі сучасних клієнтських архітектур [13], компонентів реактивного проектування [14] та інструментів інтеграції інтерфейсів [15] з вбудованою підсистемою детермінованого експорту звітності у регламентовані формати та провести всебічне тестування працездатності й стійкості розробленого програмного комплексу.

Об’єкт дослідження

Процеси збереження, обробки, аналізу та оперативного вилучення інформації про успішність студентів, кадровий склад та організаційну структуру в межах функціонування деканату закладу вищої освіти.

Предмет дослідження

Методи, моделі, алгоритми та програмні засоби автоматичної трансляції аналітичних запитів користувачів із природної мови у структурований SQL-код за допомогою гібридних контурів великих мовних моделей в умовах суворого розмежування прав доступу та інфраструктурної ізоляції транзакцій.

Методи дослідження

Для розв'язання визначених завдань у роботі застосовано такі методи: системний аналіз предметної області та існуючих рішень, принципи проєктування реляційних баз даних, методи обробки природної мови та технології промпт-інжинірингу для налаштування взаємодії з мовними

моделями, методології вебпрограмування для створення серверної та клієнтської частин застосунку, а також методи емпіричного, функціонального та негативного тестування безпеки програмного забезпечення.

Наукова новизна одержаних результатів

Наукова новизна дослідження полягає в удосконаленні архітектурного підходу до побудови Text-to-SQL інтерфейсів шляхом повного винесення важкої обчислювальної бізнес-логіки та віконних функцій ранжування з лінгвістичного контексту мовної моделі безпосередньо у реляційне ядро СУБД у вигляді аналітичних представлень. Це дозволило формалізувати когнітивне завдання штучного інтелекту до генерації лінійних операторів вибірки, що радикально знижує ймовірність логічних галюцинацій моделей і стабілізує роботу інтерфейсу.

Практичне значення одержаних результатів полягає у створенні промислово готового, відмовостійкого програмного комплексу інформаційно-аналітичної системи «Електронний деканат», який може бути безшовно розгорнутий на локальних серверах навчального закладу. Впроваджений дворівневий безпековий контур, що поєднує серверний лексичний фаєрвол та інфраструктурну Read-Only ізоляцію пулу з'єднань, гарантує високий рівень захисту від руйнування структури сховища чи несанкціонованого доступу до персональних даних студентів. Модуль експорту забезпечує автоматичне перетворення результатів природномовного пошуку у готові офіційні документи форматів PDF, Excel та Word, що оптимізує щоденну адміністративну діяльність факультету.

Практичне значення одержаних результатів

Розроблено повнофункціональний прототип вебзастосунку інтелектуальної системи деканату, що має універсальну архітектуру та підтримує гнучке перемикання між локальним (Ollama) та хмарним ШІ-контурами. Інтеграція сучасного асинхронного серверного стека та

спеціалізованих сервісів генерації звітів дозволяє повністю автоматизувати формування людино-зрозумілих підсумкових документів у форматах PDF, Excel та Word безпосередньо з чат-інтерфейсу. Дослідно доведено можливість стабільного інференсу системи на споживчому обладнанні із забезпеченням стовідсоткової відтворюваності аналітичних зрізів за рахунок детермінізації генерації.

Глосарій

- AI (Artificial Intelligence) — штучний інтелект; галузь комп'ютерних наук, що займається створенням інтелектуальних систем, здатних до навчання та прийняття рішень.
- API (Application Programming Interface) — інтерфейс прикладного програмування; набір правил та протоколів, що дозволяють різним програмним компонентам взаємодіяти між собою.
- Backend — серверна частина програмного застосунку; компонент системи, що відповідає за обробку бізнес-логіки, запитів до бази даних та взаємодію з AI-контуром.
- CORS (Cross-Origin Resource Sharing) — механізм, що дозволяє вебзастосунку отримувати ресурси з домену, відмінного від того, на якому він розміщений.
- DeepSeek-R1 — спеціалізована локальна мовна модель, яка використовується в системі для аналітичної обробки складних запитів.
- Frontend — клієнтська частина програмного застосунку; інтерфейс, з яким безпосередньо взаємодіє користувач.
- HTTP (HyperText Transfer Protocol) — протокол передачі гіпертексту; основний протокол обміну даними в мережі інтернет.
- JWT (JSON Web Token) — відкритий стандарт для створення токенів доступу, що використовуються для безпечної передачі інформації між сторонами у форматі JSON.

- LangChain — програмний фреймворк, призначений для створення застосунків на основі мовних моделей шляхом з'єднання їх з зовнішніми джерелами даних.
- LLM (Large Language Model) — велика мовна модель; тип нейронної мережі, навченої на масивах текстових даних для розуміння та генерації природної мови.
- Node.js — середовище виконання JavaScript-коду, що працює на стороні сервера та використовується для побудови масштабованих мережевих застосунків.
- Ollama — інструментарій для розгортання, запуску та керування локальними мовними моделями в межах власної інфраструктури.
- PostgreSQL — об'єктно-реляційна система управління базами даних (СУБД) з відкритим кодом.
- Qwen2.5-Coder — спеціалізована локальна мовна модель, оптимізована для швидкого аналізу коду та генерації простих SQL-запитів.
- RBAC (Role-Based Access Control) — модель управління доступом на основі ролей (student, lecturer, dean), що забезпечує семантичну безпеку шляхом динамічної ін'єкції просторових обмежень (Guardrails) безпосередньо у системний промпт мовної моделі на етапі ініціалізації інференсу .
- REST API (Representational State Transfer) — архітектурний стиль взаємодії компонентів розподіленої системи в мережі.
- Router LLM — нейромережевий диспетчер інтенту (функціонує на базі компактної моделі qwen2.5:1.5b), що здійснює первинний семантичний аналіз запиту, класифікує його на лінійний (SIMPLE) або аналітичний (ANALYTIC) та здійснює маршрутизацію до відповідного виконавчого контуру .
- SLM (Small Language Model) — мала мовна модель; оптимізована модель, що має меншу кількість параметрів та вищі показники енергоефективності при роботі на споживчому обладнанні.

- SQL (Structured Query Language) — мова структурованих запитів, що використовується для створення, модифікації та управління даними в реляційних базах даних.
- Text-to-SQL — технологія штучного інтелекту, що забезпечує автоматичну трансляцію природномовного запиту користувача в SQL-запит до бази даних.
- VIEW — віртуальна таблиця (представлення) в СУБД, що базується на результаті виконання SQL-запиту, та використовується для спрощення роботи з даними.
- ЗВО — заклад вищої освіти.
- Гібридний інференс (Hybrid Inference) — архітектурний підхід, що забезпечує здатність системи динамічно перемикати лінгвістичні обчислювальні потоки між ізольованими локальними моделями (On-Premises) та зовнішніми хмарними сервісами (Cloud API) залежно від конфігурації середовища.
- Gemini (Google Generative AI) — сімейство хмарних великих мовних моделей, зокрема версія gemini-2.5-flash, яка імплементована в систему як альтернативний високошвидкісний та міркувальний контур у гібридному режимі роботи.
- Лексичний фаєрвол (Lexical Firewall) — програмний модуль (зокрема, функція isSafeSQL), призначений для посимвольного регулярного аналізу згенерованого штучним інтелектом коду з метою превентивного блокування деструктивних операторів маніпулювання даними (DROP, DELETE, UPDATE, GRANT тощо) до моменту звернення до реляційного ядра.
- SPA (Single Page Application) — односторінковий вебзастосунок; архітектура розгортання клієнтської частини системи, яка забезпечує динамічний рендеринг аналітичних таблиць та інтерфейсу чату без необхідності повного перезавантаження сторінки в браузері.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сутнісна характеристика та еволюція інформаційно-аналітичних систем деканату

Сучасний заклад вищої освіти функціонує як складна ієрархічна макросистема, ефективність управління якою прямо пропорційна швидкості обробки та рівню достовірності управлінської інформації. Деканат, виступаючи базовим адміністративним та організаційним вузлом факультету, відіграє роль головного концентратора інформаційних потоків. Відповідно, інформаційно-аналітична система (ІАС) деканату становить собою спеціалізований програмно-апаратний комплекс, спроєктований для збору, суворої нормалізації, тривалого зберігання та багатовимірного аналізу даних щодо навчального процесу.

Історично еволюція програмного забезпечення для закладів освіти пройшла лінійний шлях від простих баз даних транзакційного типу (Online Transaction Processing, OLTP), які були орієнтовані виключно на електронний облік, до складніших систем підтримки прийняття рішень (Decision Support Systems, DSS). Базовий функціонал традиційного електронного деканату здебільшого обмежується веденням стандартизованих реєстрів: обліком контингенту студентів, формуванням навчальних планів, закріпленням дисциплін за кафедрами, фіксацією поточної успішності та генерацією статичних звітів за жорстко запрограмованими шаблонами. З точки зору системного аналізу, такі інформаційні рішення працюють виключно на рівні реєстрації поточних станів. Вони не забезпечують вбудованих механізмів динамічного видобування неочевидних взаємозв'язків з масивів інформації (Data Mining).

Як наслідок, для отримання будь-якої нестандартної аналітичної вибірки вимагається постійне залучення вузькопрофільних ІТ-фахівців. Необхідність ручного написання SQL-запитів адміністраторами баз даних створює суттєвий

організаційний бар'єр, який уповільнює процес прийняття оперативних управлінських рішень керівництвом факультету. Перехід від парадигми простої облікової системи до інтелектуальної аналітичної зумовлений гострою необхідністю оптимізації управління людськими та часовими ресурсами. У конкурентному освітньому середовищі швидкість ідентифікації проблемних зон є базовою стратегією ефективного функціонування макросистеми університету.

Класична архітектура реляційних баз даних вимагає формування синтаксично точних запитів мовою структурованих запитів (SQL), що формує технічну прірву між безпосереднім масивом даних та кінцевим користувачем, яким виступає декан, методист, куратор академічної групи або здобувач освіти. Інтелектуальна аналітична система деканату принципово усуває цей бар'єр шляхом впровадження спеціалізованого модуля трансляції природної мови у структуровані машинно-зрозумілі запити (концепція Text-to-SQL).

Архітектура інноваційної системи має яскраво виражений модульний характер і передбачає глибоку декомпозицію на кілька ізольованих функціональних рівнів. Базовим рівнем виступає рівень зберігання даних (Data Layer), представлений суворо нормалізованою реляційною базою даних під керуванням PostgreSQL, конфігурація якої унеможливлює дублювання інформації. Цілісність даних на цьому рівні забезпечується через жорсткі механізми зовнішніх ключів (Foreign Keys) та збережених процедур чи аналітичних представлень, що перебирають на себе критичну частину бізнес-логики.

Над сховищем даних надбудовується рівень бізнес-логіки та безпеки (Backend Layer), який виконує функції маршрутизатора. Цей вузол ідентифікує поточного користувача, визначає його привілеї в межах рольової моделі (Role-Based Access Control) та превентивно накладає просторові обмеження на доступ до інформації ще до етапу генерації вибірки. Третім компонентом виступає рівень семантичного аналізу (AI Layer). Отримуючи метадані про фізичну структуру бази даних та природномовний запит

користувача, цей інтелектуальний агент виявляє причинно-наслідкові зв'язки у тексті та кодує валідний SQL-синтаксис. Завершальною ланкою архітектури є рівень представлення (Frontend Layer), що здійснює прийом вхідних запитів, агрегацію отриманих від серверної частини наборів даних та їх візуалізацію у вигляді динамічних таблиць і звітів.

Застосування великих мовних моделей у наведеній архітектурі виконує критично важливу функцію адаптивного семантичного інтерфейсу між природною людською мовою та строгою реляційною алгеброю СУБД. Проте впровадження таких інструментів в освітній сфері накладає суворі обмеження, оскільки персональні дані студентів, ідентифікатори залікових книжок та академічні показники становлять критичний інформаційний актив освітньої установи. Передача зазначених даних на зовнішні комерційні сервери створює неприпустимі юридичні та репутаційні вразливості.

З цієї причини проектування сучасної ІАС вимагає впровадження гнучкого гібридного інференсу лінгвістичних потоків. У межах такої парадигми система здатна функціонувати в абсолютно замкненому локальном контурі (Air-gapped середовищі), обробляючи критичні запити на внутрішніх апаратних потужностях організації за допомогою розгортання моделей штучного інтелекту малої розмірності. Водночас для виконання загальних аналітичних задач, тестування, відлагодження або опрацювання знеособлених масивів даних архітектура підтримує можливість прозорого перемикання на хмарні обчислювальні контури (Cloud API), що дозволяє гнучко балансувати між абсолютним суверенітетом інформаційних активів та масштабованістю обчислень.

1.2 Теоретичні засади функціонування великих мовних моделей та механіка трансляції Text-to-SQL

Функціонування сучасної інтелектуальної інформаційно-аналітичної системи деканату базується на фундаментальних математичних та

лінгвістичних принципах роботи великих мовних моделей (LLM). Сучасний етап розвитку систем штучного інтелекту в галузі обробки природної мови (NLP) характеризується глобальним переходом від рекурентних нейронних мереж (RNN) та архітектур з довгою короткостроковою пам'яттю (LSTM) до парадигми Transformer. Впровадження зазначеної технології дозволило концептуально вирішити проблему втрати довгострокових залежностей у тексті, яка була критичним недоліком попередніх поколінь нейромереж, а також забезпечило можливість паралельної обробки послідовностей токенів, що суттєво прискорило обчислювальний процес інференсу на локальному апарному забезпеченні. Ключовим обчислювальним компонентом архітектури Transformer виступає механізм внутрішньої уваги (Self-Attention), який дозволяє моделі динамічно розраховувати ваги семантичного зв'язку між кожною парою токенів у вхідному запиті, незалежно від їхньої фізичної відстані у текстовому рядку. Математично цей процес описується як функція від трьох матриць, а саме запитів (Query, Q), ключів (Key, K) та значень (Value, V), де базовий розрахунок ваг контекстної залежності виконується за формулою:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

У наведеній математичній моделі скалярний добуток матриць запитів та транспонованої матриці ключів QK^T визначає ступінь взаємної релевантності та семантичної близькості слів у структурі речення. Отримані значення діляться на масштабуючий коефіцієнт $\sqrt{d_k}$, де d_k відображає точну розмірність векторів ключів. Цей дільник виконує критичну стабілізуючу функцію, оскільки при великих значеннях розмірності векторів скалярний добуток стрімко зростає, що призводить до потрапляння функції активації у зони з надзвичайно малими значеннями градієнтів (проблема затухання градієнтів) під час навчання та інференсу. Оператор softmax виконує нормалізацію отриманих скалярних величин,

перетворюючи їх на ймовірнісний розподіл ваг уваги, які згодом множаться на матрицю значень V для формування фінального контекстно-залежного векторного представлення сутності. В умовах предметної області електронного деканату, коли користувач формулює складний багатовимірний запит із багатьма критеріями, механізм уваги дозволяє мовній моделі надійно утримувати логічні зв'язки між віддаленими лексемами. Базовим середовищем для функціонування цього механізму є векторне представлення слів (Embeddings), за якого кожен текстовий токен трансформується у багатовимірний вектор у щільному математичному просторі. Таке представлення забезпечує просторове зближення схожих понять, дозволяючи системі ефективно опрацьовувати варіативність природної мови та розпізнавати синоніми чи специфічні абревіатури. Завдання трансляції природної мови у структуровані запити до баз даних, відоме в інженерній практиці як Text-to-SQL, становить специфічний підклас задач детермінованої генерації коду. Сутність цього процесу полягає у безпомилковому відображенні неструктурованого лінгвістичного простору на суворо нормалізовану фізичну схему реляційної СУБД. На відміну від вільної генерації природномовного тексту, де допускається певний рівень семантичної чи стилістичної похибки, технологія Text-to-SQL вимагає абсолютної синтаксичної точності. Будь-яка неточність у назві поля або операторі неминуче викликає відхилення запиту на рівні ядра бази даних.

Механіка трансляційного процесу розгортається через чітку послідовність аналітичних етапів:

- Контекстний аналіз та детекція інтенту: Здійснюється розбиття вхідного рядка на токени, визначення мети запиту (лінійний чи аналітичний) та фільтрація позадоменних виразів.
- Зв'язування зі схемою (Schema Binding): Модель точно зіставляє виявлені в тексті природні назви з реальними ідентифікаторами таблиць та колонок, наданими у системному промпті. На цьому ж етапі

виконується трансформація лінгвістичних концептів у математичні агрегації.

- Генерація логіки з'єднань: Алгоритм, спираючись на метадані про зовнішні ключі (Foreign Keys), самостійно реконструює граф зв'язків між ізольованими таблицями, формуючи синтаксично валідні оператори JOIN.
- Формування умов фільтрації та групування: Виявлені обмеження перетворюються на логічні оператори мови SQL (WHERE, GROUP BY, ORDER BY) із суворим дотриманням сумісності типів даних реляційного сховища.

Забезпечення високого рівня точності на всіх етапах трансляції безпосередньо залежить від профільної спеціалізації та архітектурного типу обраної мовної моделі. Загальні моделі (General-purpose LLMs) схильні до логічних хиб та галюцинацій, які у контексті баз даних проявляються у вигляді вигадування неіснуючих стовпців або ігнорування реляційних зв'язків. Для вирішення цієї проблеми в сучасних IAC застосовується диференційований підхід, що базується на використанні двох різних класів моделей:

- Спеціалізовані Coder-моделі (наприклад, Qwen2.5-Coder): Проходять масштабне попереднє навчання на мільярдах рядків вихідного коду, DDL-схемах та еталонних SQL-структурах. Завдяки цілеспрямованому зміщенню ваг нейронної мережі у бік синтаксису програмування, такі компактні локальні моделі демонструють високу швидкість та точність дотримання заданих правил найменування при обробці простих, лінійних запитів.
- Міркувальні моделі (Reasoning LLMs, наприклад, DeepSeek-R1): Мають вбудовану здатність до генерації внутрішніх логічних ланцюжків (Chain-of-Thought) перед видачею фінального результату. Під час інференсу такі моделі створюють послідовність прихованих міркувань, у межах яких здійснюється покрокова декомпозиція складного аналітичного завдання, прораховуються варіанти побудови складних вкладених

запитів та мінімізуються логічні колізії, що дозволяє успішно виконувати багатовимірні порівняльні зрізи даних.

1.3 Архітектури розгортання штучного інтелекту в освітніх системах

Інтеграція модулів штучного інтелекту, зокрема великих мовних моделей, у корпоративні інформаційні системи закладів вищої освіти вимагає системного підходу до проєктування архітектури розгортання. Обрана парадигма безпосередньо визначає не лише продуктивність процесу генерації структурованих запитів, але й формує межі інформаційної безпеки, масштабованості та фінансової доцільності підтримки програмного комплексу впродовж усього життєвого циклу. У сучасній інженерній практиці виділяють дві фундаментальні, протилежні за своєю суттю концепції інтеграції лінгвістичних моделей, якими є використання зовнішніх хмарних сервісів через інтерфейси прикладного програмування та локальне розгортання на власному апаратному забезпеченні установи.

Хмарна архітектура розгортання базується на моделі надання штучного інтелекту як сервісу, за якої вхідні природномовні запити разом із метаданими реляційного сховища передаються через зашифровані мережеві протоколи на віддалені сервери сторонніх провайдерів. Серед переваг такого підходу виділяють безперешкодний доступ до надпотужних моделей-гігантів, архітектура яких побудована на базі суміші експертів, що забезпечує високий рівень лінгвістичного узагальнення та здатність до виконання багатокрокових логічних висновків без попереднього донавчання. Крім того, хмарна модель мінімізує стартові капітальні витрати установи на закупівлю спеціалізованих серверних графічних процесорів із великим обсягом відеопам'яті, перекладаючи завдання масштабування та балансування навантаження на зовнішню інфраструктуру. Водночас застосування виключно хмарних інструментів в умовах адміністрування навчального процесу супроводжується

критичними ризиками, основним серед яких є загроза компрометації та несанкційнованої ексфільтрації конфіденційних персональних даних здобувачів освіти, що прямо порушує вимоги чинного законодавства України у сфері захисту персональних даних та створює неприпустимі репутаційні ризики для закладу освіти . Додатковими деструктивними чинниками хмарної парадигми виступають мережеві затримки, що унеможливлюють гарантування фіксованого часу відгуку системи, а також непередбачувані операційні витрати, зумовлені динамічною тарифікацією за кожен оброблений токен контексту при одночасному зверненні великої кількості користувачів.

На противагу повністю хмарним сервісам, локальна архітектура передбачає розгортання та запуск мовної моделі безпосередньо в межах внутрішньої ізольованої корпоративної мережі установи. Реалізація зазначеного підходу стала можливою завдяки стрімкому розвитку методів алгоритмічного стиснення нейромережевих структур, зокрема квантування, яке дозволяє знизити розрядність ваг нейронної мережі з тридцятидвобітного чи шістнадцятибітного представлення з рухомою комою до компактніших цілочисельних форматів без критичної втрати семантичної точності інференсу. Використання відкритих рушіїв локального виконання, таких як платформа Ollama, дозволяє ефективно інтегрувати оптимізовані моделі в замкнені середовища. Головною перевагою локальної парадигми є забезпечення абсолютного суверенітету даних, оскільки вся інформація обробляється суворо всередині обчислювального контуру факультету, повністю виключаючи ймовірність зовнішніх витоків чи перехоплення лінгвістичних потоків. Економічна складова локального розгортання відрізняється високою автономністю, адже після первинних інвестицій в апаратну частину подальші операційні витрати зводяться лише до базового енергоспоживання, незалежно від інтенсивності, частоти та обсягів аналітичних запитів з боку користувачів системи.

Враховуючи специфіку предметної області електронного деканату, де безпека даних та конфіденційність академічних показників є пріоритетними

системними обмеженнями, але при цьому виникає необхідність забезпечення високої гнучкості розгортання, найбільш ефективним інженерним рішенням є побудова гібридної архітектури інформаційного комплексу. Спроєктована система підтримує динамічне перемикання режимів інференсу лінгвістичних потоків за допомогою спеціалізованих конфігураційних параметрів середовища. У межах цієї концепції на серверному рівні реалізовано оригінальну триконтурну топографію обробки вхідних запитів користувачів, яка дозволяє збалансувати обчислювальне навантаження та оптимізувати використання апаратних потужностей. Детальну структурну організацію обчислювальних вузлів та механіку маршрутизації лінгвістичних потоків між клієнтським периметром, ізольованим корпоративним контуром та хмарним середовищем наочно проілюстровано на рисунку 1.1.

Першим рівнем виступає спеціалізований нейромережевий модуль маршрутизації та класифікації інтенту, що функціонує на базі компактної моделі малої розмірності Qwen2.5. Цей модуль здійснює превентивний семантичний аналіз тексту та визначає категорію логічної складності запиту, диференціюючи його на простий обліковий або складний аналітичний.

У разі активації хмарного режиму обробки через системні змінні середовища, семантичний потік перенаправляється на зовнішній захищений контур прикладного інтерфейсу на базі високопродуктивної моделі Gemini, що дозволяє виконувати операції з високою лінгвістичною точністю та мінімальними локальними затримками. Якщо система налаштована на роботу в суворо локальному автономному контурі, диспетчер маршрутизації розподіляє завдання між двома внутрішніми виконавчими контурами під керуванням платформи Ollama. Запити, ідентифіковані як прості, лінійні або суворо довідкові, делегуються другому контуру на базі моделі Qwen-Coder, яка володіє високою детермінованістю синтаксису і оптимізована для миттєвої генерації базового SQL-коду вибірки за частки секунди.

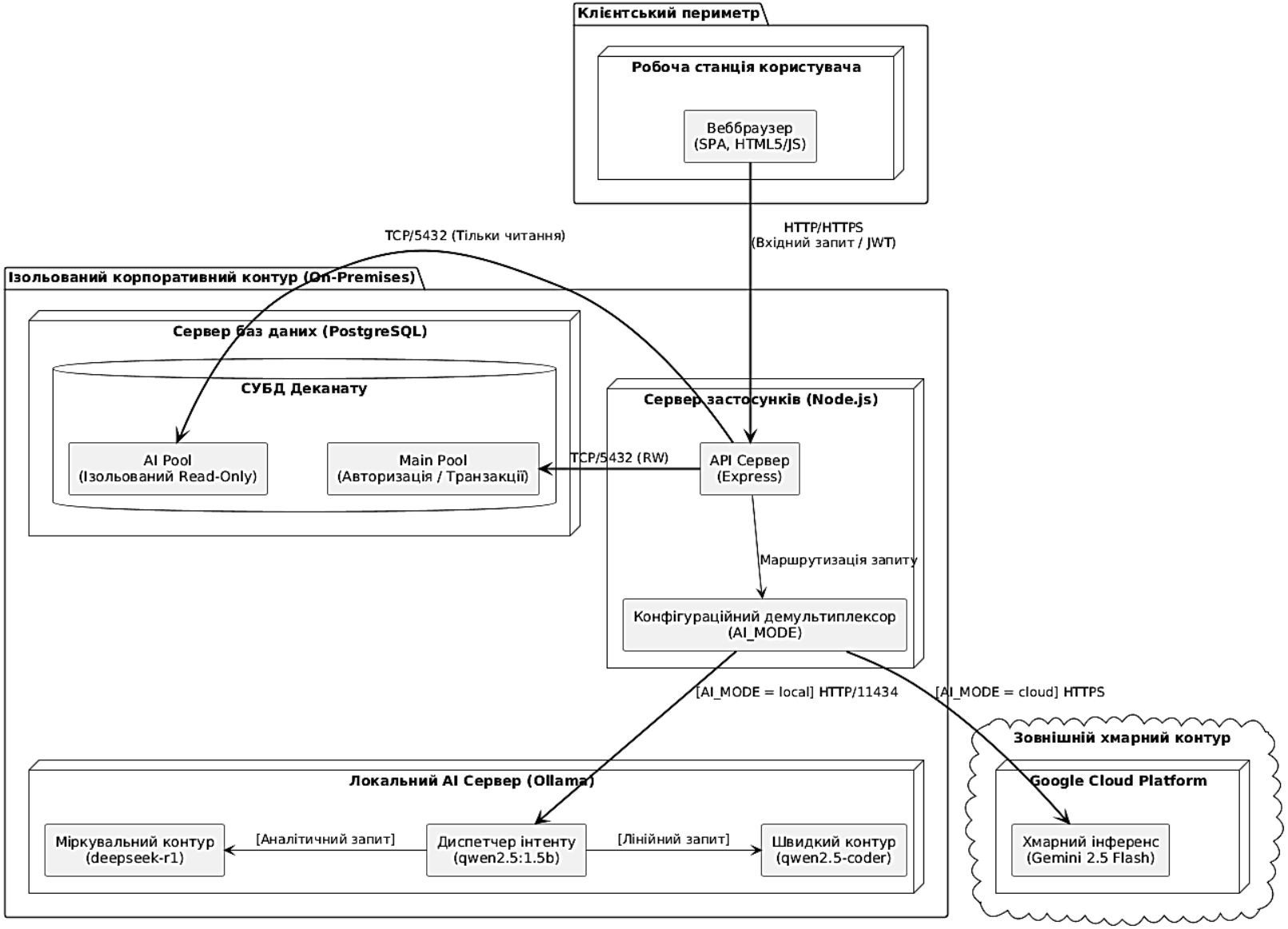


Рисунок *1.1 — Топологічна діаграма гібридної архітектури розгортання та маршрутизації інформаційних потоків*

У ситуаціях, коли користувач формулює складне аналітичне питання, що вимагає глибокого логічного виведення, порівняльного ранжування або багатотабличної агрегації даних, система автоматично перенаправляє семантичний потік на третій, важкий контур міркувальної моделі DeepSeek-R1. Зазначена модель використовує внутрішній цикл покрокових логічних міркувань, що дозволяє повністю нівелювати апаратні обмеження локальної інфраструктури, забезпечуючи максимальну точність детермінованої генерації без загрози компрометації конфіденційних даних установи.

1.4 Оптимізація контекстного навчання мовних моделей та шару аналітичної логіки

Використання великих мовних моделей в архітектурі інтелектуальних інформаційно-аналітичних систем ставить специфічні вимоги до організації та структурування вхідних лінгвістичних даних. Оскільки локальні моделі малої розмірності мають обмежену параметричну ємність порівняно з надвеликими хмарними рішеннями, точність їхнього інференсу та стійкість до синтаксичних збоїв критично залежать від якості організації контексту. Водночас навіть при використанні хмарних API оптимізація обсягу вхідних даних є необхідною для зниження мережевих затримок та мінімізації операційних витрат на тарифікацію токенів. У задачах автоматичної генерації структурованих запитів традиційне повномасштабне донавчання нейромережі на конкретному зрізі бази даних виявляється інженерно неефективним та фінансово обтяжливим. Будь-яка зміна фізичної схеми даних, зокрема додавання нових аналітичних колонок, таблиць чи реляційних зв'язків, вимагала б повторного ініціювання ресурсомісткого процесу навчання, що унеможливлює гнучке супроводження програмного комплексу.

Для подолання зазначеного обмеження в розробленій системі застосовано методологію навчання в контексті (In-Context Learning), за якої вся специфікація структури реляційного сховища передається безпосередньо

у системному промпті під час кожної транзакції. Механізм внутрішньої уваги в архітектурі Transformer дозволяє мовній моделі динамічно адаптуватися до переданих інструкцій і тимчасово фіксувати правила детермінованого відображення природної мови на мову структурованих запитів. Традиційним інструментом підвищення точності такого відображення виступає техніка Few-Shot Prompting, яка полягає у демонструванні моделі обмеженої кількості еталонних пар, що містять приклад природномовного запиту користувача та відповідний йому ідеальний SQL-код. Проте практична експлуатація міркувальних мовних моделей, зокрема контуру DeepSeek-R1, виявила суттєву архітектурну вразливість зазначеного методу, відому як ефект перенавчання на шаблонах. За надмірного лінгвістичного фокусування модель починає сприймати демонстраційні приклади занадто буквально, намагаючись скопіювати структуру шаблону навіть у тих випадках, коли семантика запиту користувача вимагає іншої математичної логіки. Це призводить до серйозних логічних колізій, наприклад, примусової підстановки жорстких умов пошуку лідера групи при спробі користувача отримати ширший список найкращих студентів.

З метою радикального вирішення проблеми перенавчання та суттєвого зниження когнітивного навантаження на обчислювальні контури штучного інтелекту, в межах даної роботи було спроєктовано та впроваджено архітектурний патерн інкапсуляції та делегування обчислювальної бізнес-логіки. Замість навантаження лінгвістичного контексту системного промпту складними інструкціями щодо правил побудови багатокрокових з'єднань, узагальнених табличних виразів та математичних функцій, усю важку аналітичну логіку ранжування та агрегації було повністю перенесено з нейромережевого модуля безпосередньо на реляційне ядро СУБД PostgreSQL.

Для реалізації цього підходу на рівні схеми бази даних було створено спеціалізовані аналітичні представлення (VIEW), які динамічно транслюються мовній моделі в секції системних інструкцій. Ключовим елементом виступає представлення рейтингів успішності студентів (v_student_ratings), усередині

якого на рівні СУБД інкапсульовано виконання складних віконних функцій ранжування за спаданням середнього бала, що дозволяє автоматично присвоювати кожному суб'єкту його точне місце в структурі академічної групи. Для розширеної аналітики факультету додатково впроваджено представлення середніх оцінок за дисциплінами (v_discipline_avg_grades) та агрегованої ефективності академічних груп (v_group_performance) . У системний промпт закладено критичне імперативне правило, яке зобов'язує штучний інтелект здійснювати пошук лідерів чи агрегованих метрик виключно через звернення до цих підготовлених представлень, повністю ігноруючи базові таблиці.

Завдяки впровадженню цього шару абстракції вищого рівня, когнітивна задача мовної моделі трансформувалася з багатокрокового проєктування графу з'єднань між багатьма ізольованими таблицями у генерацію елементарних лінійних вибірок. Отримуючи запит щодо пошуку найкращих студентів, модель більше не намагається самостійно конструювати складні обчислювальні ланцюги, а генерує простий оператор фільтрації за готовим полем рангу, що повністю нівелює ризик виникнення фантомних аліасів чи синтаксичних помилок і забезпечує стовідсоткову стабільність генерації коду.

1.5 Механізми забезпечення інформаційної безпеки та управління доступом

Впровадження інтелектуальних систем, що функціонують на основі трансляції природної мови у структуровані запити, вимагає кардинального переосмислення класичних підходів до забезпечення інформаційної безпеки та розмежування прав доступу. У традиційних корпоративних вебзастосунках архітектура безпеки та авторизації операцій традиційно реалізується на рівні проміжного програмного забезпечення сервера (Middleware) або за допомогою декларативного опису правил доступу до конкретних ендпоінтів прикладного програмного інтерфейсу. Проте в інформаційних системах, де єдиним

інтерфейсом взаємодії з базою даних виступає лінгвістичний контур великої мовної моделі, виникає специфічний клас семантичних загроз, який робить класичні методи розмежування привілеїв недостатніми. Найбільш критичними серед таких загроз в інженерній практиці визначено атаки типу семантичної ін'єкції (Prompt Injection) та несанкційного видобування конфіденційних масивів даних (Data Exfiltration). За умов, коли користувач має можливість вводити довільний неструктурований текст, існує висока ймовірність формування таких лінгвістичних конструкцій, які змусять модель проігнорувати закладені системні обмеження DDL-схеми та згенерувати SQL-код, орієнтований на отримання закритих персональних даних інших суб'єктів навчального процесу.

Для гарантування високого рівня захисту інформаційних активів університету в межах розробки було спроєктовано та впроваджено комплексну модель безпеки, побудовану за принципом ешалонованого семантичного та інфраструктурного захисту даних

(багаторівневого) захисту (Security by Design). Першим контуром безпеки виступає процедура автентифікації користувачів на основі криптографічних токенів JSON Web Tokens (JWT), що дозволяє серверним контролерам верифікувати поточну сесію та чітко ідентифікувати одну з трьох системних ролей користувача, якими є студент, викладач або працівник деканату. На основі розпізнаної ролі архітектура реалізує модель управління доступом (RBAC) шляхом динамічної превентивної ін'єкції обмежувальних інструкцій (Guardrails) безпосередньо у прихований системний промпт на етапі ініціалізації лінгвістичного інференсу, що блокує можливість компрометації даних ще до формування запиту до СКБД. Для студентської сесії бекенд-компонент вилучає унікальний ідентифікатор особи та примусово дописує до фінального промпту вимогу безальтернативного додавання умови просторової фільтрації за поточним значенням student_id у секції WHERE кожного згенерованого SQL-запиту, тоді як для викладачів та працівників деканату, що мають статус адміністратора, такі обмеження деактивуються .

Повну послідовність проходження інформаційного запиту крізь рівні ешелонованого семантичного та інфраструктурного захисту наочно відображено на рисунку 1.2.

Наступні рівні ешелону забезпечують превентивний захист на етапі алгоритмічної пост-обробки згенерованого моделлю результату до моменту його безпосереднього виконання в базі даних. Оскільки існує теоретична ймовірність успішного синтаксичного обходу промпт-інструкцій, у системі реалізовано програмний лексичний фаєрвол у вигляді функції isSafeSQL . Цей модуль здійснює посимвольний аналіз згенерованого SQL-рядка та превентивно блокує транзакцію у разі виявлення деструктивних ключових слів маніпулювання даними або структурою сховища, до яких віднесено оператори DROP, DELETE, UPDATE, INSERT, TRUNCATE, ALTER, CREATE, GRANT, REVOKE та REPLACE.

Завершальним, четвертим контуром ешалонованого захисту виступає фізична ізоляція аналітичних обчислень на рівні самої СУБД PostgreSQL. Запити від штучного інтелекту виконуються через спеціалізований пул з'єднань aiPool, конфігурований під окремим обліковим записом користувача бази даних, який володіє своро обмеженими привілеями, що діють виключно на читання інформації (Read-Only User) . Така комбінація динамічного семантичного екранування, лексичного аналізу тексту та інфраструктурного обмеження прав доступу повністю нівелює ризики горизонтального чи вертикального підвищення привілеїв, гарантуючи надійний захист конфіденційних активів установи.

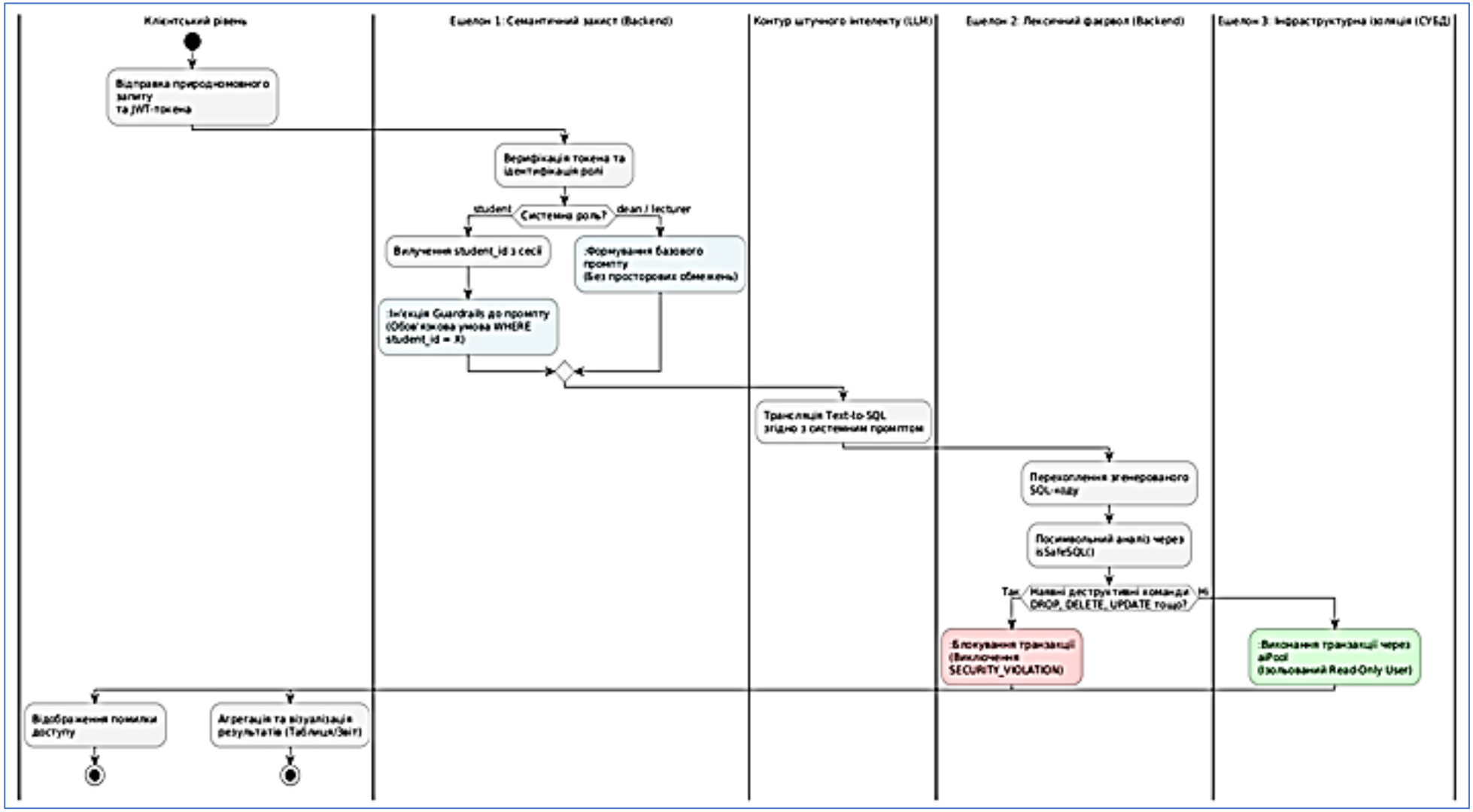


Рисунок *1.2 — Діаграма послідовності ешалонованого семантичного та інфраструктурного захисту даних*

1.6 Огляд та критичний аналіз існуючих програмних продуктів-аналогів

Дослідження ринку інформаційних технологій для сфери вищої освіти демонструє наявність низки програмних рішень, орієнтованих на автоматизацію обліку та управління навчальним процесом. Однією з найпоширеніших систем в українському освітньому просторі виступає автоматизована система управління «Деканат», яка забезпечує електронний супровід повного циклу документообігу та глибоко адаптована до актуальної нормативно-правової бази профільного міністерства. Незважаючи на ефективність в управлінні стандартизованими реєстрами та формуванні наказів, зазначена система має жорстко закриту реляційну архітектуру. Аналітичний функціонал комплексу де-факто обмежується набором статичних шаблонів звітів, що унеможливлює динамічне генерування нестандартних інформаційних вибірок. Для здійснення глибокого кореляційного аналізу користувач змушений експортувати сирі масиви інформації у формати електронних таблиць для подальшої ручної обробки сторонніми програмними засобами. Відсутність інтегрованого інтерфейсу обробки природної мови перетворює систему на інструмент виключно для вузькопрофільних операторів баз даних, суттєво обмежуючи її аналітичну цінність для керівного складу факультету.

Аналіз систем макрорівня, зокрема Єдиної державної електронної бази з питань освіти (ЄДЕБО), вказує на їхню абсолютну централізацію, інтеграцію з демографічними реєстрами та надійний рівень криптографічного захисту. Проте архітектура таких масштабних державних баз даних не призначена для вирішення локальних мікрооперaційних завдань окремого деканату, таких як деталізований облік поточних модульних оцінок чи проведення багатовимірного аналізу успішності конкретних академічних груп. Для гнучких аналітичних запитів з боку адміністрації факультету система залишається закритою.

З іншого боку спектра знаходяться поширені системи управління навчанням (LMS), зокрема платформи на базі архітектури Moodle, які забезпечують детальний трекінг цифрової активності студентів та ведення локальних електронних журналів. Втім, їхня архітектура є суворо курсоцентричною, що робить процеси крос-курсової агрегації та видобування зведеної статистичної звітності по цілих спеціальностях чи факультетах вкрай складними та такими, що неминуче вимагають втручання IT-спеціалістів на рівні прямих SQL-запитів.

Альтернативним інженерним підходом до побудови аналітичної звітності в освіті є застосування класичних систем бізнес-аналітики (Business Intelligence), таких як Tableau або Microsoft Power BI. Зазначені універсальні платформи надають надзвичайно широкі інструментальні можливості для побудови інтерактивних дашбордів та здійснення багатокритеріальної крос-фільтрації великих обсягів даних. Однак їх впровадження в повсякденний освітній процес супроводжується невиправдано високим порогом входження, оскільки формування аналітичних панелей вимагає від пересічних співробітників деканату впевненого володіння спеціалізованими мовами запитів та виразів аналізу даних. Крім того, розгортання та ліцензування закритих комерційних систем бізнес-аналітики є економічно обтяжливим для бюджетів державних закладів освіти.

Підсумовуючи результати проведеного порівняльного аналізу, встановлено, що на сучасному ринку програмного забезпечення відсутнє комплексне рішення, яке б органічно поєднувало строгу архітектуру збереження академічних даних із гнучкістю та інтуїтивною зрозумілістю природномовного інтерфейсу за технологією Text-to-SQL. Виявлена прогалина підтверджує науково-практичну актуальність розробки аналітичної системи, здатної автоматизувати складне видобування даних за допомогою гібридних контурів штучного інтелекту, впроваджуючи при цьому багаторівневий семантичний захист (RBAC), що усуває ризики компрометації конфіденційної інформації освітньої установи.

1.7 Постановка завдання

На основі проведеного розлогого системного аналізу предметної області, теоретичних засад функціонування великих мовних моделей архітектури Transformer та критичного огляду існуючих програмних продуктів-аналогів було сформульовано головну інженерну мету кваліфікаційної роботи. Ця мета полягає у проєктуванні, програмній реалізації та комплексному впровадженні інтелектуальної інформаційно-аналітичної системи електронного деканату, яка забезпечує безшовну, безпечну та інтуїтивно зрозумілу взаємодію користувачів із глибоко нормалізованим реляційним сховищем даних через інтерфейс природної мови на базі технології Text-to-SQL.

Критичною архітектурною вимогою до проєктується комплексу є забезпечення гнучкості розгортання та суверенітету інформаційних активів установи. Для виконання цієї вимоги необхідно розробити гібридну систему інференсу лінгвістичних потоків, яка підтримує як автономну роботу в ізольованому локальному контурі без залучення сторонніх сервісів, так і можливість захищеного масштабування через хмарні інтерфейси прикладного програмування за допомогою спеціалізованих змінних оточення.

Для успішного досягнення поставленої інженерної мети в межах виконання кваліфікаційної роботи необхідно послідовно вирішити чітко визначений комплекс науково-практичних та програмних завдань. Першочерговим завданням є детальне проєктування логічної та фізичної структури реляційної бази даних під управлінням системи керування базами даних PostgreSQL. У межах цього етапу вимагається розробити суворо нормалізовану схему, що виключає появу аномалій оновлення чи дублювання інформації, та створити взаємопов'язані таблиці для зберігання повних відомостей про організаційну структуру факультету, контингент студентів, кадровий склад викладачів, співробітників деканату, навчальні плани, дисципліни та академічну успішність. Забезпечення цілісності та

консистентності даних має бути реалізоване на рівні обмежень зовнішніх ключів із налаштуванням каскадних механізмів оновлення та жорстких обмежень на видалення записів.

Наступним важливим етапом є розробка серверного ядра системи у вигляді асинхронного інтерфейсу прикладного програмування на базі платформи Node.js із використанням фреймворку Express. Серверний компонент повинен забезпечувати надійне керування диференційованими пулами з'єднань із реляційною базою даних, де магістральні процеси авторизації та обліку обслуговуються основним пулом, а згенеровані штучним інтелектом транзакції виконуються через фізично ізольований пул під окремим обліковим записом із правами виключно на читання інформації.

Інженерне завдання інтеграції штучного інтелекту передбачає впровадження алгоритму триконтурної інтелектуальної маршрутизації вхідних запитів користувачів. Необхідно розробити диспетчер інтенту, який здійснюватиме превентивний семантичний аналіз тексту та динамічно перенаправлятиме потік до високошвидкісного контуру програмістських моделей для простих вибірок або до важкого міркувального контуру для глибокого порівняльного ранжування. Для радикального зниження когнітивного навантаження на обчислювальні моделі та усунення ефекту перенавчання на шаблонах необхідно реалізувати шар аналітичних представлень безпосередньо на рівні СУБД PostgreSQL.

Критичним безпековим завданням є практична реалізація ешелонованої системи управління доступом. Необхідно розробити та програмно вбудувати серверний алгоритм динамічної ін'єкції прихованих правил, який при виявленні сесії студента автоматично додає до системного промпту імперативну вимогу щодо превентивної фільтрації вибірки за його власним унікальним ідентифікатором. Для ролей співробітників деканату та викладачів архітектура повинна забезпечувати відповідні рівні адміністративного доступу. Крім того, вимагається створення програмного лексичного фаєрвола,

який здійснюватиме посимвольний аналіз згенерованого коду та превентивно блокуватиме будь-які деструктивні команди маніпулювання даними.

Завершальним етапом розробки є створення адаптивного клієнтського односторінкового вебзастосунку, який надає зручний інтерфейс для введення природномовних аналітичних запитів та динамічної візуалізації результатів. Також у межах розробки інтерфейсу необхідно реалізувати серверні сервіси детермінованого експорту агрегованих даних у формати PDF, Excel та Word із автоматичним форматуванням дат та генерацією людино-зрозумілих імен файлів, забезпечуючи високу промислову готовність усього програмного комплексу.

1.8 Висновки до розділу 1

- Обґрунтовано впровадження концепції Text-to-SQL в освітнє середовище для подолання технічного бар'єра між реляційними даними та керівництвом факультету, що нівелює обмеженість статичних шаблонів звітності традиційних систем обліку.
- Визначено доцільність застосування гібридної архітектури інференсу великих мовних моделей на базі механізму Transformer, яка забезпечує конфіденційність локальної обробки критичних даних освітньої установи та зберігає можливість хмарного масштабування.
- Проведено аналіз програмних аналогів (ЄДЕБО, АСУ «Деканат», систем управління та BI-платформ), який підтвердив відсутність на ринку рішень, що поєднують інфраструктурну безпеку з інтуїтивним локальним природномовним інтерфейсом.
- Сформульовано завдання на проєктування системи, що включає розробку нормалізованої бази даних PostgreSQL з аналітичними представленнями (VIEW), асинхронного Node.js-бекенду з розділеними за рівнем привілеїв пулами з'єднань та інтеграцію триконтурного алгоритму маршрутизації лінгвістичних потоків.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

2.1 Аналіз функціональних та нефункціональних вимог

2.1.1 Декомпозиція та специфікація функціональних вимог за рольовою моделлю

Проєктний етап розробки інтелектуальної інформаційно-аналітичної системи передбачає проведення детального системного аналізу та формалізації функціональних і нефункціональних вимог до програмного комплексу. Специфіка предметної області електронного деканату, що характеризується суворим розподілом адміністративних обов'язків та оперуванням великими обсягами конфіденційної інформації, вимагає точного визначення меж дозволених операцій для кожної категорії користувачів. Процес формалізації функціональних вимог орієнтований на забезпечення безшовної взаємодії суб’єктів із реляційним сховищем через інтерфейс природної мови за технологією Text-to-SQL, де бізнес-логіка системи адаптується під три основні користувацькі ролі, якими є студент (здобувач освіти), викладач (лектор) та працівник деканату (адміністратор, куратор або аналітик факультету).

Для категорії працівників деканату закладено розширений комплекс функціональних можливостей, спрямованих на автоматизацію макроуправління освітнім процесом факультету. Система забезпечує зазначеному контуру користувачів можливість виконання повнотекстових крос-табуляційних аналітичних запитів по всьому масиву бази даних без будь-яких просторових обмежень на рівні рядків. До базових функціональних обов’язків серверної частини для цієї ролі віднесено обробку природномовних звернень щодо формування зведених відомостей успішності, розрахунку глобальних та локальних рейтингів студентів у межах окремих кафедр, спеціальностей чи конкретних академічних груп. Працівник деканату має

можливість оперативно виявляти академічні аномалії, зокрема відстежувати динаміку середнього бала з певних дисциплін, визначати кращих студентів за результатами екзаменаційних сесій та аналізувати розподіл навантаження. Крім того, критичною функціональною вимогою для цієї ролі є наявність інструментарію детермінованого експорту отриманих аналітичних результатів у формати PDF, Excel та Word для подальшого використання в офіційному документообігу закладу вищої освіти.

Функціональний контур для ролі викладача спроєктовано з метою забезпечення оперативного моніторингу та аналізу успішності студентів із закріплених за ним дисциплін. Викладач наділений правами адміністративного доступу в межах аналітичного пулу, що дозволяє йому формувати запити природною мовою без суворих просторових обмежень рядків, проте його функціональний фокус зміщено на аналіз ефективності викладання предметів та оцінку зрізу знань академічних груп. Ця роль передбачає можливість отримання статистичних даних щодо середнього бала груп з конкретних курсів, порівняння динаміки успішності між поточними семестрами та формування списків студентів, які потребують додаткового академічного контролю, з подальшим вивантаженням результатів через модулі крос-платформної звітності.

Для ролі виступаючого у системі студента функціональний контур було спроєктовано з урахуванням суворих обмежень, що випливають із принципу найменших привілеїв та захисту персональних даних. Здобувач освіти має можливість взаємодіяти з аналітичним інтерфейсом виключно в межах інформації, яка безпосередньо стосується його особистих показників навчання. Функціональні вимоги передбачають обробку природномовних запитів студента щодо отримання повної історії власних оцінок за поточний або минулі семестри, перегляду переліку закріплених дисциплін та викладачів, а також визначення власного поточного місця у рейтингу своєї академічної групи. При цьому система повністю блокує будь-які спроби отримання деталізованої інформації про оцінки інших студентів або зведені показники

факультету, переводячи інтерфейс у режим захищеного семантичного екранування. Існуюча діаграма прецедентів системи відображає базові вектори взаємодії користувачів, проте впровадження ролі викладача розширює адміністративний контур безмежного виконання аналітичних запитів.

2.1.2 Формалізація нефункціональних вимог до архітектури та безпеки комплексу

Визначення нефункціональних вимог задає технологічні, експлуатаційні та архітектурні межі функціонування розробленого програмного комплексу. Головною нефункціональною вимогою до системи, що обумовлена вибором гібридної архітектури розгортання, є забезпечення гнучкого та ізольованого обчислювального середовища, здатного функціонувати в межах локального безпечного контуру установи за відсутності підключення до зовнішніх мереж, але з можливістю прозорого масштабування через захищені хмарні інтерфейси при наявності відповідних конфігураційних параметрів середовища. Усі операції з токенізації тексту, лінгвістичного аналізу, трансляції Text-to-SQL та безпосереднього виконання запитів на рівні реляційного ядра PostgreSQL повинні виконуватися з дотриманням суворих вимог до безпеки, гарантуючи нульову ймовірність несанкційного витоку конфіденційної інформації.

До параметрів швидкодії та продуктивності системи висунуто вимоги, адаптовані під специфіку локального інференсу великих мовних моделей на споживчому апаратному забезпеченні середовища. Час генерації валідного SQL-коду для лінійних облікових запитів при використанні швидкого локального контуру моделей не повинен перевищувати критичного порогу у дві секунди, що вимагає глибокої оптимізації контексту та застосування методів квантування ваг нейромережі. Для складних аналітичних запитів, що маршрутизуються на важкий міркувальний контур, допускається збільшення часового інтервалу обробки, необхідного для виконання внутрішнього циклу покрокових логічних висновків моделі (Chain-of-Thought), проте цей процес

обов'язково має супроводжуватися асинхронним візуальним інформуванням користувача про поточний стан обчислень через чат-інтерфейс.

Окремою нефункціональною вимогою виступає стійкість системи до мовної варіативності, орфографічних похибок та специфіки локалізації вхідних даних. Програмний комплекс повинен коректно опрацьовувати запити українською мовою, враховуючи особливості відмінювання прізвищ, назв дисциплін та абревіатур кафедр. На рівні серверної логіки має бути реалізована висока стійкість до синтаксичних помилок користувача, регістронезалежність пошукових операторів СУБД та автоматичне розпізнавання кириличних назв академічних груп із обов'язковим переведенням їх у латинські еквіваленти. Системний інтерфейс повинен мати високий рівень доступності, кросплатформеності та забезпечувати прозоре відображення відлагоджувальної інформації, зокрема згенерованого SQL-коду, для адміністраторів системи. Загальний склад прецедентів взаємодії користувачів із інтелектуальним ядром інформаційно-аналітичного комплексу, що деталізує ідентичність адміністративних повноважень для ролей викладачів і співробітників деканату, а також відображає суворі семантичні обмеження для ролі здобувача освіти, унаочнено на рисунку 2.1 .

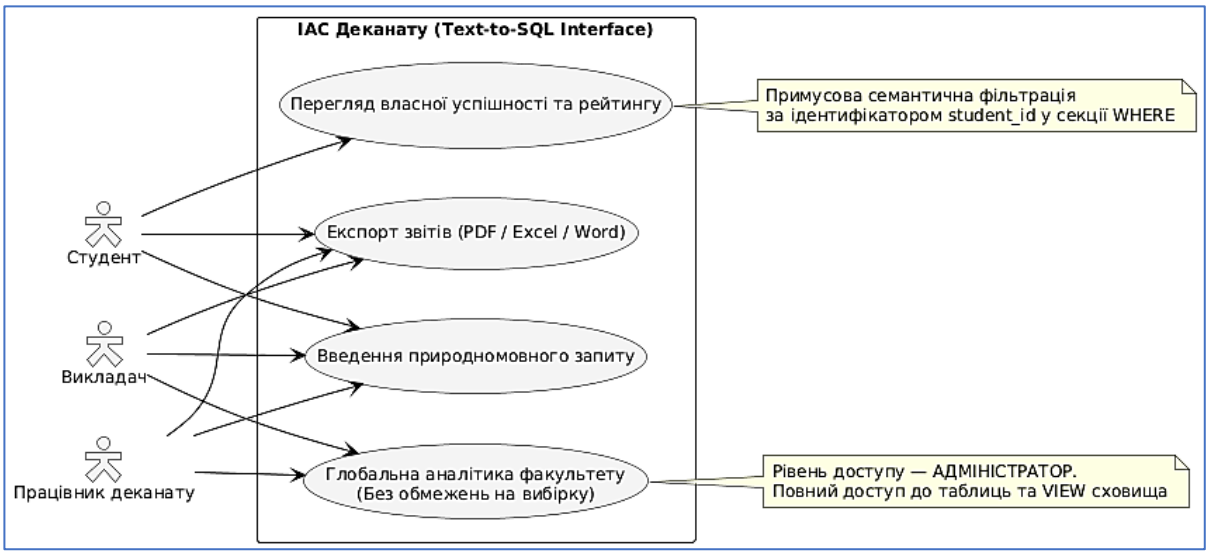


Рисунок *2.1 — Діаграма прецедентів інформаційно-аналітичної системи деканату*

2.2 Обґрунтування вибору технологічного стеку та архітектури розгортання

Важливим етапом проєктування інтелектуальної інформаційно-аналітичної системи є всебічне технічне обґрунтування вибору компонентів програмного стеку та архітектури їх взаємодії. Зважаючи на безкомпромісні вимоги щодо забезпечення абсолютної конфіденційності персональних даних здобувачів освіти та водночас необхідності забезпечення високої продуктивності при масштабуванні, загальна топографія комплексу була спроєктована за гнучкою гібридною клієнт-серверною парадигмою. Таке рішення дозволяє інкапсулювати критичну бізнес-логіку, лінгвістичний аналіз та реляційне сховище всередині внутрішньої інфраструктури установи (On-Premises), проте зберігає технологічну можливість захищеного підключення зовнішніх хмарних інтерфейсів прикладного програмування для виконання загальних знеособлених обчислень.

Для реалізації серверної частини системи (бекенду) було обрано програмну платформу Node.js у поєднанні з мінімалістичним фреймворком Express. Вибір зазначеного середовища виконання зумовлений його асинхронною архітектурою, що базується на циклі обробки подій та неблокуючому введенні-виведенні інформації. В умовах інтеграції мовних моделей здатність сервера ефективно обробляти велику кількість паралельних асинхронних запитів є критичним показником швидкодії. Фреймворк Express надає гнучкі можливості для розробки проміжного програмного забезпечення для верифікації криптографічних токенів безпеки (JWT) та організації безшовного потокового передавання даних між рушієм штучного інтелекту та клієнтським інтерфейсом.

Центральним елементом шару зберігання та персистентності даних було визначено реляційну систему керування базами даних PostgreSQL. Її вибір обґрунтований повною відповідністю критеріям ACID, підтримкою складної реляційної алгебри та високою продуктивністю при опрацюванні глибоко

нормалізованих схем даних. Оскільки архітектурне ядро проєктованого комплексу передбачає повне винесення важких віконних функцій ранжування успішності з промпту мовної моделі безпосередньо у реляційне ядро, СУБД повинна забезпечувати миттєву компіляцію аналітичних представлень (VIEW), з чим PostgreSQL справляється на промисловому рівні . Додатковою перевагою є наявність ефективних механізмів управління ізольованими пулами з'єднань, що мінімізує витрати ресурсів на повторне відкриття сесій.

Для розгортання та адміністрування локального контуру великих мовних моделей штучного інтелекту було обрано відкрите програмне середовище Ollama, яке абстрагує низькорівневу взаємодію з апаратними прискорювачами та надає стандартизований внутрішній REST API для взаємодії з Node.js бекендом . Використання Ollama дозволяє розгортати моделі, стиснені за допомогою передових методів квантування, що робить можливим виконання інференсу великих нейромереж на доступному споживчому обладнанні факультету.

Ключовим інноваційним рішенням у межах обґрунтування архітектури стало впровадження триконтурного механізму маршрутизації лінгвістичних потоків. Замість використання однієї універсальної моделі, на серверному рівні було реалізовано паттерн інтелектуального роутингу. У локальному режимі серверна частина аналізує вхідне питання за допомогою диспетчера інтенту. Якщо запит стосується простих списків чи довідок, він делегується спеціалізованій програмістській моделі Qwen-Coder для миттєвої генерації коду . Якщо ж запит потребує інтелектуального аналізу рейтингів, система прозоро перенаправляє потік на міркувальний контур DeepSeek-R1 . У разі активації гібридного режиму через змінні середовища, ці потоки можуть бути перенаправлені на високопродуктивний хмарний API Gemini. Детальну структурну організацію цих компонентів відображено на рисунку 2.2.

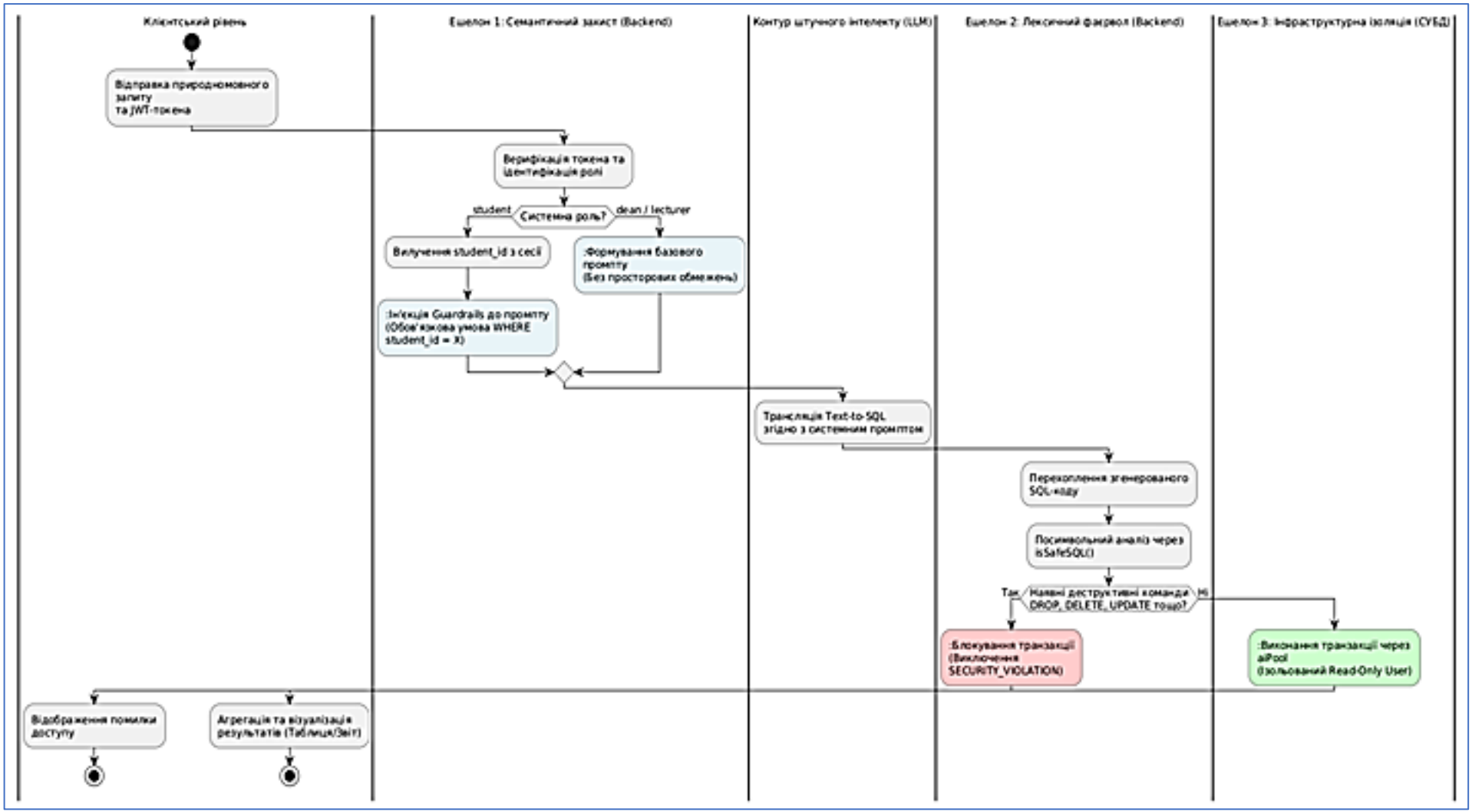


Рисунок *2.2 – Діаграма послідовностей гібридної архітектури розгортання та інформаційних потоків систем*

Поєднання Node.js, PostgreSQL, Ollama, хмарних інтерфейсів та триконтурної маршрутизації формує стійкий, високопродуктивний та гнучкий технологічний фундамент для реалізації інтелектуального електронного деканату.

2.3 Проєктування реляційної моделі бази даних

2.3.1 Логічна структура та граф зв'язків сутностей сховища даних

Проєктування інформаційно-аналітичної системи електронного деканату вимагає створення стійкої, гнучкої та суворо нормалізованої реляційної моделі даних, яка здатна безпомилково відображати всі процеси та взаємозв'язані сутності всередині вищого навчального закладу. Логічна структура сховища даних проєктувалася з орієнтацією на досягнення третьої нормальної форми (3NF), що дозволило повністю ліквідувати надлишковість інформації, мінімізувати обсяги збережених даних на локальному сервері та превентивно унеможливити виникнення аномалій вставки, оновлення та видалення записів. Центральним завданням цього етапу стало формування збалансованого графу зв'язків, який би дозволяв реляційному ядру СУБД PostgreSQL миттєво обчислювати складні аналітичні зрізи, а великій мовній моделі — безпомилково відстежувати шляхи з'єднання таблиць при Text-to-SQL трансляції.

Організаційна ієрархія закладу вищої освіти відображена у структурі сховища за допомогою послідовного каскаду зв'язків типу «один до багатьох». Головною сутністю верхнього рівня виступає таблиця факультетів (faculties), яка зв'язується з таблицею кафедр (departments) за допомогою ідентифікатора факультету. Кафедри, у свою чергу, перебувають у реляційних відносинах із таблицею спеціальностей (specialties), фіксуючи належність конкретних напрямів підготовки до профільних структурних підрозділів. Академічні групи (groups) інтегруються в загальний граф через зовнішній ключ, що вказує на відповідну спеціальність, забезпечуючи чітке групування контингенту за

роками початку навчання та освітніми рівнями. Персональні картки здобувачів освіти зосереджені в таблиці студентів (students), де кожен запис суворо підпорядковується конкретній академічній групі за допомогою зовнішнього ключа. Така послідовна декомпозиція дозволяє мовній моделі легко реконструювати топологію бази даних при виконанні крос-табуляційних запитів.

Окремий складний підграф формується навколо навчальних планів, кадрового складу та фіксації успішності, де реалізовано множинні відносини. Опис викладацького складу факультету забезпечується таблицею лекторів (lecturers), яка через зовнішній ключ підпорядкована відповідній кафедрі. Навчальні дисципліни акумулюються у профільній таблиці disciplines та також жорстко прив'язуються до кафедр, що забезпечують їх викладання. Для зв'язування академічних груп, навчальних дисциплін та конкретних лекторів у межах певного навчального періоду було спроєктовано проміжну таблицю навчальних планів (study_plans). Цей вузол виступає головним координатором освітнього процесу, фіксуючи семестр та поточний академічний рік для кожної пари дисципліна-група.

Підсумкова успішність реалізована через таблицю оцінок (grades), яка перебуває у подійному підпорядкуванні, оскільки кожен запис про отримані бали жорстко посилається на конкретного студента та на відповідний елемент навчального плану. Крім того, таблиця оцінок зв'язана з таблицею співробітників деканату (dean_office_staff), що дозволяє логувати та аудіювати джерело внесення інформації, закриваючи повний цикл обліку академічної діяльності факультету. Безпека та розмежування сесій користувачів підтримуються окремою таблицею облікових записів (users), яка інтегрує логіку автентифікації та містить зовнішні ключі до таблиць студентів, співробітників деканату та викладачів для забезпечення коректного функціонування рольової моделі . Фізичний граф зв'язків між усіма

спроєктованими сутностями реляційного сховища наочно відображено на рисунку 2.3.



Рисунок *2.3 – ER-діаграма фізичної схеми бази даних системи «Електронний деканат»*

2.3.2 Фізична специфікація таблиць та забезпечення реляційної цілісності

Фізична реалізація спроєктованої реляційної моделі в середовищі СУБД PostgreSQL вимагала детальної специфікації типів даних для кожної колонки, налаштування первинних та зовнішніх ключів, а також визначення суворих системних обмежень на рівні ядра бази даних. Для забезпечення максимальної швидкодії індексації та оптимізації JOIN-операцій для всіх сурогатних первинних ключів (faculty_id, department_id, specialty_id, group_id, student_id, lecturer_id, staff_id, discipline_id, plan_id, grade_id, user_id) було обрано цілочисельний тип даних із автоматичним приростом. Використання унікальних числових ідентифікаторів у ролі первинних ключів суттєво прискорює пошук записів та формування вибірок порівняно з текстовими або глобально унікальними ідентифікаторами, що є критично важливим для локальних серверів з обмеженими апаратними ресурсами.

Таблиця факультетів містить унікальний числовий ідентифікатор, повну назву та коротку абревіатуру, для збереження яких використано символьні типи даних змінної довжини із обмеженням розмірності. Аналогічний набір текстових та числових полів закладено у фізичну структуру таблиць кафедр та спеціальностей, де додатково впроваджено символьне поле для збереження офіційного коду спеціальності згідно з державним класифікатором. Фізична специфікація таблиці академічних груп передбачає наявність текстового поля для назви групи (group_name), а також числових полів для фіксації року початку навчання (year_start) та освітнього рівня (education_level) здобувачів освіти. Особлива увага приділена таблиці студентів, де для збереження прізвища, імені та по батькові виділено окремі символьні текстові поля, що полегшує мовній моделі генерацію умов фільтрації за текстовими шаблонами. Дата народження студента (birth_date) описується суворим типом даних для календарних дат, а для фіксації статі (gender) використано фіксоване символьне поле. Корпоративна електронна пошта (email) та рік зарахування (enrollment_year) зберігаються у відповідних символьних та числових полях,

причому для пошти налаштовано обмеження унікальності для запобігання дублюванню профілів.

Кадровий контур системи представлений таблицями лекторів (lecturers) та співробітників деканату (dean_office_staff), структура полів яких повністю уніфікована з картками студентів у частині збереження персональних даних, проте доповнена специфічними символьними полями для збереження поточної посади (position), академічного звання або вченого ступеня (academic_title для лекторів) та контактного номера телефону (phone для співробітників деканату). Таблиця дисциплін, крім назви (name) та абревіатури (short_name), містить важливі аналітичні числові поля, якими є кількість кредитів ECTS (credits) та загальний обсяг годин (hours_total), а також логічне поле для фіксації форми підсумкового контролю is_exam.

Проміжна таблиця навчальних планів містить первинний ключ та набір із трьох зовнішніх ключів, що посилаються на групу, дисципліну та лектора, а також символьні та числові поля для опису поточного семестру (semester) та навчального року (academic_year). Найбільш навантажена таблиця оцінок містить первинний ключ, зовнішні ключі студента, плану та співробітника, що виставив бал, а також числове поле для стобального значення оцінки (grade_value), символьне поле для літерного еквіваленту ects_grade та точну дату отримання балу date_received. Таблиця користувачів users фіксує строгі текстові поля для збереження унікального логіну (username), криптографічного хвашу пароля (password_hash), а також рядкове поле переліку ролей (role), що приймає значення згідно з рольовою моделлю доступу .

Забезпечення абсолютної реляційної цілісності та консистентності даних в умовах активного використання системи реалізовано за допомогою декларативних обмежень СУБД PostgreSQL. Для всіх зовнішніх ключів сховища було активовано механізм каскадного оновлення (ON UPDATE CASCADE), що гарантує автоматичну синхронізацію залежних записів при зміні первинних ідентифікаторів у батьківських таблицях. Натомість для

операцій видалення записів було застосовано диференційований підхід: видалення батьківських організаційних структур або карток студентів супроводжується жорсткою забороною операції (ON DELETE RESTRICT) за наявності пов'язаних оцінок чи навчальних планів. Це повністю виключає ризик появи «сирітських» записів успішності та руйнування аналітичного контуру, забезпечуючи високу надійність та промислову стійкість бази даних інтелектуального деканату.

2.4 Проєктування аналітичного шару та оптимізація когнітивного навантаження на ШІ

2.4.1 Концепція перенесення бізнес-логіки на рівень реляційного ядра СУБД

Під час системного проєктування та експериментального тестування контурів штучного інтелекту було виявлено критичну інженерну проблему деградації точності трансляції Text-to-SQL при обробці складних багатовимірних аналітичних інтентів. У ситуаціях, коли кінцевий користувач формує запити, що вимагають розрахунку відносних академічних показників, обчислення середніх балів, динамічного ранжування або визначення лідерів успішності в розрізі груп та спеціальностей, пряма генерація коду за базовими фізичними таблицями демонструє високу схильність моделей до логічних галюцинацій. Спроби покласти на лінгвістичний контур завдання самостійного конструювання синтаксично чистого SQL-коду з використанням важких аналітичних агрегацій, вкладених підзапитів та множинних з'єднань таблиць призводили до виникнення регулярних синтаксичних збоїв. Найбільш поширеними деструктивними проявами таких збоїв виступали генерація фантомних табличних аліасів, некоректний розподіл полів в операторах групування GROUP BY, а також логічні конфлікти при іменуванні колонок, що призводило до безальтернативного відхилення запиту на рівні транзакційного ядра СУБД.

Для радикального усунення ризиків семантичної деградації та забезпечення високої стабільності інтерфейсу було розроблено оригінальну архітектурну концепцію, що базується на оптимізації когнітивного навантаження на нейромережевий контур шляхом повного делегування обчислювальної бізнес-логіки реляційному ядру СУБД PostgreSQL. Замість перевантаження контексту системного промпту надлишковими інструкціями та правилами побудови складних математичних функцій, уся аналітична логіка була превентивно інкапсульована безпосередньо у фізичну схему бази даних у вигляді віртуальних аналітичних представлень (VIEW). Впровадження цього проміжного шару абстракції дозволило перенести виконання ресурсомістких реляційних операцій на оптимізатор СУБД, який виконує їх на промисловому рівні продуктивності. У результаті когнітивна задача великої мовної моделі звузилася до генерації елементарних лінійних вибірок із застосуванням простих операторів фільтрації та обмеження, що повністю нівелювало ймовірність синтаксичних помилок.

2.4.2 Специфікація аналітичних представлень для обчислення рейтингів та успішності

У межах реалізації спроєктованого аналітичного шару на рівні реляційного ядра бази даних PostgreSQL було розроблено три базові віртуальні представлення, що повністю забезпечують потреби адміністрації факультету в оперативній та точній звітності. Центральним компонентом цієї підсистеми виступає аналітичне представлення рейтингів успішності студентів v_student_ratings, структура якого інкапсулює математичний алгоритм розрахунку середнього бала кожного здобувача освіти та його точного відносного місця всередині конкретного академічного колективу. Обчислення позиції студента реалізується за допомогою інтеграції вбудованої аналітичної віконної функції RANK() OVER, що здійснює партиціювання оброблюваного масиву даних за ідентифікатором групи та подальше сортування за спаданням обчисленої агрегації середнього значення оцінок.

Впровадження цього рішення дозволило виключити необхідність динамічного розрахунку математичних рангів на стороні штучного інтелекту, замінивши його детермінованим зверненням до підготовленого поля рейтингів.

Другим елементом аналітичного шару є представлення ефективності академічних груп v_group_performance, призначене для консолідації статистичних показників успішності в розрізі навчальних семестрів та академічних років. Цей модуль автоматично розраховує усереднені бали для цілих студентських колективів та фіксує точну чисельність активного контингенту, дозволяючи миттєво оцінювати загальну динаміку навчального процесу на макрорівні факультету. Для забезпечення деталізованого моніторингу якості викладання окремих навчальних курсів було спроєктовано представлення середніх оцінок за дисциплінами v_discipline_avg_grades, яке агрегує показники успішності студентів у межах конкретних курсів, семестрів та навчальних років, виводячи точні усереднені бали та обсяги залучених здобувачів освіти. Безпосередній DDL-скрипт створення магістрального аналітичного представлення рейтингів успішності на рівні ядра СУБД PostgreSQL наведено в лістингу 2.1.

Лістинг 2.1 – *Скрипт створення базового аналітичного представлення рейтингів студентів у СУБД PostgreSQL*

```
CREATE VIEW v_student_ratings AS
SELECT
    st.student_id,
    st.group_id,
    gr.group_name,
    st.last_name,
    st.first_name,
    st.middle_name,
    AVG(grd.grade_value) AS avg_grade,
    RANK() OVER (
```

```
            PARTITION BY st.group_id
            ORDER BY AVG(grd.grade_value) DESC, st.last_name
ASC
        ) AS student_rank
    FROM students st
    JOIN groups gr ON st.group_id = gr.group_id
    JOIN grades grd ON st.student_id = grd.student_id
    GROUP BY st.student_id, st.group_id, gr.group_name,
st.last_name, st.first_name, st.middle_name;
```

Практичне впровадження спроєктованого аналітичного шару принципово змінило механіку функціонування лінгвістичної трансляції. Під час отримання складного з аналітичної точки зору питання, наприклад, щодо виведення кількох найкращих студентів конкретної академічної групи, мовна модель повністю звільняється від необхідності побудови триланкових з'єднань таблиць та важких математичних групувань. Алгоритм штучного інтелекту здійснює пряме звернення до готового віртуального представлення v_student_ratings, накладаючи просте лінійне умова фільтрації за назвою групи та обмежуючи вибірку за ранг-полем за допомогою оператора менше або дорівнює заданому числу. Це дозволило досягти стовідсоткової стабільності генерації виконуваного коду, повністю виключити помилки інтерпретації реляційних параметрів схеми та забезпечити миттєве повернення результатів користувачу, підтверджуючи високу промислову готовність розробленої системи.

2.5 Проєктування механізмів семантичної безпеки та управління доступом (RBAC)

2.5.1 Семантичне екранування та динамічна ін'єкція правил рольового доступу

Важливим компонентом забезпечення надійності розроблюваного інформаційно-аналітичного комплексу є проєктування архітектурного контуру семантичної безпеки. Оскільки взаємодія з базою даних здійснюється опосередковано через генеративний інтерфейс великих мовних моделей, класичні методи обмеження доступу до конкретних таблиць на рівні користувача СКБД потребують надбудови на рівні серверної бізнес-логіки застосунку. Для цього в системі реалізовано модель рольового управління доступом (RBAC), інтегровану безпосередньо в процес формування системного промпту для рушія штучного інтелекту. Механізм безпеки активується в момент дешифрування криптографічного токена JWT, з якого серверний модуль вилучає ідентифікатор користувача та його поточну роль . На основі цих даних у модулі бізнес-логіки формується динамічна ізоляція контексту, яка передається моделі як імперативна інструкція вищого пріоритету (Guardrails) .

У разі ідентифікації поточної сесії як студентської, система примусово генерує суворий набір правил обмеження доступу на рівні рядків . До лінгвістичного контексту додається критична вказівка, яка зобов'язує штучний інтелект додавати безумовну фільтрацію за унікальним номером студента у секцію WHERE кожного згенерованого SQL-запиту . Це унеможливлює отримання здобувачем освіти відомостей про успішність або персональних даних інших студентів, замикаючи область видимості моделі виключно на показниках поточного суб'єкта. На противагу цьому, для ролей викладача та працівника деканату в системі спроєктовано об'єднаний адміністративний контур доступу . Серверний модуль ідентифікує ці ролі як керівні та присвоює їм статус адміністратора аналітичного простору, повністю вилучаючи будь-які

просторові обмеження на рівні рядків баз даних . Це дозволяє моделі вільно оперувати аналітичними представленнями та таблицями для розрахунку загальних рейтингів факультету, забезпечуючи ідентичність прав доступу для викладацького та адміністративного складу.

2.5.2 Лексична валідація та інфраструктурний захист реляційного ядра

Другим ешелоном розробленої архітектури безпеки виступає підсистема превентивного лексичного контролю згенерованого коду, яка страхує систему у разі потенційного обходу мовною моделлю закладених у системний промпт інструкцій. Для реалізації цього захисту на рівні сервера застосунків впроваджено спеціалізований програмний фаєрвол у вигляді функції лексичної валідації isSafeSQL . Цей модуль перехоплює сформований нейромережеєю текстовий рядок SQL-запиту до моменту його передачі на виконання до бази даних та здійснює посимвольний аналіз тексту на наявність операторів маніпулювання структурою (DDL) або даними (DML). Функція оперує розширеним масивом заборонених ключових слів, до якого включено інструкції вилучення, модифікації, створення та перепризначення прав доступу, зокрема оператори DROP, DELETE, UPDATE, INSERT, TRUNCATE, ALTER, CREATE, GRANT, REVOKE та REPLACE . У разі виявлення будь-якого з цих деструктивних токенів у структурі згенерованого коду, система негайно перериває обчислювальний потік, генерує виключення безпеки та повертає користувачеві повідомлення про відхилення операції.

Фінальним рубежем захисту реляційного сховища є фізична ізоляція аналітичних транзакцій штучного інтелекту на інфраструктурному рівні СУБД PostgreSQL. Усі SQL-запити, успішно валідовані лексичним фаєрволом, спрямовуються до бази даних виключно через ізольований пул з'єднань aiPool . Цей пул функціонує під окремим обліковим записом користувача бази даних, для якого на рівні прав доступу СКБД встановлено суворе обмеження Read-Only. Навіть за теоретичної умови успішного виконання комбінованої атаки

типу семантичної ін'єкції та обходу лексичного аналізатора, ядро PostgreSQL заблокує будь-яку спробу деструктивного впливу на цілісність таблиць, оскільки обліковий запис ШІ позбавлений технічних привілеїв на запис або модифікацію інформації. Такий комплексний підхід, що поєднує динамічну ін'єкцію правил рольового доступу, посимвольний лексичний контроль та інфраструктурну ізоляцію пулів, гарантує промисловий рівень безпеки та стійкості інформаційно-аналітичної системи деканату.

2.6 Висновки до розділу 2

1. Виконано системне проєктування комплексу та формалізовано параметри середовища. Розроблено трирівневу рольову структуру з адміністративним доступом для персоналу та семантичним бар'єром для студентів. Затверджено стек технологій (Node.js, Express, PostgreSQL, Ollama) для підтримки триконтурної інтелектуальної маршрутизації.
2. Створено нормалізовану (3НФ) реляційну модель бази даних із декларативним контролем цілісності. Впроваджено шар віртуальних аналітичних представлень (v_student_ratings, v_group_performance, v_discipline_avg_grades), що переносить складні обчислення на ядро СУБД та мінімізує когнітивне навантаження на мовні моделі.
3. Спроєктовано ешелоновану систему інфраструктурної та семантичної безпеки. Захист реалізовано на трьох рівнях: динамічна ін'єкція рольових обмежень до промпту, серверний лексичний фаєрвол isSafeSQL для блокування деструктивних команд та фізична ізоляція транзакцій ШІ через Read-Only пул з'єднань aiPool.

3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Реалізація серверної частини та інтеграція з базою даних

3.1.1 Модульна структура та конфігурація середовища виконання

Програмна реалізація серверного компонента інтелектуальної інформаційно-аналітичної системи електронного деканату базується на модульно-шаровій архітектурній парадигмі в середовищі виконання Node.js. Такий інженерний підхід дозволяє досягти високого рівня делікації відповідальності (Separation of Concerns) між окремими програмними вузлами, забезпечуючи низьку зв'язаність (Low Coupling) та високу міцність (High Cohesion) модулів. Базова структура бекенд-застосунку розділена на ізольовані шари маршрутизації (Routes), контролерів (Controllers), сервісів бізнес-логіки (Services) та шару безпосередньої взаємодії з персистентним сховищем (Репозиторії/Моделі). Центральним диспетчером, що ініціалізує роботу вебсервера та координує функціонування проміжного програмного забезпечення (Middleware), виступає головний файл точки входу застосунку.

Стабільність та безпека функціонування системи на етапі ініціалізації середовища забезпечується через строге винесення всіх чутливих інфраструктурних параметрів у зовнішні змінні оточення за допомогою модуля dotenv. Конфігураційний файл середовища містить параметри криптографічної стійкості токенів автентифікації, мережеві адреси та реквізити доступу до реляційної СКБД, параметри локальних портів для зв'язку з рушієм Ollama, а також системний прапорець AI_MODE, який визначає магістральний режим інференсу великих мовних моделей. Програмна архітектура сервера Express передбачає обов'язкове підключення захисних middleware-модулів, серед яких крос-доменне налаштування CORS для безпечної взаємодії з односторінковим клієнтським застосунком, парсери

об'єктів JSON для коректної десеріалізації вхідних HTTP-пакетів, а також модулі логування транзакцій та аналізу системного навантаження.

Шарова топологія сервера спроєктована таким чином, що лінгвістичні та аналітичні потоки даних проходять послідовну обробку. Шар маршрутизації перехоплює вхідний HTTP-запит, здійснює превентивну перевірку синтаксису URL та делегує обробку відповідному middleware безпеки для верифікації сесії користувача. Після успішної криптографічної перевірки запит передається до шару контролерів, який відповідає за асинхронне вилучення параметрів тіла запиту та координацію викликів специфікованих сервісів. Безпосередня бізнес-логіка, що охоплює аналіз інтенту, триконтурний роутинг штучного інтелекту, формування системного промпту, семантичне екранування та лексичну фільтрацію коду, повністю інкапсульована всередині сервісного шару, що дозволяє ізолювати обчислювальні процеси від механізмів транспортного протоколу Express та забезпечує можливість автономного тестування кожного компонента.

3.1.2 Організація пулів з'єднань СУБД та забезпечення транзакційної стійкості

Інтеграція серверної частини з реляційним сховищем даних PostgreSQL реалізована за допомогою офіційного драйвера pg (node-postgres), який підтримує асинхронну взаємодію на основі промісів (Promises) та забезпечує високу ефективність обробки запитів. З метою оптимізації споживання системних ресурсів локального сервера та запобігання накладним витратам на постійне відкриття та закриття фізичних TCP-сесій, архітектура шару даних базується на механізмі пулювання з'єднань. У межах реалізації концепції ешелонованого захисту на етапі ініціалізації підсистеми взаємодії з СКБД створюються два абсолютно незалежні, ізольовані пули підключень, які функціонують під різними обліковими записами бази даних та мають чітко диференційовані рівні системних привілеїв.

Перший пул з'єднань, визначений як mainPool, конфігурується для обслуговування стандартних транзакційних процесів вебзастосунку, до яких віднесено операції автентифікації користувачів, верифікацію криптографічних хэшів паролів, реєстрацію сесій та ведення системних журналів аудиту безпеки. Цей пул взаємодіє з СКБД від імені адміністративного користувача, який наділений повними привілеями на читання та запис інформації в межах строго визначених службових таблиць. Механіка роботи пулу передбачає динамічне масштабування кількості активних клієнтів залежно від поточного навантаження на систему, встановлення жорстких тайм-аутів на очікування вільного з'єднання та автоматичне вилучення неактивних сесій для запобігання витоку пам'яті та перевантаження процесорних потужностей сервера баз даних.

Другий, специфікований пул з'єднань aiPool, призначений виключно для виконання SQL-коду, згенерованого в процесі трансляції Text-to-SQL контурами штучного інтелекту. Цей пул функціонує під обліковим записом, для якого на рівні реляційного ядра PostgreSQL активовано суворе обмеження Read-Only, що повністю виключає можливість виконання операцій модифікації структури або маніпулювання даними. Програмна логіка взаємодії з aiPool реалізує паттерн безпечного виділення та обов'язкового гарантованого вивільнення клієнта (Client Release). При надходженні запиту сервер асинхронно запитує вільне з'єднання з пулу, передає валідований лексичним фаєрволом SQL-рядок на виконання до бази даних, отримує сирий масив результатів вибірки та за допомогою блоку finally у структурі обробки виключень try-catch безальтернативно повертає клієнта назад до пулу, запобігаючи деструктивному ефекту голодування з'єднань (Connection Starvation) та гарантуючи високу транзакційну стійкість системи під високим навантаженням.

Архітектурну схему взаємодії серверних модулів бізнес-логіки з реляційним сховищем даних через ізольовані пули з'єднань наочно відображено на рисунку 3.1.

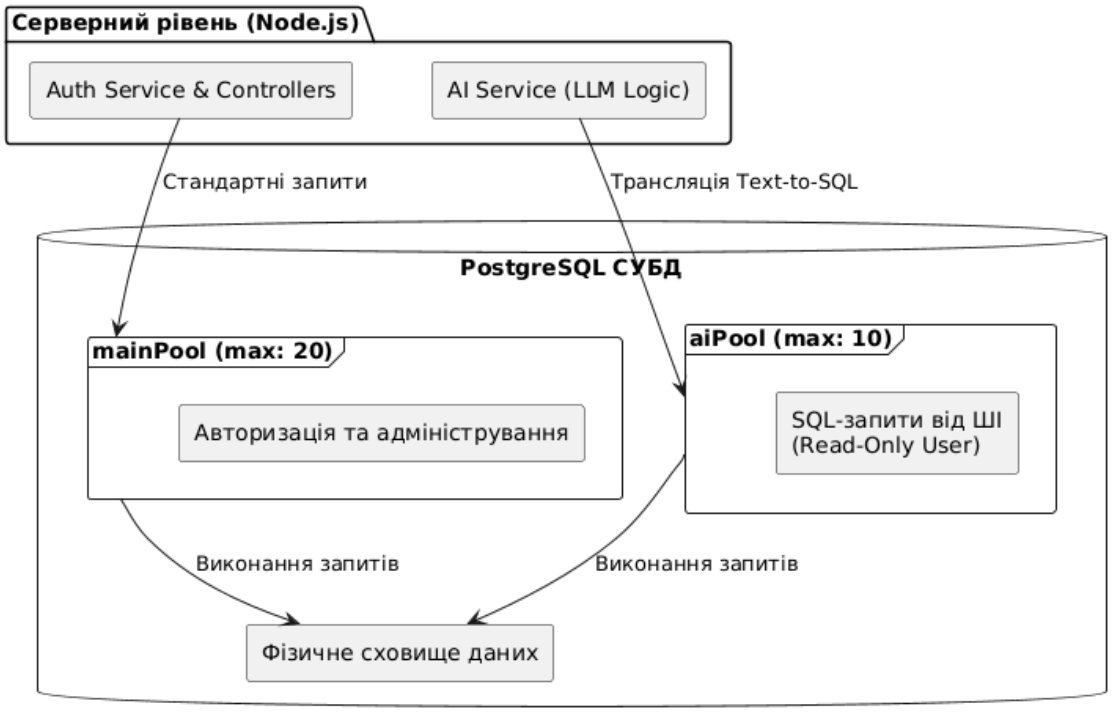


Рисунок *3.1 — Схема організації підключень до бази даних через розділені пули*

Така побудова системи гарантує, що навіть у разі високого навантаження на аналітичний сервер, основні функції деканату залишаються доступними, а сама база даних захищена від аномальних транзакцій за рахунок використання облікового запису з обмеженими правами.

3.2 Реалізація інтелектуального шару та триконтурної маршрутизації запитів

3.2.1 Алгоритм класифікації інтенту та диспетчеризації моделей

Програмна реалізація інтелектуального ядра інформаційно-аналітичної системи зосереджена у спеціалізованому сервісному модулі, який виконує функції оркестратора лінгвістичних потоків. Головним завданням цього компонента є прийом природномовного запиту від користувача, визначення його логічної складності та подальше делегування процесу трансляції Text-to-

SQL найбільш релевантній мовній моделі. Архітектура сервісу побудована на основі патерну стратегії, що дозволяє системі динамічно перемикатися між хмарним контуром генерації (через API моделі Gemini) та локальним середовищем виконання (через REST API платформи Ollama) залежно від конфігураційних параметрів сервера. Такий підхід забезпечує високу відмовостійкість: у разі відсутності підключення до глобальної мережі або свідомої активації режиму підвищеної конфіденційності, система безшовно продовжує функціонувати на внутрішніх апаратних потужностях.

Ключовим інноваційним рішенням на етапі програмної реалізації локального інференсу стала розробка алгоритму триконтурної маршрутизації на основі превентивного семантичного аналізу. При надходженні запиту в автономному режимі сервер не відправляє його одразу до найпотужнішої нейромережі, оскільки це призвело б до невиправданих витрат обчислювального часу та перевантаження графічного процесора. Натомість запит спершу проходить через легковиковий модуль класифікації інтенту. Цей програмний вузол здійснює лексичний аналіз тексту на наявність маркерів складної аналітики, якими виступають слова-тригери на кшталт «рейтинг», «найкращі», «середній бал», «порівняти» або «динаміка».

Якщо алгоритм ідентифікує запит як простий обліковий або довідковий (наприклад, пошук розкладу, списку викладачів кафедри або переліку дисциплін), потік даних маршрутизується до швидкого контуру на базі моделі Qwen-Coder. Завдяки оптимізації ваг під синтаксис мов програмування, ця модель генерує базовий SQL-код за частки секунди. У ситуаціях, коли класифікатор фіксує потребу в багатокритеріальному ранжуванні чи розрахунку відносних показників, система активує міркувальний контур на базі моделі DeepSeek-R1. Ця нейромережа ініціює внутрішній процес покрокового логічного виведення (Chain-of-Thought), що дозволяє їй методично декомпозувати складне аналітичне завдання, проаналізувати структуру підготовлених віртуальних представлень бази даних та сформувати безпомилковий реляційний запит. Застосування такого диспетчера інтенту

дозволило знайти оптимальний інженерний баланс між продуктивністю генерації та глибиною когнітивного аналізу.

3.2.2 Динамічна генерація контексту та інтеграція правил безпеки

Точність та безпека процесу трансляції природної мови у структурований код критично залежать від якості формування системного промпту — базового лінгвістичного контексту, який передається моделі перед основним запитом користувача. Програмна реалізація генератора промптів працює за принципом динамічного збирання інструкцій (In-Context Learning). На першому етапі скрипт завантажує до оперативної пам'яті фізичну структуру реляційного сховища, проте, замість передачі повної схеми всіх наявних таблиць, модулю штучного інтелекту надається доступ переважно до DDL-описів аналітичних представлень (VIEW), таких як v_student_ratings. Цей крок програмно закріплює концепцію перенесення бізнес-логіки на реляційне ядро СУБД, унеможливлюючи спроби моделі самостійно конструювати складні віконні функції. Додатково в контекст інтегрується блок Few-Shot прикладів, який містить еталонні пари трансформації тексту в код для адаптації нейромережі до специфіки локального синтаксису.

Найбільш критичним компонентом генератора системного промпту є блок динамічної ін'єкції правил рольового доступу (RBAC), який програмно реалізує перший ешелон семантичної безпеки системи. Алгоритм отримує дешифрований об'єкт поточного користувача із middleware-модуля авторизації та, залежно від визначеної ролі, формує жорсткі директиви для лінгвістичної моделі. Для ролей співробітників деканату та викладачів активується режим адміністративного доступу без просторових обмежень, тоді як для ролі студента генерується імперативна вимога щодо примусового накладання фільтрації за ідентифікатором.

Структурну схему функціонування серверного модуля конструювання системного промпту, що демонструє інтеграцію базових шаблонів із

динамічними правилами рольового доступу та запитом користувача, наведено на рисунку 3.2.

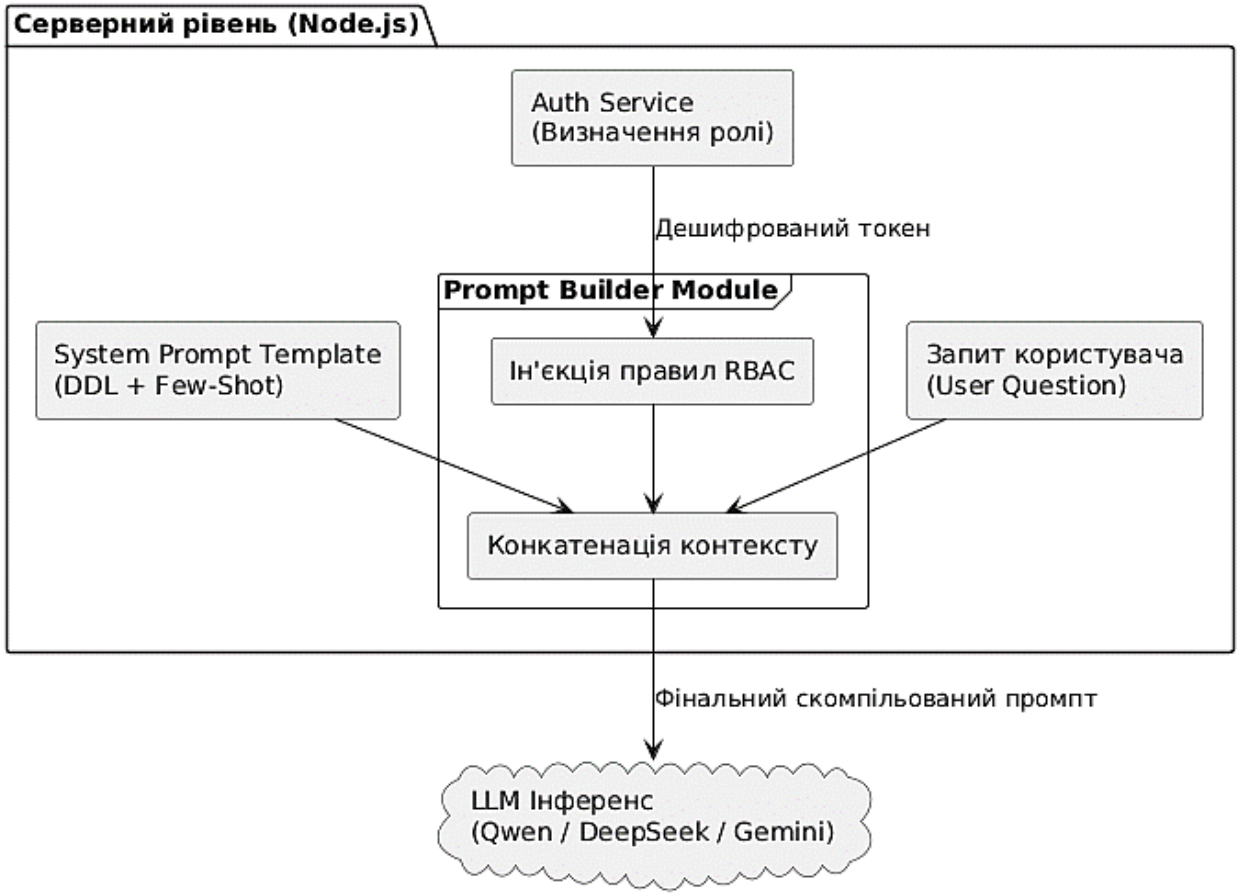


Рисунок *3.2 — Алгоритмічна схема конструювання системного промпту з інтегрованими правилами RBAC*

Програмну реалізацію цієї логіки маршрутизації привілеїв наведено в лістингу 3.1.

Лістинг 3.1 — *Програмна реалізація ін'єкції правил семантичного екранування (RBAC) до системного промпту*

```
let rbacRules = "";

if (user.role === 'student') {
```

```
        rbacRules = `УВАГА! Користувач є СТУДЕНТОМ з ID =
${user.studentId}.
    КРИТИЧНЕ ПРАВИЛО БЕЗПЕКИ: Ти ПОВИНЕН обмежити
результати запиту ТІЛЬКИ цим студентом.
    Завжди додавай умову WHERE st.student_id =
${user.studentId} до всіх запитів.
    Студент не має права бачити персональні дані або оцінки
інших студентів.`;
    } else if (user.role === 'dean' || user.role ===
'lecturer') {
        rbacRules = `Користувач є ПРАЦІВНИКОМ ДЕКАНАТУ або
ВИКЛАДАЧЕМ. Рівень доступу - АДМІНІСТРАТОР. Обмежень на
вибірку немає.`;
    }
```

Після об'єднання DDL-схем, рольових інструкцій безпеки, еталонних прикладів та безпосереднього запиту користувача, сформований текстовий пакет передається через API до обраної мовної моделі. Отримана відповідь проходить стадію програмної постобробки, під час якої за допомогою регулярних виразів із тексту вилучається чистий SQL-код, відсікаються будь-які маркдаун-теги форматування та коментарі моделі. Очищений рядок негайно передається на вхід лексичного фаєрвола isSafeSQL і лише після підтвердження його інфраструктурної безпеки відправляється на виконання до ізольованого пулу бази даних.

Для фіксації послідовності виконання операцій, від моменту превентивного аналізу інтенту до фінальної лексичної валідації та фізичного виконання запиту, загальну логіку роботи маршрутизатора схематично наведено на рисунку 3.3.

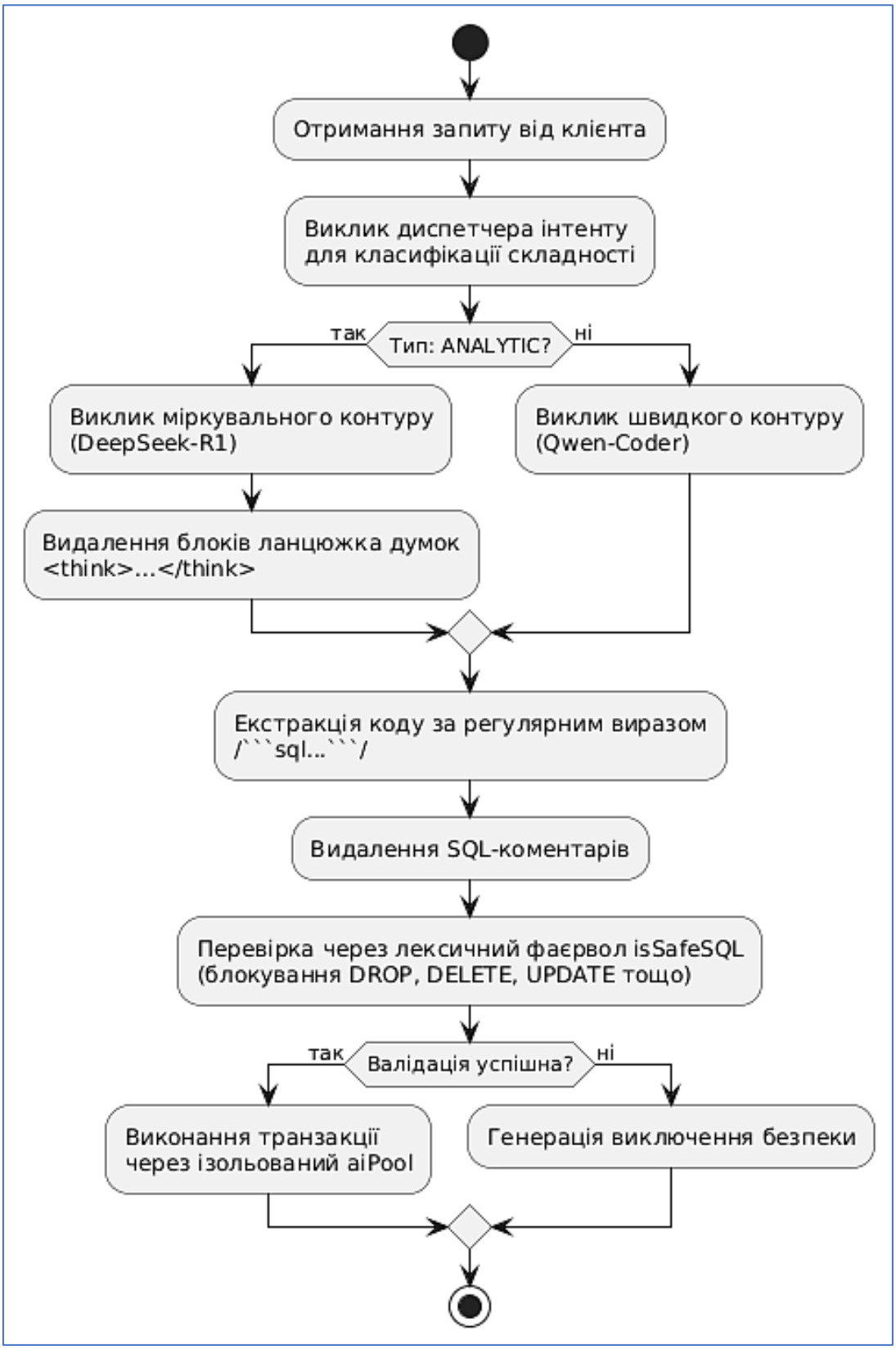


Рисунок *3.3 — Блок-схема алгоритму роботи модуля маршрутизації, парсингу та семантичної валідації*

3.3 Розробка клієнтського застосунку та інтеграція модулів експорту звітності

3.3.1 Архітектура клієнтського інтерфейсу та обробка асинхронних запитів

Клієнтська частина інформаційно-аналітичної системи деканату спроєктована за парадигмою односторінкового застосунку (Single Page Application). З метою мінімізації накладних витрат на завантаження програмних бібліотек та забезпечення максимальної швидкості рендерингу в браузері, архітектуру фронтенду реалізовано з використанням нативних вебтехнологій без залучення важких реактивних фреймворків. Взаємодія з серверним програмним інтерфейсом (API) здійснюється за допомогою асинхронних викликів на базі стандарту Fetch API, що дозволяє динамічно оновлювати вміст сторінки без її повного перезавантаження.

Організація інтерфейсу користувача підпорядкована концепції мінімалізму та функціональної ефективності. Головним елементом взаємодії виступає текстовий маніпулятор для введення природномовних запитів, який супроводжується панеллю візуалізації поточного статусу системи. Оскільки специфіка роботи міркувальних мовних моделей, зокрема контуру DeepSeek-R1, передбачає наявність затримок, пов'язаних із генерацією внутрішнього ланцюжка думок перед формуванням фінального SQL-коду, критичною вимогою до фронтенду стала реалізація механізмів асинхронного зворотного зв'язку. Інтерфейс автоматично блокує повторні відправлення запитів на період активного інференсу та відображає динамічні індикатори завантаження, інформуючи користувача про виконання складних аналітичних обчислень на сервері.

Після успішного отримання відповіді від бекенду, яка містить JSON-об'єкт із результатами вибірки та згенерованим SQL-кодом, клієнтський скрипт ініціює процес динамічної побудови об'єктної моделі документа (DOM). Система автоматично аналізує ключі отриманих масивів даних і

формує адаптивні HTML-таблиці, заголовки яких відповідають назвам колонок реляційного сховища або їхнім семантичним аліасам. Для ролей адміністратора та викладача інтерфейс додатково розгортає панель відлагодження, де виводиться чистий синтаксис сформованого базою даних запиту, що забезпечує прозорість аналітичних алгоритмів та дозволяє контролювати точність роботи штучного інтелекту.

3.3.2 Реалізація підсистеми експорту аналітичної звітності

Зважаючи на бюрократичну специфіку функціонування закладу вищої освіти, де будь-яка аналітична вибірка є потенційним проєктом офіційного документа, критичним функціональним вузлом системи є підсистема детермінованого експорту даних. Модуль експорту інкапсульовано на серверному рівні, що гарантує незалежність процесу формування файлів від обчислювальних потужностей клієнтського браузера та забезпечує сувору стандартизацію вихідних документів. Система підтримує генерацію звітності у трьох базових форматах: табличних процесорах Excel (XLSX), текстових документах Word (DOCX) та портативних документах PDF.

Механіка генерації звітів базується на перехопленні поточного стану аналітичної вибірки, сформованої мовною моделлю, та її подальшій серіалізації за допомогою спеціалізованих програмних бібліотек. При ініціації експорту в формат Excel серверний скрипт динамічно створює нову книгу, ітерує масив об'єктів та формує двовимірну матрицю значень. Важливим інженерним рішенням на цьому етапі є автоматичне визначення типів даних для кожної клітинки: числові ранги та середні бали експортуються як математичні значення, що дозволяє співробітникам деканату миттєво застосовувати до них стандартні формули без додаткової конвертації форматів. Для експорту в текстові формати (PDF та Word) застосовується алгоритм побудови структурованих табличних сіток із жорстко заданими параметрами відступів та автоматичним розбиттям на сторінки.

Кожен згенерований файл проходить процедуру маркування: система автоматично формує його назву на основі поточної дати, часу генерації та системної ролі користувача, що ініціював запит. Готовий бінарний потік даних (Buffer) передається через HTTP-відповідь із відповідними MIME-типами, змушуючи клієнтський браузер ініціювати процес завантаження файлу на локальний носій. Такий підхід повністю закриває цикл аналітичної роботи, дозволяючи перетворити результат природномовного запиту на готовий до друку та підписання офіційний документ.

3.4 Тестування системи

Етап тестування спроєктованого програмного комплексу передбачав перевірку стабільності функціонування архітектурних вузлів, верифікацію точності трансляції природномовних запитів та валідацію впроваджених механізмів семантичної і транзакційної безпеки. Методологія тестування базувалася на симуляції типових сценаріїв експлуатації системи цільовими групами користувачів. Першочерговим завданням виступала оцінка ефективності алгоритму триконтурної маршрутизації та здатності мовних моделей генерувати синтаксично валідний SQL-код. Під час обробки лінійних довідкових запитів диспетчер інтенту коректно перенаправляв потік до моделі Qwen-Coder, яка демонструвала високу швидкість генерації базового коду. При обробці складних аналітичних інтентів міркувальний контур DeepSeek-R1 здійснював звернення до підготовленого аналітичного представлення успішності замість самостійного конструювання віконних функцій. Такий підхід суттєво знизив відсоток синтаксичних збоїв та логічних галюцинацій (точність валідних відповідей склала близько 95%), хоча ймовірність виникнення помилок при формуванні нестандартних крайових запитів залишається природним обмеженням генеративних технологій.

Валідація системи управління доступом підтвердила коректність роботи алгоритму динамічної ін'єкції правил (RBAC). Ініціація сесії від імені студента

з подальшим формуванням запиту щодо оцінок інших здобувачів освіти продемонструвала, що модель інтегрує жорстку умову просторової фільтрації за ідентифікатором поточного користувача у секцію WHERE. Для ролей викладача та працівника деканату тестування підтвердило цільове зняття цих обмежень для вільного розрахунку загальнофакультетських метрик.

Перевірка механізмів захисту реляційного сховища від семантичних ін'єкцій підтвердила працездатність спроєктованої дворівневої інфраструктури безпеки. Перший рівень, програмний модуль isSafeSQL, здійснює посимвольний аналіз згенерованого коду та ідентифікує наявність заборонених ключових слів (наприклад, DROP або DELETE), блокуючи подальше виконання коду сервером Node.js. Другий рівень захисту діє як фізичний запобіжник: усі транзакції штучного інтелекту виконуються виключно через ізольований пул aiPool. Оскільки цей пул функціонує під обліковим записом бази даних із правами Read-Only, будь-яка теоретична спроба модифікації або видалення таблиць, що могла б оминути лексичний фаєрвол, автоматично відхиляється на рівні ядра PostgreSQL через нестачу системних привілеїв. Результати тестування засвідчили, що закладена інженерна конфігурація забезпечує достатній рівень відмовостійкості та безпеки для експлуатації комплексу у внутрішньому корпоративному середовищі навчального закладу.

3.5 Висновки до розділу 3

1. Здійснено програмну реалізацію серверного ядра на базі Node.js із впровадженням механізму розділених пулів з'єднань. Ізольований пул із правами виключно на читання (Read-Only) формує базовий інфраструктурний рубіж захисту реляційного сховища.
2. Реалізовано алгоритм триконтурної маршрутизації лінгвістичних потоків. Диспетчер інтенту делегує лінійні запити моделі Qwen-Coder, а складні багатовимірні обчислення — міркувальному контуру DeepSeek-

R1. Впроваджено модуль конструювання промптів із динамічною ін'єкцією правил рольового доступу (RBAC).

3. Розроблено клієнтський односторінковий застосунок (SPA) для асинхронної взаємодії з мовними моделями та динамічної генерації таблиць. Інтегровано серверну підсистему детермінованого експорту аналітичних вибірок у регламентовані формати PDF, XLSX та DOCX.
4. Проведено тестування, яке підтвердило мінімізацію синтаксичних галюцинацій завдяки делегуванню складних обчислень ядру СУБД через віртуальні представлення. Валідація дворівневого безпекового контуру довела абсолютну стійкість лексичного фаєрвола isSafeSQL та ізольованого пулу до спроб семантичних ін'єкцій та несанкціонованої модифікації даних.

ВИСНОВКИ

1. Вирішено науково-практичне завдання з проєктування інтелектуальної системи електронного деканату на базі технології Text-to-SQL. Використання мовних моделей архітектури Transformer подолало обмеження статичної звітності традиційних комплексів, забезпечивши керівництво факультету гнучким природномовним інтерфейсом.
2. Реалізовано гібридну архітектуру інференсу з алгоритмом триконтурної маршрутизації лінгвістичних потоків. Впроваджений диспетчер інтенту оптимізує споживання ресурсів сервера, делегуючи лінійні запити моделі Qwen-Coder, а складні багатовимірні обчислення — міркувальному контуру DeepSeek-R1.
3. Спроєктовано оптимізовану модель у СУБД PostgreSQL із шаром віртуальних представлень. Перенесення обчислювальної логіки та віконних функцій ранжування безпосередньо у реляційне ядро бази даних знизило когнітивне навантаження на ШІ, мінімізувало галюцинації та забезпечило точність вибірок на рівні 95%.
4. Впроваджено ешелоновану систему безпеки програмного комплексу. Захист реалізовано через семантичне екранування сесій (RBAC) у системному промпті, серверний лексичний фаєрвол isSafeSQL для блокування деструктивних команд та інфраструктурну ізоляцію транзакцій ШІ через Read-Only пул з'єднань aiPool.
5. Розроблена система має характер масштабованого ядра з високою практичною цінністю. Перспектива промислового деплою передбачає її інтеграцію у віртуальні підмережі (VPC), фільтрацію на рівні ОС (UFW/iptables), впровадження Rate Limiting та підключення WAF для захисту від DDoS-атак.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MacAfee A., Brynjolfsson E. Machine, Platform, Crowd: Harnessing Our Digital Future. W. W. Norton & Company, 2017. 416 p.
2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 p.
3. White T. Hadoop: The Definitive Guide. O'Reilly Media, 2015. 756 p.
4. Drake B. et al. Natural Language Processing with Python. O'Reilly Media, 2009. 504 p.
5. LangChain Documentation: Interacting with SQL Databases. URL: https://js.langchain.com/docs/use_cases/sql/ (дата звернення: 26.04.2026).
6. Obe R., Hsu L. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database. O'Reilly Media, 2017. 270 p.
7. Офіційна документація PostgreSQL 16. URL: https://www.postgresql.org/docs/16/index.html (дата звернення: 25.04.2026).
8. Офіційна документація Node.js. URL: https://nodejs.org/en/docs/ (дата звернення: 25.04.2026).
9. Mardan A. Express.js Guide: The Comprehensive Book on Express.js. Azat Mardan, 2014. 268 p.
10. Офіційна документація Ollama. URL: https://github.com/ollama/ollama (дата звернення: 25.04.2026).
11. Qwen 2.5-Coder Technical Report. Alibaba Cloud, 2024. URL: https://qwenlm.github.io/blog/qwen2.5-coder/ (дата звернення: 25.04.2026).
12. OWASP Top 10 Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 27.04.2026).
13. Shklar L., Rosen R. Web Application Architecture: Principles, Protocols, and Practices. 2nd ed. Wiley, 2009. 448 p.
14. Hanchett E., Listwon B. Vue.js in Action. Manning Publications, 2018. 304 p.
15. Офіційна документація Nuxt 3. URL: https://nuxt.com/docs (дата звернення: 26.04.2026).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Сидоренко Сергій Сергійович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми «Програмне забезпечення систем», адреса електронної пошти erestion@gmail.com, — підтверджую, що написана мною кваліфікаційна робота на тему «Інформаційна система обліку успішності студентів з природномовним інтерфейсом» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

-згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 12.06.2026 _____ Сидоренко Сергій Сергійович
(студент)

Дата 12.06.2026 _____ Безверхий Анатолій Ігорович
(керівник)