Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДІВ АВТОМАТИЧНОГО ОЦІНЮВАННЯ SQL-СКРИПТІВ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Науменко В. В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник доц. Яковлева О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри

_____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Науменку Вадиму Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методів автоматичного оцінювання SQL-скриптів з використанням великих мовних моделей для системи тестування знань студентів

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 26 листопада 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, платформа розробки .Net, мова програмування C#, фреймворк Angular, система керування базами даних PostgresQL, середовище розробки Microsoft Visual Studio Code, великі мовні моделі.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих сервісів для оцінювання.
2. Вивчення сучасних підходів та механізмів реалізації застосунку з методами автоматичного оцінювання.
3. Ознайомлення з великими мовними моделями.
4. Проєктування архітектури застосунку.
5. Розробка алгоритмів для автоматичного оцінювання.
6. Дослідження та розробка метода оцінювання на основі LLM.
7. Розробка інтерфейсу користувача.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми автоматичного оцінювання та проведення навчального процесу, постановка задачі, діаграма архітектури застосунку, діаграма бази даних застосунку, зображення результатів тестування програми.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Аналіз технічних засобів	15.10.25-20.10.25	
5	Розробка алгоритмів	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	25.11.25-27.11.25	
10	Перевірка на плагіат	27.11.25-29.11.25	
11	Рецензування	30.11.25	
12	Підготовка презентації та доповіді	01.12.25-02.12.25	
13	Занесення роботи в електронний архів	02.12.25	
14	Попередній захист кваліфікаційної роботи	03.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Яковлева О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 83 с., 12 табл., 23 рис., 50 джерел.

АВТОМАТИЧНЕ ОЦІНЮВАННЯ, ВЕЛИКІ МОВНІ МОДЕЛІ, НАВЧАЛЬНІ СИСТЕМИ, ШТУЧНИЙ ІНТЕЛЕКТ, GEMINI-2.0-FLASH, GPT-3.5-TURBO, SQL-SCRIPTS.

Об’єктом дослідження є процес автоматичного оцінювання SQL-скриптів студентів у системах електронного тестування знань.

Предметом дослідження є порівняння та перевірки правильності SQL-запитів із використанням великих мовних.

Метою дослідження є розробка та дослідження методів автоматичної оцінки SQL-скриптів на основі LLM, які забезпечать комплексну перевірку рішень студентів з урахуванням коректності, ефективності та якості написаного коду, а також формування адаптивного зворотного зв'язку.

Використано комбінацію евристичних та інтелектуальних підходів.

Наукова новизна роботи полягає у поєднанні методів евристичного аналізу SQL-структур із глибинними мовними моделями, що дозволяє не лише автоматично визначати правильність запиту, а й формувати інтерпретовані коментарі для студента природною мовою.

Взаємозв’язок з іншими роботами полягає у використанні сучасних принципів автоматизованого аналізу програмного коду та NLP-технологій, адаптованих до навчальних систем тестування знань.

Рекомендації щодо використання результатів роботи передбачають впровадження розробленого алгоритму у системи онлайн-тестування для вищих навчальних закладів.

У результаті дослідження створено прототип інтелектуального модуля автоматичного оцінювання SQL-скриптів.

ABSTRACT

Explanatory note to the qualification work: 83 pages, 12 table, 23 figures, 50 sources.

AUTOMATIC EVALUATION, LARGE LANGUAGE MODELS, TRAINING SYSTEMS, ARTIFICIAL INTELLIGENCE, GEMINI-2.0-FLASH, GPT-3.5-TURBO, SQL-SCRIPTS.

The object of the research is the process of automatic evaluation of students' SQL scripts in electronic knowledge testing systems.

The subject of the research is the comparison and verification of the correctness of SQL queries using large language models.

The aim of the research is to develop and experimentally investigate methods for the automatic evaluation of SQL scripts using LLM models, which allow determining the correctness of student queries and reducing the workload on the teacher.

A combination of heuristic and intelligent approaches is used.

The scientific novelty of the work lies in the combination of heuristic analysis methods of SQL structures with deep language models, which allows not only to automatically determine the correctness of the query, but also to form interpreted comments for the student in natural language.

The connection with other works lies in the use of modern principles of automated program code analysis and NLP technologies adapted to educational knowledge testing systems.

Recommendations for using the results of the work involve implementing the developed algorithm in online testing systems for higher education institutions.

As a result of the research, a prototype of an intelligent module for automatic evaluation of SQL scripts has been created.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8

Вступ 9

1 Аналіз ролі мови SQL, підходів та сервісів для її вивчення 11

1.1 Роль мови SQL та її затребуваність 11

1.2 Огляд програм для навчання програмуванню 15

1.3 Успіхи у вирішенні завдань комп'ютерного зору та обробки природної мови 18

1.4 Огляд програм для вивчення SQL 20

1.5 Постановка задачі дослідження 22

2 Розробка методів і алгоритмів оцінювання SQL-скриптів 25

2.1 Призначення методів оцінювання 25

2.2 Підготовка датасету 26

2.3 Метод грубий на основі евристичних правил і перевірки за допомогою LLMs 28

2.3.1 Евристичний метод 28

2.3.2 Алгоритм перевірки евристичної оцінки за допомогою LLMs 30

2.3.2.1 Вибір LLMs для подальшого тестування 31

2.3.2.2 Метрики для порівняння результатів моделей 34

2.3.2.3 Розробка та тестування промптів 34

2.3.2.4 Алгоритм на основі декількох LLMs 41

2.4 Метод гнучкої оцінки на основі AI 43

2.4.1 Гнучке оцінювання на основі готових LLMs 43

2.4.2 Аналіз результатів гнучкого оцінювання 45

2.5 Вибір остаточного алгоритму 48

3 Впровадження запропонованих алгоритмів у систему тестування знань студентів 50

3.1 Опис функціональності застосунка 50

3.2 Опис програмного забезпечення для застосунка 51

3.3 Архітектура застосунку 52

3.4 Ілюстрація роботи застосунка 58

3.4.1 Створення груп студентів 58

3.4.2 Створення тем, завдань та тестів 59

3.4.3 Проходження тестів та перегляд результатів 67

3.5 Оцінка роботи застоснку 69

Висновки 75

Перелік джерел посилання 78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс програмного застосунку)

Backend – серверна частина застосунку

Frontend – користувацький інтерфейс

GPT – Generative Pre-trained Transformer (Породжувальний попередньо натренований трансформер)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертекстових документів)

JSON – JavaScript Object Notation (запис об’єктів JavaScript)

LLM – велика мовна модель

REST – Representational State Transfer (передача репрезентативного стану)

SOAP – Simple Object Access Protocol (простий протокол доступу до об'єктів)

SQL – Structured Query Language (структурована мова запитів)

UI – User Interface (користувацький інтерфейс)

XML-RPC – Extensible Markup Language Remote Procedure Call (XML-виклик віддалених процедур)

ВСТУП

Сучасна освіта стрімко трансформується під впливом цифрових технологій та інтелектуальних систем підтримки навчання. З поширенням дистанційних курсів, онлайн-платформ та систем електронного тестування, таких як Moodle, Coursera, edX, виникла потреба в автоматизації процесів оцінювання студентських робіт. Особливо це стосується завдань із програмування та баз даних, де правильність рішення визначається не лише синтаксисом, а й логічною структурою запиту.

Предмет мови SQL є одним із ключових у підготовці спеціалістів з інформаційних технологій, аналітики даних та програмування. Проте перевірка SQL-запитів студентів – це трудомісткий процес, який потребує аналізу не лише результату виконання, а й самого коду. Викладачу необхідно враховувати можливі варіації правильних рішень, відмінності в синтаксисі, оптимізацію запитів і навіть логіку побудови умов. У масштабних онлайн-курсах або університетських потоках із сотнями студентів ручна перевірка таких завдань стає практично неможливою.

Актуальність теми полягає у зростаючій потребі автоматизованих систем оцінювання SQL-запитів, здатних аналізувати не лише правильність синтаксису, а й семантичну відповідність завданню. Традиційні методи, засновані на порівнянні з еталонними відповідями або тестових результатах виконання запиту, мають суттєві обмеження – вони не враховують альтернативні правильні рішення, не здатні формулювати коментарі до помилок і часто некоректно оцінюють часткову правильність.

Розвиток великих мовних моделей (Large Language Models, LLM), таких як GPT-3.5-Turbo, Gemini-2.0-Flash, Claude, Mistral тощо, відкрив нові можливості для автоматизації перевірки програмного коду. Такі моделі здатні інтерпретувати логіку SQL-запитів, аналізувати завдання, порівнювати рішення з еталоном і навіть формувати текстові пояснення до оцінки. Це робить їх

перспективним інструментом у сфері освіти, де важлива не лише оцінка, а й навчальний ефект.

У даній роботі запропоновано підхід до автоматичного оцінювання SQL-скриптів із використанням великих мовних моделей, який поєднує інтелектуальний аналіз коду з евристичними правилами перевірки. Система базується на строгому оцінюванні – визначення повної або часткової правильності (0 / 0,7 / 1) на основі збігу логіки запиту з еталонним рішенням.

У процесі дослідження проводиться тестування декількох промптів різних структур і мов (українською та англійською), а також експериментальна перевірка двох LLM-моделей – GPT-3.5-Turbo і Gemini-2.0-Flash. Для оцінювання ефективності застосовуються метрики MAE (Mean Absolute Error) і RMSE (Root Mean Square Error), що дозволяють визначити ступінь відхилення автоматичної оцінки від викладацької.

Отже, у роботі виконується комплексне дослідження можливостей великих мовних моделей у контексті автоматичного оцінювання SQL-запитів. Розроблений алгоритм дозволяє не лише виставляти оцінку, а й формувати інформативні коментарі до кожного рішення, що робить процес перевірки прозорим і навчально корисним для студентів. Результати дослідження можуть бути використані для впровадження інтелектуальних систем оцінювання у закладах вищої освіти, а також для підвищення ефективності масових онлайн-курсів з баз даних і програмування.

1 АНАЛІЗ РОЛІ МОВИ SQL, ПІДХОДІВ ТА СЕРВІСІВ ДЛЯ ЇЇ ВИВЧЕННЯ

1.1 Роль мови SQL та її затребуваність

Мова структурованих запитів (SQL) є стандартним засобом взаємодії з реляційними базами даних та займає ключове місце у сучасних інформаційних системах. Його унікальність полягає в декларативному характері, що дозволяє описувати, що саме потрібно отримати або змінити в даних, не вказуючи конкретних кроків виконання, що відрізняє його від імперативних мов програмування [1]. SQL заснований на строгих математичних принципах реляційної алгебри та теорії множин, що робить його не тільки інструментом для роботи з даними, але й засобом формування аналітичного та структурованого мислення у студентів. Знання SQL є необхідним для широкого спектру фахівців – розробників програмного забезпечення, аналітиків даних, інженерів за даними та дослідників, оскільки практично всі сучасні сховища даних, системи бізнес-аналітики (BI) та ETL-конвеєри спираються на SQL або діалекти.

Роль SQL в освітній та професійній сфері визначається його багатофункціональністю та універсальністю. По-перше, він є інструментом моделювання та забезпечення цілісності даних. Засоби опису структури баз даних (DDL) та механізми накладання обмежень, таких як первинні та зовнішні ключі, унікальні обмеження та перевірки даних, дозволяють будувати коректні та надійні моделі даних. По-друге, SQL забезпечує широкий набір інструментів для запитів та перетворення даних (DML), включаючи фільтрацію, об'єднання таблиць, агрегації та вкладені підзапити. Ці можливості формують у тих, хто навчається так зване «множинне мислення», що вимагає оперування цілими наборами даних, а не окремими елементами. По-третє, мова SQL є основою транзакційної обробки та управління конкурентним доступом, що забезпечує надійність і передбачуваність роботи розрахованих на багато користувачів

систем. Крім того, сучасні діалекти SQL включають розвинені аналітичні можливості, такі як віконні функції та розширені механізми угруповання даних, що робить його застосовним у завданнях бізнес-аналітики, когортного аналізу та обробки часових рядів. Важливим напрямком розвитку мови є підтримка напівструктурованих даних (JSON, XML), геопросторових типів та засобів повнотекстового пошуку, що розширює сферу його застосування. Нарешті, SQL відіграє ключову роль у забезпеченні продуктивності та масштабованості систем: механізми індексації, партиціонування та аналізу планів виконання запитів дозволяють студентам та спеціалістам усвідомлено проектувати та оптимізувати системи зберігання та обробки даних.

В освітньому процесі навчання SQL сприяє формуванню точності формулювань, навичок декомпозиції завдань та суворої перевірки гіпотез. SQL виступає у ролі універсальної мови взаємодії в мультиплатформенному середовищі, оскільки діалекти, які у таких системах як Spark, Trino, BigQuery чи Snowflake, зберігають основну структуру мови. Це робить знання SQL фундаментальними та довготривалими.

Підходи до вивчення SQL можуть відрізнятися залежно від цілей та рівня підготовки студентів. Найбільш поширеними є такі підходи: концептуально-теоретичний, практико-орієнтований, проєктний, дослідницький, тест-орієнтований, активного навчання, спірального навчання.

Концептуально-теоретичний підхід передбачає вивчення основ проектування баз даних, включаючи моделювання сутностей та зв'язків (ER-моделювання), нормалізацію даних та проектування фізичних схем. Цей підхід розвиває розуміння того, як вибір структури даних та обмежень впливає на коректність та ефективність роботи системи.

Практико-орієнтований підхід ґрунтується на послідовних вправах. В рамках нього учні починають із простих операцій фільтрації, потім переходять до з'єднань, агрегацій, підзапитів та віконних функцій. Такий метод сприяє формуванню «практичної пам'яті» на синтаксис та поширені патерни запитів.

Проєктний підхід це підхід, у якому студенти пропонують вирішувати комплексні завдання – від проектування схеми даних до написання звітів та аналізу результатів. Він забезпечує зв'язок теоретичних знань із реальними бізнес-завданнями.

Дослідницький підхід передбачає аналіз та оптимізацію запитів, порівняння стратегій індексації, вивчення впливу різних конструкцій SQL на продуктивність.

Тестувально-орієнтований передбачає навчання, у якому кожне завдання супроводжується формалізованими критеріями правильності результату. Для перевірки можуть використовуватись автоматичні тести, що оцінюють коректність вибірки, відсутність дублікатів та стійкість до відсутніх значень.

Активного навчання передбачає навчання з елементами peer-review, коли студенти аналізують рішення один одного, проводять спільний розбір помилок та пропонують альтернативні способи реалізації запитів, що сприяє розвитку критичного мислення та здатності до колективної роботи.

Спірального навчання передбачає багаторазове повернення до вже вивчених тем з поступовим збільшенням складності завдань та порівнянням особливостей різних діалектів SQL, що важливо у сучасних умовах різноманітності СУБД.

Серед типових труднощів щодо SQL можна виділити проблеми, пов'язані з розумінням роботи з NULL і тризначною логікою, специфіку зовнішніх з'єднань і фільтрації даних, різницю між агрегацією і віконними функціями, і навіть питання детермінізму виконання запитів без явного вказівки порядку (ORDER BY). Окремою темою є індексація, яка потребує глибокого розуміння впливу структури індексу на продуктивність.

Сучасне цифрове середовище надає широкий спектр інструментів та сервісів для вивчення SQL. Найбільш популярні такі:

– інтерактивні пісочниці (DB Fiddle, SQL Fiddle, SQLite Online) [2–4], які дозволяють виконувати запити без необхідності встановлення СУБД, що є особливо корисним на початкових етапах навчання;

– хмарні середовища, такі як BigQuery Sandbox або Snowflake Trial, що надають доступ до сучасних діалектів SQL та реальних обсягів даних;

– інтерактивні тренажери та курси (SQLZoo, SQLBolt, DataCamp, Codecademy, LeetCode Database) [5-9], які забезпечують автоматичну перевірку рішень та містять широкий набір вправ різного рівня складності;

– задачники та довідники, наприклад, PostgreSQL Exercises або w3resource SQL, які дозволяють закріпити знання з конкретних тем та структур даних;

– набір навчальних даних, таких як Northwind, Sakila або IMDB, які використовуються для моделювання реальних завдань аналізу даних;

– локальні середовища розробки, що включають DBeaver, pgAdmin, Docker-контейнери та розширення для VS Code [10-13], що забезпечують повний контроль над процесом розробки та відтворюваність експериментів;

– інструменти якості та моделювання, такі як dbt (data build tool), що дозволяють впроваджувати практики промислової розробки, включаючи автоматичне тестування та CI/CD для SQL-запитів;

– BI-інструменти (Power BI, Looker Studio, Metabase), які навчають студентів пов'язувати SQL-запити з візуалізацією даних та аналітичними метриками.

Вибір конкретного інструменту залежить від завдань та рівня підготовки. Важливими критеріями є підтримка потрібного діалекту, наявність засобів аналізу продуктивності (EXPLAIN/ANALYZE), можливість завантажувати власні дані, прозору структуру даних та механізми автоперевірки рішень.

Сучасні досягнення в галузі штучного інтелекту, особливо поява великих мовних моделей (LLM), відкривають нові можливості для навчання SQL. LLM можуть пояснювати помилки зрозумілою мовою, генерувати приклади даних, пропонувати альтернативні підходи до вирішення завдань, проводити автоматичне рецензування коду і навіть імітувати реальних викладачів у діалогових системах. Однак використання LLM пов'язане з певними ризиками, такими як змішування синтаксису різних діалектів, генерація некоректних або

неефективних рішень та відсутність обліку конкретної структури бази даних. Для мінімізації ризиків необхідно забезпечувати явну специфікацію використовуваної СУБД, передавати моделі точний опис схеми даних та використовувати автоматизовані тести для перевірки коректності результатів.

Таким чином, SQL є не тільки ключовим інструментом роботи з даними, але й потужним засобом формування професійних компетенцій студентів у галузі аналізу та управління інформацією. Ефективне навчання SQL можливе лише при поєднанні теоретичної бази, практичного відпрацювання навичок, системного зворотного зв'язку та використання сучасних цифрових інструментів. Автоматизація оцінки SQL-скриптів із застосуванням великих мовних моделей стає важливим напрямом розвитку освітніх технологій, оскільки дозволяє масштабувати процес перевірки знань, забезпечувати об'єктивність оцінювання та індивідуалізувати навчання за рахунок гнучкого зворотного зв'язку та аналізу не тільки результату виконання запиту, а й його структури, якості та продуктивності.

1.2 Огляд програм для навчання програмуванню

У сучасному освітньому процесі широкого поширення набули програми та платформи, призначені для навчання програмуванню. Їхня основна мета полягає у спрощенні процесу освоєння мов програмування за рахунок надання інтерактивних інструментів, що не потребують складного налаштування середовища розробки. Дані програми застосовуються у вузівських курсах, онлайн-освіті та при самостійному вивченні, забезпечуючи доступність матеріалів та автоматизацію процесу перевірки знань. Їх розвиток пов'язаний з активним впровадженням технологій гейміфікації, мікронавчання та штучного інтелекту, що дозволяє реалізувати персоналізований підхід до навчання та підвищити його ефективність.

Програмні рішення для навчання програмування виконують кілька ключових функцій.

По-перше, вони знижують поріг входу рахунок спрощених інтерфейсів і покрокового освоєння матеріалу.

По-друге, такі програми забезпечують оперативний зворотний зв'язок: автоматична перевірка завдань дозволяє користувачеві швидко виявляти помилки та коригувати рішення.

По-третє, вони сприяють регулярній практиці рахунок коротких вправ, що відповідає принципам безперервного навчання.

У ряді додатків використовуються адаптивні механізми, що аналізують прогрес користувача та формують завдання залежно від рівня підготовки.

Найбільш поширені програми та платформи: SoloLearn, Mimo, Grasshopper, Enki, Programming Hub, Dodona, візуальні навчальні програми.

SoloLearn підтримує широкий набір мов програмування, включаючи Python, JavaScript, C++ та SQL. Містить теоретичні матеріали, практичні завдання та систему автоматичної перевірки. Застосовується на початкових етапах навчання, але обмежена у можливостях поглибленого аналізу та складних проектів [14].

Mimo це застосунок з акцентом на мікронавчання. Структура курсу побудована на коротких інтерактивних уроках, що сприяє регулярному освоєнню матеріалу. Основний набір мов, що підтримуються, включає Python, JavaScript, HTML, CSS і SQL [15].

Grasshopper це проект Google, орієнтований навчання основам програмування на JavaScript. Використовує візуальні завдання та поетапне введення синтаксичних конструкцій мови [16].

Enki це програма, що надає завдання та короткі матеріали з мов програмування та суміжних технологічних областей. Призначено для закріплення знань та повторення матеріалу [17].

Programming Hub це система, що підтримує до 18 мов програмування, що включає вбудований інтерпретатор та каталог навчальних матеріалів [18].

Dodona це академічна платформа, орієнтована на автоматизоване навчання з наданням зворотного зв'язку за виконаними завданнями. Використовується в університетських курсах та відрізняється можливістю аналізу успішності студентів [19].

Візуальні навчальні програми, такі як Lightbot і ScratchJr [20, 21], призначені для формування базових уявлень про логіку алгоритмів у користувачів-початківців і дітей молодшого шкільного віку. Вони застосовують візуальні елементи та ігрові завдання для пояснення базових понять – циклів, умов, процедур.

Переваги даних рішень:

– доступність, яка передбачає можливість навчання без встановлення локальних СУБД чи IDE;

– автоматизація перевірки, тобто коли миттєва оцінка результатів дозволяє виключити суб'єктивність та прискорити процес навчання;

– гейміфікація, або ж використання системи балів, досягнень та рейтингів для підвищення залученості;

– різноманітність мов та широкий вибір інструментів для різних напрямів підготовки;

– підтримка індивідуальної траєкторії, це коли деякі системи надають персоналізовані рекомендації.

До основних обмежень можна віднести:

– недостатню глибину опрацювання матеріалу, особливо в галузях, пов'язаних з оптимізацією та промисловою розробкою;

– спрощеність завдань, що не відображає реалій роботи з реальними даними та системами;

– обмежені засоби аналізу, такі як профільування продуктивності та візуалізація планів виконання;

– залежність низки функцій від платної передплати;

– складнощі при переході від візуального формату до професійних інструментів розробки.

Ефективність застосування даних додатків найбільша на початкових етапах навчання, коли потрібно сформувати базові навички роботи з синтаксисом та алгоритмічним мисленням. Для комплексного формування компетенцій необхідно їх використання у поєднанні з проектною діяльністю та традиційними освітніми методами. Інтеграція додатків із системами автоматичної перевірки рішень дозволяє створювати гібридні моделі навчання, за яких студенти отримують автоматичну оцінку та зворотний зв'язок за якістю рішень, а викладачі – інструменти моніторингу та аналітики.

1.3 Успіхи у вирішенні завдань комп'ютерного зору та обробки природної мови

Досягнення в галузі комп'ютерного зору та обробки природної мови обумовлені сукупністю факторів: масштабуванням даних та обчислень, переходом до уніфікованих архітектур на основі трансформерів, розвитком навчання самонагляду та подальшою адаптацією під інструкції користувача. На рівні прикладних завдань це виявляється у стійкому підвищенні якості за широким спектром сценаріїв: розпізнавання та розуміння зображень та відео (класифікація, детекція, сегментація, візуальне питання-відповідь, OCR та аналіз документів/діаграм), генерація та редагування зображень та відео за текстовим описом, машинний переклад, вилучення фактів рефакторинг коду, а також аудіо-завдання «аудіо→текст» (ASR), «текст→аудіо» (TTS), «аудіо→аудіо» (переклад мови) та мультимодальні взаємодії «текст↔зображення/відео/аудіо».

У комп'ютерному зорі ключовим зрушенням стало заміщення виключно згорткових архітектур трансформерами (Vision Transformer та його модифікації) та поширення самосупервізії (масковане моделювання, контрастивні цілі), що усунуло залежність від масштабної ручної розмітки. Паралельно дифузійні моделі стали стандартом для високоякісної генерації та редагування

зображень/відео за текстом. Вони доповнюються керуючими контурами (контури пози/глибини, «instruct»-адаптації) та зв'язками з мовними моделями, які виступають «планувальниками» та формують докладні промпти та обмеження для генератора. У завданнях документного зору та наукової візуалізації (сторінки з версткою, таблиці, графіки) зростання якості забезпечується поєднанням OCR, сегментації макета та візуально-мовних моделей, що дозволяє виконувати структуроване вилучення та семантичне анотування. Також існує багато задач, де відбувається поєднання нейромережевого підходу з класичними алгоритмами комп’ютерного зору [22–29].

У сфері обробки природної мови найбільш значний прогрес пов'язані з великими мовними моделями (LLM) з урахуванням декодерних трансформерів [30–36]. Передбачення на великих неконтрольованих корпусах з цілями автодоповнення формує універсальні розподілені уявлення, здатні до навчання за підказкою. Наступне доналаштування за інструкціями та уподобаннями підвищує відповідність виведення вимогам користувача та керованість моделі. Розширення контекстного вікна, поява механізмів планування та інструментного режиму дозволяє переносити модель з режиму «тексто-генератора» в режим «агента» [37], який:

– звертається до зовнішніх інструментів (наприклад, виконавців коду, баз знань, компіляторів, СУБД),

– використовує RAG для підвищення фактичної точності;

– контролює багатокрокові рішення.

Для інженерних завдань істотна спеціалізація LLM на коді: навчання на корпусах вихідників, парах «завдання-тести», патчах та обговореннях рев'ю дає моделі здатність до синтезу програм, пояснення помилок та автогенерації тестів.

З точки зору освітніх додатків, успіхи в CV, NLP, LLM та мультимодальності забезпечують нові формати взаємодії: голосові та візуальні інтерфейси до завдань програмування, автоматичне розпізнавання та розбір

зображень з умовою (наприклад, схем БД, діаграм), генерацію допоміжних артефактів (псевдокод) персоналізований зворотний зв'язок, узгоджений з цілями курсу. Для дисциплін, пов'язаних із базами даних, стає практично значущим зв'язати LLM-агента з інструментами БД: модель формує або виправляє SQL-запит, запускає його на ізольованих наборах даних, аналізує план, перевіряє інваріанти та пояснює знайдені невідповідності; за наявності візуального контексту (знімки схем, ER-діаграми) мультимодальна частина забезпечує «розуміння» структури. Перспективи розвитку пов'язані з інтеграцією формально-верифікованих кроків (typed-AST, перевірка еквівалентності запитів, синтез контрприкладів), а також з багатоагентними композиціями, де спеціалізовані агенти за кодом, даними та візуальними структурами координуються над єдиним навчальним сценарієм. Такий підхід підвищує як якість автоматичної оцінки, і педагогічну цінність зворотний зв'язок, зберігаючи перевіряемость результату.

1.4 Огляд програм для вивчення SQL

В даний час існує широкий спектр додатків та онлайн-платформ, призначених для вивчення мови SQL. Їх можна умовно розділити на дві групи: додатки, які використовують штучний інтелект (ІІ) у навчальному процесі, та рішення, які не використовують ШІ та засновані на традиційних методах навчання. Кожна група має свої особливості, підходи до організації навчання та цільові сценарії застосування.

Програми без використання ШІ орієнтовані класичне покрокове освоєння SQL. Вони надають учню теоретичний матеріал, статичний набір вправ і автоматичну перевірку рішень за заданими шаблонами. Основною метою таких платформ є формування базових навичок роботи з реляційними базами даних та закріплення синтаксису SQL.

До найпоширеніших рішень належать: SQLZoo, Mode SQL Tutorial,

HackerRank SQL, LeetCode SQL, W3Schools SQL.

SQLZoo це інтерактивний тренажер із завданнями щодо фільтрації даних, об'єднання таблиць та агрегацій. Перевірка здійснюється за фіксованими еталонними відповідями.

Mode SQL Tutorial це онлайн-сервіс з уроками та вбудованою пісочницею для написання та перевірки запитів.

HackerRank SQL і LeetCode SQL це платформи з великою кількістю практичних завдань, які використовуються як для навчання, так і для підготовки до співбесід.

W3Schools SQL це ресурс з базовими прикладами та вбудованим середовищем для виконання простих запитів.

Переваги цих систем: низький поріг входу, стабільність роботи та передбачуваність результатів. Однак у них є обмеження: завдання статичні, не адаптуються до рівня учня, а система не надає пояснень помилок і рекомендацій щодо поліпшення рішень.

Додатки, що використовують ШІ, є сучаснішим класом систем, які застосовують алгоритми машинного навчання та великі мовні моделі для персоналізації навчального процесу та інтелектуальної підтримки учня. У таких системах ІІ виконує кілька ключових функцій:

– динамічна генерація завдань, ШІ формує унікальні бази даних та запити, що запобігає заученню готових відповідей;

– інтелектуальна перевірка рішень, коли аналізується не лише результат запиту, а й його структура, можливі помилки та неефективність. Система може запропонувати рекомендації щодо оптимізації;

– пояснення помилок природною мовою це процес, коли той, хто навчається, отримує докладний розбір своїх рішень;

– адаптивне навчання, коли завдання підлаштовуються під рівень знань та прогрес користувача;

– мультимодальні сценарії це можливість роботи з візуальними матеріалами, наприклад, схемами баз даних або ER-діаграмами.

Приклади таких рішень:

– ChatGPT та Copilot Chat допомагають формулювати SQL-запити, пояснюють їхню логіку та пропонують виправлення;

– DataCamp AI Assistant це вбудований у платформу DataCamp інтелектуальний помічник, який спрямовує користувача під час вирішення завдань;

– Text-to-SQL системи (наприклад, AI SQL Tutor) перетворюють текстові інструкції на SQL-код і аналізують його;

– Kaggle AI SQL Playground це середовище, де ІІ генерує завдання та аналізує якість рішень.

Основна перевага ШІ-застосунків – можливість формування персоналізованої траєкторії навчання та глибокого аналізу рішень. На відміну від традиційних систем, вони здатні не тільки фіксувати правильність відповіді, але й пояснювати причини помилок, що особливо важливо щодо складних конструкцій SQL. Однак такі програми вимагають великих обчислювальних ресурсів, а також включають ризик створення некоректних або неефективних запитів. Тому вони потребують додаткової верифікації та інтеграції механізмів контролю.

1.5 Постановка задачі дослідження

Сучасні освітні процеси у галузі інформаційних технологій потребують ефективних інструментів для перевірки знань студентів. Мова SQL займає ключове місце у підготовці фахівців з баз даних, аналізу даних та розробки програмного забезпечення. Традиційні системи тестування знань, що ґрунтуються на статичних завданнях та шаблонній перевірці рішень, мають низку обмежень: вони не здатні адаптуватися до рівня студента, не надають розгорнутих пояснень помилок та не оцінюють якість SQL-скриптів за критеріями оптимальності та читаності. У той же час розвиток великих мовних

моделей (LLM) та мультимодальних моделей відкрив нові можливості для автоматичної генерації завдань, аналізу та інтелектуальної оцінки рішень.

Застосування LLM дозволяє не тільки перевірити коректність результату виконання SQL-запиту, але й аналізувати його структуру, виявляти потенційні помилки, формувати рекомендації щодо оптимізації та надавати учню пояснення природною мовою. Це суттєво підвищує якість зворотного зв'язку та сприяє глибшому розумінню матеріалу студентами. Таким чином, розробка методів автоматичної оцінки SQL-скриптів з використанням LLM є актуальним завданням, спрямованим на підвищення ефективності систем тестування знань та наближення їх до сучасних освітніх стандартів.

Об’єктом дослідження є процес автоматичного оцінювання SQL-скриптів студентів у системах електронного тестування знань.

Метою дослідження є розробка та дослідження методів автоматичної оцінки SQL-скриптів на основі LLM, які забезпечать комплексну перевірку рішень студентів з урахуванням коректності, ефективності та якості написаного коду, а також формування адаптивного зворотного зв'язку.

Для досягнення цієї мети було сформульовано наступні задачі:

– провести аналіз існуючих підходів та додатків для вивчення SQL, виділивши їх сильні та слабкі сторони, включаючи системи, які використовують ШІ, та традиційні платформи без ШІ;

– дослідити можливості сучасних великих мовних моделей та мультимодальних систем для завдань аналізу коду та автоматичного тестування, приділяючи увагу їх застосовності до SQL-запитів;

– розробити методологію представлення SQL-скриптів та еталонних даних для автоматизованої перевірки, що включає тестові набори та критерії оцінки;

– розробити метод автоматичної інтерпретації та оцінки SQL-запитів з використанням LLM, що включає:

– перевірку коректності результату виконання запиту;

– аналіз структури та синтаксису скрипту;

– виявлення неефективних конструкцій та потенційних помилок;

– формування пояснень та рекомендацій для студента;

– реалізувати прототип системи автоматичної оцінки SQL-скриптів, інтегрувавши їх у навчальну платформу чи систему тестування знань;

– провести експериментальне дослідження розроблених методів, порівнявши їх із традиційними підходами за критеріями точності оцінки, часу обробки та якості зворотного зв'язку;

– сформулювати рекомендації щодо застосування LLM в освітніх системах для автоматичної перевірки знань, включаючи аспекти надійності, точності та інтерпретації рішень.

Реалізація даної роботи дозволить підвищити об'єктивність та якість оцінки знань студентів, забезпечити індивідуалізований підхід до навчання та знизити навантаження на викладачів за рахунок автоматизації процесу перевірки та формування зворотного зв'язку.

2 РОЗРОБКА МЕТОДІВ І АЛГОРИТМІВ ОЦІНЮВАННЯ SQL-СКРИПТІВ

2.1 Призначення методів оцінювання

Методи автоматичного оцінювання призначені для аналізу та перевірки правильності SQL-скриптів, що виконуються студентами в системі тестування знань. На вхід алгоритмів подаються:

- структура бази даних;
- текст завдання та його вимоги;
- еталонний SQL-запит;
- студентський SQL-запит.

На виході формується оцінка коректності рішення та коментар, що пояснює, наскільки студент виконав поставлене завдання.

Система оцінювання реалізована у двох форматах: формат грубого оцінювання та гнучкого оцінювання.

Грубе оцінювання буде використовується для швидкої загальної перевірки коректності рішення. Алгоритм видає підсумкову оцінку в діапазоні 0, 0,7 або 1, де: «0» – запит невірний або звертається до неправильних даних; «0,7» – рішення частково правильне, відповідає завданню, але не дотримується всіх вимог; «1» – запит повністю коректний, відповідає завданню та містить усі необхідні елементи.

Гнучке оцінювання – застосовується для детального аналізу за кількома критеріями (адекватність, синтаксис, вибір таблиць, атрибутів, відповідність результату та вимог). Для кожного критерію виставляється оцінка в діапазоні від 0 до 1, що дозволяє більш точно визначити ступінь виконання завдання і формувати розгорнутий зворотний зв'язок.

2.2 Підготовка датасету

Для проведення експериментів і навчання методів автоматичного оцінювання SQL-скриптів було підготовлено спеціалізований набір даних [38, 39], що містить структуровані дані про завдання, еталонні рішення та відповіді студентів. Основна мета формування такого набору даних – забезпечити можливість систематичного тестування та аналізу роботи алгоритмів як евристичних, так і заснованих на великих мовних моделях (LLM).

Набір даних являє собою таблицю, де кожен рядок відповідає одному навчальному завданню та його виконанню студентом. Набір складається з 170 рядків, де кожен рядок – це пройдене студентом та оцінене вчителем питання.

Він включає шість ключових стовпців: «question», «etalonAnswer», «databaseScript», «dataScript», «studentAnswer», «teacherComment».

Стовбець «question» містить текст формулювання завдання, яке повинен вирішити студент. Це може бути опис завдання на вибірку, агрегацію або з'єднання таблиць в SQL.

Стовбець «etalonAnswer» містить еталонний SQL-запит, який вважається правильним рішенням завдання. Він служить базою для порівняння з відповіддю студента.

Стовбець «databaseScript» містить скрипт, що створює структуру бази даних: таблиці, атрибути, зв'язки між ними. Цей елемент необхідний для коректного аналізу запитів, оскільки дозволяє алгоритму розуміти, до яких таблиць і полів звертається студент.

Стовбець «dataScript» містить скрипт з тестовими даними, які заповнюють створені таблиці. Ці дані використовуються для реального виконання SQL-запитів і порівняння результатів виконання еталонного і студентського рішень.

Стовбець «studentAnswer» містить SQL-запит, написаний студентом. Саме цей елемент підлягає аналізу з боку розроблених алгоритмів оцінювання.

Стовбець «teacherComment» містить коментар викладача, що містить експертну оцінку коректності рішення. Це поле використовується як «істинна мітка» (ground truth) для перевірки якості роботи автоматичних методів оцінювання.

Вигляд набору зображено на рисунку 2.1.

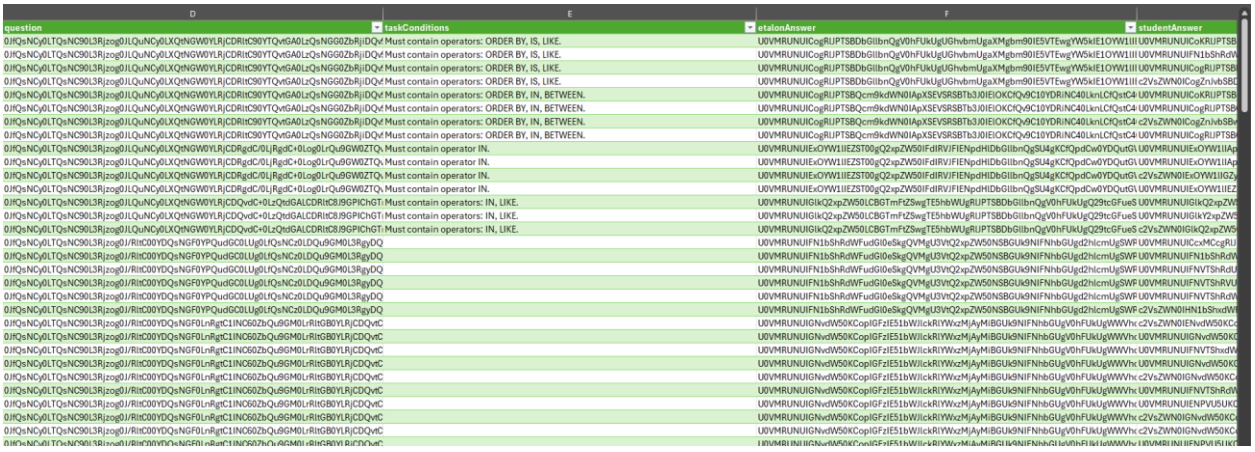


Рисунок 2.1 – вигляд набору даних

Для запобігання втрати інформації, пов'язаної з кодуванням символів та особливостями SQL-синтаксису, всі текстові дані в колонках question, etalonAnswer, studentAnswer, databaseScript та dataScript були закодовані у формат base64. Такий захід забезпечує:

– збереження спеціальних символів (наприклад, лапок, переносів рядків, відступів, знаків пунктуації);

– коректну передачу та зберігання даних у форматі CSV, де деякі символи можуть інтерпретуватися як роздільники;

– запобігання пошкодженню SQL-запитів під час експорту, імпорту та обробки в різних середовищах;

– однозначне відновлення вихідного тексту під час декодування.

Процес підготовки набору даних включав такі етапи:

Етап 1. Збір даних – формулювання SQL-завдань і підготовка еталонних рішень викладачем.

Етап 2. Генерація структур баз даних і тестових даних – створення SQL-скриптів, що описують схему таблиць і наповнення їх демонстраційними значеннями.

Етап 3. Збір студентських рішень – отримання SQL-запитів, написаних студентами в рамках лабораторних або тестових завдань.

Етап 4. Аналіз і розмітка – додавання експертних коментарів викладача, що відображають правильність і якість рішення.

Етап 5. Кодування в base64 – перетворення вмісту всіх текстових полів, крім «teacherComment», у формат base64 для забезпечення сумісності та цілісності даних.

Етап 6. Формування підсумкового CSV-файлу – об'єднання всіх записів в єдину таблицю, придатну для подальшого завантаження та обробки.

2.3 Метод грубий на основі евристичних правил і перевірки за допомогою LLMs

2.3.1 Евристичний метод

Евристичний метод призначений для автоматичної перевірки коректності SQL-запитів студентів на основі порівняння результатів їх виконання з еталонним рішенням. Він використовує формальні правила і послідовність обчислювальних кроків, що дозволяють визначити ступінь збігу результатів і на цій основі виставити оцінку. Метод не аналізує змістовну сторону запиту, а оцінює тільки його вихідні дані та структуру, що робить його швидким, відтворюваним і простим у реалізації.

Робота евристичного методу включає наступні основні кроки:

Крок 1. Отримання результатів виконання двох сценаріїв. Під час цього кроку виконуються два SQL-запити: еталонний і студентський. Результати їх виконання зберігаються у вигляді таблиць даних (наборів рядків і стовпців), які далі порівнюються між собою.

Крок 2. Перетворення результатів в уніфікований формат, тобто обидві вибірки конвертуються в масив словників, де кожен словник представляє окремий рядок, а ключами виступають імена атрибутів (стовпців). Це дозволяє проводити порівняння поелементно, незалежно від порядку даних.

Крок 3. Порівняння кількості рядків, тобто якщо кількість рядків у результатах відрізняється, це вказує на неповне або надмірне вилучення даних студентом. У такому випадку базова оцінка знижується наполовину від максимально можливої.

Крок 4. Порівняння кількості атрибутів (стовпців), тобто якщо структура результатів відрізняється за кількістю повернутих стовпців, оцінка додатково зменшується на 10 % від поточного значення.

Крок 5. Порівняння назв атрибутів. Під час цього кроку проводиться зіставлення імен стовпців у результатах еталонного та студентського запитів. Якщо частина атрибутів названа неправильно або відсутня, оцінка знижується ще на 10 %.

Крок 6. Порівняння даних у таблицях. Під час цього кроку обчислюється кількість збіжних значень між результатами обох запитів:

Крок 7. Визначається максимальна кількість можливих збігів (добуток кількості атрибутів на кількість рядків у максимальному з двох результатів).

Крок 8. Обчислюється коефіцієнт збігу як відношення знайдених збігів до максимальної кількості;

Крок 9. Підсумкова оцінка множиться на цей коефіцієнт і округлюється до двох знаків після коми.

Результатом роботи евристичного методу є числова оцінка в діапазоні від 0 до 1, що відображає ступінь схожості між результатами еталонного і студентського SQL-запитів. Чим ближче значення до 1, тим вищий ступінь відповідності.

Таким чином, евристичний метод виконує роль формального рівня оцінки, який швидко визначає, наскільки висновок студентського скрипта збігається з еталонним.

2.3.2 Алгоритм перевірки евристичної оцінки за допомогою LLMs

Евристичний метод, незважаючи на свою простоту і швидкість, не може вважатися повністю надійним інструментом для оцінки SQL-запитів студентів. Його основне обмеження полягає в тому, що він аналізує тільки зовнішню збіжність результатів виконання запитів, не враховуючи логічну коректність і смислову складову коду. У деяких випадках студентський запит може випадково повернути ті ж дані, що й еталонний, але при цьому використовувати невірні таблиці, некоректні фільтри або пропустити важливі умови завдання. Така ситуація призводить до хибно позитивних результатів, коли алгоритм виставляє високу оцінку некоректному рішенню.

Щоб підвищити точність і достовірність системи оцінювання, було прийнято рішення ввести етап перевірки евристичної оцінки за допомогою великих мовних моделей (LLM). Цей етап виконує роль вторинного контролера, який інтерпретує результати евристичного аналізу і уточнює коректність виставленої оцінки.

Основне призначення методу перевірки за допомогою LLM полягає в наступних етапах:

Етап 1. Підтвердження коректності евристичної оцінки. Модель аналізує не тільки результати виконання запитів, але і структуру бази даних, формулювання завдання, вимоги і зміст SQL-скрипта. Це дозволяє встановити, чи дійсно студент реалізував потрібну логіку.

Етап 2. Виявлення помилкових збігів. Якщо евристичний метод присвоїв високу оцінку через випадковий збіг даних, LLM здатний визначити логічну помилку в запиті і знизити оцінку.

Етап 3. Формування пояснень. Модель дає текстовий коментар, що пояснює, чому оцінка є коректною або вимагає перегляду. Це підвищує прозорість процесу оцінювання.

Процес перевірки здійснюється наступним чином [40, 41]:

– паралельна робота двох алгоритмів, тобто евристичний метод і LLM одночасно отримують одні й ті ж вхідні дані: структуру бази даних, текст завдання, вимоги, еталонний SQL-запит і запит студента. Кожен з них незалежно обчислює оцінку;

– зіставлення результатів, а саме після завершення роботи обох методів система порівнює їх оцінки;

– перевірка на викид (розбіжність), якщо різниця між оцінками евристичного алгоритму і LLM перевищує 20 %, це вважається індикатором невизначеності;

– фіксація можливої помилки, якщо у разі виявлення викиду система позначає дану роботу як таку, що потребує додаткової перевірки, і передає інформацію викладачеві;

– інтерпретація результату, тобто якщо розбіжність між двома оцінками не перевищує поріг, підсумкова оцінка приймається як підтверджена, а коментар LLM використовується для пояснення студенту.

2.3.2.1 Вибір LLMs для подальшого тестування

Для автоматичного оцінювання SQL-скриптів було розглянуто кілька сімейств великих мовних моделей (LLM): OpenAI (GPT), Google (Gemini), Anthropic (Claude) [42-44], а також альтернативи в схожому ціновому та якісному сегменті. Порівняння проводилося за трьома практичними критеріями:

– вартість обробки токенів;

– швидкість/затримка відповіді;

– якість і стабільність виведення (в тому числі дотримання формату JSON, робота з українською та англійською мовами, стійкість до варіативності промптів).

Порівняння моделей за цінами зображено у таблиці 2.1.

Таблиця 2.1 – Порівняння моделей за цінами

Сімейство	**Модель**	**Ціна input/output (за 1M токенів)**
Google	Gemini 2.0 Flash	$0.10 / $0.40
OpenAI	GPT-3.5-turbo	~$1.5 / ~$2.0
Anthropic	Claude 3.5 Sonnet	$3 / $15
Anthropic	Claude Opus 4.1	$15 / $75

Якщо порівнювати за швидкості/затримці, то за відкритими вимірами та оглядами, Gemini 2.0 Flash позиціонується як швидка «прикладная» модель: середня затримка до першого токена близько ~0,5 с, швидкість генерації близько ~170 ток/с (за незалежним зведеним оглядом Artificial Analysis, процитованим в The Batch / deeplearning.ai). Це робить її зручною для інтерактивних сценаріїв і багаторазових прогонів оцінювання. Даних з таким же ступенем деталізації для GPT-3.5-turbo в офіційній документації менше; практичні вимірювання в індустрії показували, що GPT-3.5-turbo зазвичай швидший за «великі» моделі, але поступається спеціалізованим «flash/mini»-класам. У підсумку, за «швидкістю/ціною» Gemini 2.0 Flash виглядає кращим варіантом для масового прогону студентських рішень.

При порівнянні якості та стабільності єдиних «абсолютних» метрик для нашого вузькоспеціалізованого завдання (асистент, що оцінює SQL-скрипти) немає; тому орієнтувалися на:

– здатність суворо дотримуватися формату JSON;

– стійкість до промптів, мова яких не збігалася з необхідною мовою виведення інформації;

– стабільність при повторних прогонах;

– усереднені результати незалежних зведених таблиць за сучасними бенчмарками (reasoning/код/універсальні), де моделі класу Claude 3.5 Sonnet і Gemini 2.0 демонструють високий рівень, а GPT-3.5 – «базовий/надійний» рівень при низькій ціні [44].

Хоча Claude 3.5 Sonnet і лінійка Opus показують сильні результати на reasoning-бенчмарках і діалоговій дисципліні, але вартість виведення (особливо $15/M токенів у Sonnet і $75/M у Opus) значно вища, ніж у Gemini Flash і GPT-3.5-turbo. Для нашого завдання оцінювання безлічі SQL-відповідей (170+ кейсів і потенційно тисячі прогонів валідації) це помітно збільшує бюджет. Тому остаточний вибір пав на моделі GPT-3.5-turbo (OpenAI) і Gemini 2.0 Flash (Google) через перелічені нижче аргументи.

Модель Gemini 2.0 Flash дає один з найкращих показників «ціна/швидкість» в індустрії: $0,10/$0,40 за 1 млн токенів (in/out), низька латентність і висока швидкість виведення. Для нашого пайплайну автоматичної перевірки SQL це критично: можна швидко проганяти великі партії відповідей і оперативно підбирати оптимальні промпти [43].

Модель GPT-3.5-turbo історично є дуже дешевою і масовою моделлю OpenAI, що корисно для базової валідації і як «друга думка» при вкрай низькій вартості токена [42].

Модель Gemini 2.0 Flash – низька затримка і висока пропускна здатність (зручно для інтерактивного контуру і швидкої перевірки промптів).

У наших тестах Gemini 2.0 Flash краще утримував формат JSON і дисципліну інструкцій, що зменшувало пост-обробку. Це узгоджується з позиціонуванням моделі як «швидкої та практичної».

Модель GPT-3.5-turbo іноді чутливіший до формулювання промпта, але саме цей факт корисний в нашому підході так як, розбіжності між моделями допомагають виловлювати сумнівні рішення і помічати їх для ручної перевірки викладачем.

Обидві моделі мають стабільні API, SDK і широку підтримку в інструментах моніторингу/білінгової аналітики. У Gemini є AI Studio з прозорим прайсингом; у OpenAI – стандарт індустрії з інтеграцій [44].

2.3.2.2 Метрики для порівняння результатів моделей

Для того, щоб порівняти результати моделей з оцінками викладача було використано наступні метрики MAE – Mean Absolute Error та RMSE – Root Mean Squared Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a_i - b_i|\,, \tag{2.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - b_i)^2}, \tag{2.2}$$

де a_i – оцінка вчителя;

b_i – оцінка моделі;

n – кількість експериментів.

2.3.2.3 Розробка та тестування промптів

Розробка промптів (текстових інструкцій) є ключовим етапом створення системи автоматичного оцінювання SQL-скриптів із використанням великих мовних моделей (LLM) [45, 46]. Від правильності структури, точності формулювань і відповідності контексту завдання залежить якість результатів, які формує модель. Тому у межах роботи було проведено серію експериментів із проєктування, оптимізації та тестування промптів для моделей GPT-3.5-Turbo та Gemini-2.0-Flash.

Метою етапу було створення таких формулювань інструкцій, які дозволяють моделі:

– правильно розуміти структуру бази даних та умови завдання;

– аналізувати SQL-запити студентів на рівні семантики, а не лише синтаксису;

– формувати об’єктивну оцінку з урахуванням якості виконання завдання;

– надавати коментар українською мовою у форматі JSON, придатному для автоматичної обробки системою.

У кожному промпті використовувалася однакова логічна структура, що містила такі елементи:

– опис структури бази даних, необхідний для контексту оцінювання;

– формулювання завдання, яке мав виконати SQL-запит;

– еталонний (викладацький) SQL-скрипт, який відповідає правильному рішенню;

– студентський SQL-скрипт, що підлягає оцінюванню.

– правила оцінювання, за якими модель мала виставити одну з оцінок: «1» – запит повністю відповідає завданню; «0,7» – запит містить незначні недоліки (помилки у сортуванні, некоректні псевдоніми, зайві колонки тощо); «0» – запит не відповідає завданню або дає зовсім інший результат.

Для забезпечення стабільності виводу було також визначено чітку структуру JSON-відповіді з полями mark (оцінка) і comment (коментар українською мовою).

У процесі роботи було розроблено та протестовано чотири варіанти промптів, які відрізнялися за мовою інструкцій, рівнем формалізації та точністю структури [47].

Варіант 1 – базовий промпт українською мовою. Містив загальні інструкції та короткий опис правил оцінювання (рис. 2.2). Модель мала оцінити SQL-запит за трьома рівнями (0 / 0,7 / 1). Цей промпт використовувався як еталонний для первинної перевірки якості моделей. Проблемою виявилася деяка неточність у розумінні синтаксису української мови та варіативність у коментарях.

```
original_template_ua = '''
Ти виконуєш автоматичне оцінювання знань студентів мови SQL,
тобі дають:
1) структуру реляційної бази даних:
  {}
2) завдання (що має скрипт отримати):
  {}
3) оператори, які треба використовувати або не використовувати студенту під час написання скрипта:
  {}
4) еталонний (правильний)-скрипт:
  {}
5) скрипт-відповідь студента:
  {}

Тобі треба прокоментувати відповідь студента, наскільки вона правильна, чи відповідає вона завданню і
поставити оцінку в діапазоні від 0 до 1 за таким принципом:
- оцінка 1, якщо в результаті виконання скрипта студента:
а) буде виконуватися поставлене завдання (пункт 1);
б) у результаті еталонного скрипта і скрипта студента вийде однаковий набір кортежів (збігається
кількість рядків і стовпців, а значення клітинок збігаються);
в) буде використано необхідний оператор (пункт 3);
- оцінка 0.7:
а) буде виконуватися поставлене завдання (пункт 1);
б) у результаті еталонного скрипта і скрипта студента вийде однаковий набір кортежів (збігається
кількість рядків і стовпців, а значення клітинок співпадуть);
в) буде НЕ використано необхідний оператор (пункт 3);
- оцінка 0: якщо в результаті виконання еталонного скрипта і скрипта студента НЕ вийде однаковий
набір кортежів
Постав, будь ласка, оцінку в бальній системі від 0 до 1 і дай коментар до відповіді студента щодо
правильності та відповідності завданню
Формат відповіді має бути у json з полями mark, comment. Коментар повинен бути українською. Не використовуй markdown.
'''
```

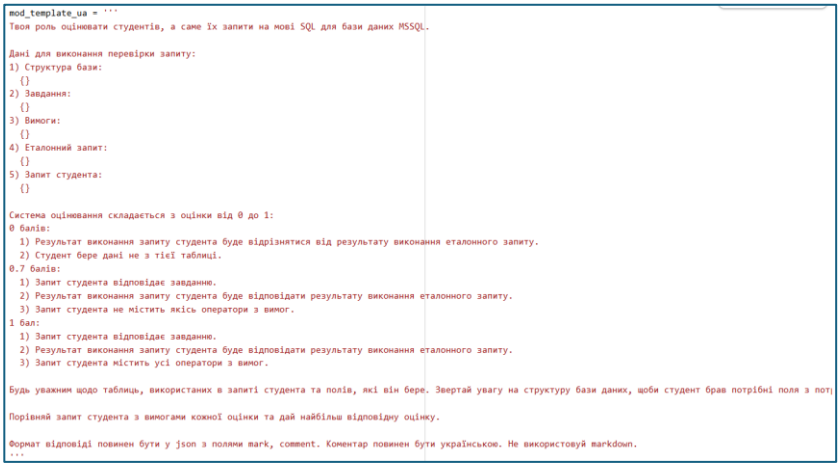
Рисунок 2.2 – Перший варіант промпту

Варіант 2 – переклад першого варіанта англійською мовою (рис 2.3). Дозволив підвищити точність розуміння завдання моделями. Проте деякі відповіді містили коментарі англійською, всупереч вимозі – українською. У цілому цей варіант показав найкращі результати за якістю оцінювання, особливо при використанні моделі Gemini-2.0-Flash.

```
original_template_en = '''
You perform an automated assessment of students' knowledge of the SQL language,
you are given:
1) the structure of a relational database:
  {}
2) the assignment (what the script should retrieve):
  {}
3) the operators the student should use or not use when writing the script:
  {}
4) reference (correct)-script:
  {}
5) student's script-response:
  {}

You need to comment on the student's answer, how correct it is, whether it matches the assignment and
give a grade between 0 and 1 based on the following principle:
- grade 1 if, as a result of the student's script:
a) the assigned task (point 1) will be fulfilled;
b) the reference script and the student's script will result in the same set of tuples (the same number of rows and columns, the same
number of rows and columns, and cell values will match);
c) the required operator (item 3) will be used;
- score 0.7:
a) the task will be performed (item 1);
b) the reference script and the student's script will result in the same set of tuples (the same
number of rows and columns, and cell values will match);
c) the required operator will NOT be used (point 3);
- score 0: if the execution of the reference script and the student's script does NOT result in the same set of tuples (the number of rows and columns is the
set of tuples
Please give a score in the point system from 0 to 1 and comment on the student's answer regarding
correctness and compliance with the assignment
The format of the answer must be in json with mark, comment fields. Comment should be in ukrainian.
'''
```

Рисунок 2.3 – Другий варіант промпту

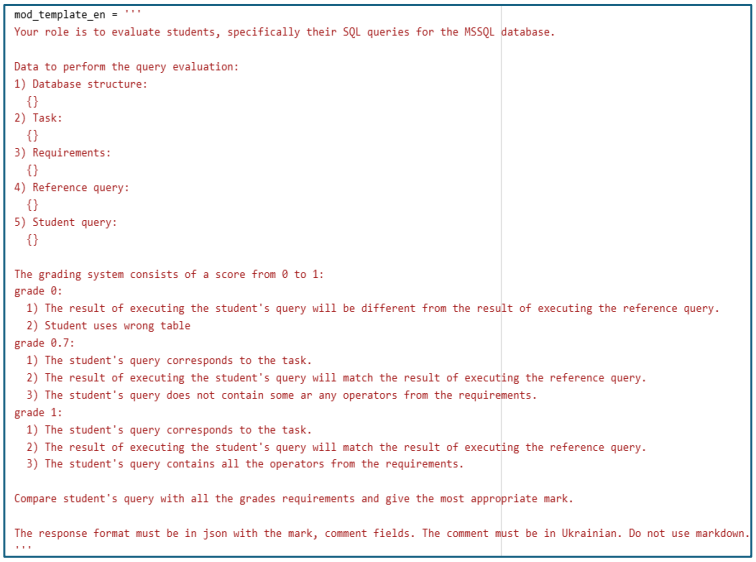
Варіант 3 – уточнений і розширений промпт українською мовою (рис. 2.4). Містив детальнішу рольову інструкцію («Твоя роль – оцінювати студентів…»), конкретизовані критерії для кожного типу оцінки та суворі вимоги до формату JSON. Це зменшило кількість випадків неправильного форматування, проте дещо погіршило узгодженість оцінок між моделями.



```
mod_template_ua = '''
Твоя роль оцінювати студентів, а саме їх запити на мові SQL для бази даних MSSQL.

Дані для виконання перевірки запиту:
1) Структура бази:
  {}
2) Завдання:
  {}
3) Вимоги:
  {}
4) Еталонний запит:
  {}
5) Запит студента:
  {}

Система оцінювання складається з оцінки від 0 до 1:
0 балів:
  1) Результат виконання запиту студента буде відрізнятися від результату виконання еталонного запиту.
  2) Студент бере дані не з тієї таблиці.
0.7 балів:
  1) Запит студента відповідає завданню.
  2) Результат виконання запиту студента буде відповідати результату виконання еталонного запиту.
  3) Запит студента не містить якісь оператори з вимог.
1 бал:
  1) Запит студента відповідає завданню.
  2) Результат виконання запиту студента буде відповідати результату виконання еталонного запиту.
  3) Запит студента містить усі оператори з вимог.

Будь уважним щодо таблиць, використаних в запиті студента та полів, які він бере. Звертай увагу на структуру бази даних, щоби студент брав потрібні поля з пот

Порівняй запит студента з вимогами кожної оцінки та дай найбільш відповідну оцінку.

Формат відповіді повинен бути у json з полями mark, comment. Коментар повинен бути українською. Не використовуй markdown.
'''
```

Рисунок 2.4 – Третій варіант промпту

Варіант 4 – англомовна версія третього варіанта (рис 2.5). Цей варіант показав високу стабільність для моделі Gemini, але незначно гірші показники точності порівняно з варіантом 2. Він став оптимальним компромісом між структурованістю та швидкістю генерації відповіді.



```
mod_template_en = '''
Your role is to evaluate students, specifically their SQL queries for the MSSQL database.

Data to perform the query evaluation:
1) Database structure:
  {}
2) Task:
  {}
3) Requirements:
  {}
4) Reference query:
  {}
5) Student query:
  {}

The grading system consists of a score from 0 to 1:
grade 0:
  1) The result of executing the student's query will be different from the result of executing the reference query.
  2) Student uses wrong table
grade 0.7:
  1) The student's query corresponds to the task.
  2) The result of executing the student's query will match the result of executing the reference query.
  3) The student's query does not contain some ar any operators from the requirements.
grade 1:
  1) The student's query corresponds to the task.
  2) The result of executing the student's query will match the result of executing the reference query.
  3) The student's query contains all the operators from the requirements.

Compare student's query with all the grades requirements and give the most appropriate mark.

The response format must be in json with the mark, comment fields. The comment must be in Ukrainian. Do not use markdown.
'''
```

Рисунок 2.5 – Четвертий варіант промпту

Для оцінювання ефективності промптів було проведено тестування серед студентів, у якому зібрано 270 SQL-відповідей. Кожен SQL-скрипт оцінювався двома моделями – GPT-3.5-Turbo та Gemini-2.0-Flash.

На підставі отриманих даних для кожної версії промпта були розраховані показники MAE (Mean Absolute Error) і RMSE (Root Mean Square Error) для кращого аналізу оцінювання відповідей моделей на відповідь студента, використовуючи певний варіант промпта. Зведені результати представлені в таблиці 2.2.

Таблиця 2.2 – Точність оцінювання SQL скриптів моделями

Варіант промпту	**Модель**	**MAE**	**RMSE**
Варіант 1	GPT-3.5-Turbo	0,23	0,36
Варіант 1	Gemini-2.0-flash	0,14	0,30
Варіант 2	GPT-3.5-Turbo	0,22	0,33
Варіант 2	Gemini-2.0-flash	0,12	0,28
Варіант 3	GPT-3.5-Turbo	0,26	0,42
Варіант 3	Gemini-2.0-flash	0,15	0,30
Варіант 4	GPT-3.5-Turbo	0,24	0,39
Варіант 4	Gemini-2.0-flash	0,14	0,29

З результатів, що зазначені в таблиці 2.2 можна сказати, що окремо взяті моделі показали різну чутливість до структури промптів:

- Gemini-2.0-flash демонструвала послідовно менші помилки (MAE = 0,12–0,15, RMSE = 0,28–0,30), що вказує на її вищу стійкість і точність при інтерпретації формальних інструкцій;
- GPT-3.5-Turbo, навпаки, проявила більшу варіативність (MAE = 0,22–0,26, RMSE = 0,33–0,42), особливо при використанні складних або промптів мова яких не збігалася з необхідною мовою виведення інформації (мова промпта: англійська, мова відповіді: українська).

Для кожного студентського запиту виконувалася паралельна оцінка двома моделями, після чого їхні результати порівнювалися та агрегувалися. У разі, якщо розбіжність між оцінками моделей перевищувала 20 %, система позначала

результат як невпевнений, і він направлявся на повторну перевірку викладачеві. Якщо ж розбіжність була в допустимих межах, обчислювалося середнє значення, яке ставало підсумковою оцінкою студента. Точність оцінювання на основі усереднення оцінок від моделей GPT-3.5-Turbo та Gemini-2.0-flash наведена у таблиці 2.3.

Таблиця 2.3 – Точність оцювання SQL скриптів на основі усереднення оцінок від моделей GPT-3.5-Turbo та Gemini-2.0-flash

Варіант промпту	**MAE**	**RMSE**
Варіант 1	0.18	0.28
Варіант 2	0.17	0.27
Варіант 3	0.20	0.33
Варіант 4	0.19	0.31

Також було підраховано відсоток випадків, коли розбіжність між оцінками моделей була більше 20%, та було отримано наступні значення:

- варіант 1 – 35% випадків;
- варіант 2 – 23% випадків;
- варіант 3 – 25% випадків;
- варіант 4 – 24% випадків.

Однак при спільному застосуванні обох моделей ці відмінності були використані як взаємодоповнюючі переваги: Gemini забезпечувала стабільність, а GPT посилювала інтерпретаційну гнучкість.

Також аналіз даних, наведених у таблиці 2, показав, що використання комбінованого підходу Gemini + GPT забезпечило високу стабільність і точність оцінювання студентських SQL-скриптів. Результати свідчать про те, що найнижчі значення середньої абсолютної помилки (MAE = 0,12) та середньоквадратичної помилки (RMSE = 0,28) були отримані саме для другого варіанта промпта, який характеризувався збалансованою структурою інструкцій та чітким формулюванням вимог англійською мовою.

Також було встановлено, що спільна робота моделей Gemini-2.0-flash і GPT-3.5-Turbo дала змогу зменшити кількість значних розбіжностей між автоматичними та викладацькими оцінками.

Комбінація Gemini та GPT використовувалася за принципом взаємного контролю:

- обидві моделі незалежно формували оцінку та коментар у форматі JSON;
- система обчислювала різницю між їхніми результатами;
- якщо відхилення ≤ 20 %, обчислювалося середнє значення оцінки;
- якщо відхилення > 20 %, результат позначався як сумнівний і вимагав перевірки викладача;
- за результатами та переглядом відповідей від моделей було вирішено брати коментар від моделі Gemini, оскільки коментарі, що було надані неї мають більш інформативний зміст ніж коментарі від моделі GPT.

Такий підхід дозволив:

- знизити ймовірність хибнопозитивних оцінок (коли модель помилково ставить високий бал за некоректний запит);
- підвищити достовірність фінальних оцінок, оскільки рішення приймається на основі узгодженості двох незалежних моделей;
- виявляти нестандартні випадки, де моделі розходяться в судженнях, що корисно для аналізу складності завдань.

Також результати показали, що якість оцінювання сильно залежить від структури та мови промпту:

– варіант 1 (базовий, з мінімальною структурою) показав помірні результати (MAE ≈ 0,23 для GPT і 0,14 для Gemini). Варіант промпту номер 1 зображено на рисунку 2.2;

- варіант 2 (переклад першого варіанту англійською мовою) продемонстрував найкращі результати, особливо в поєднанні Gemini + GPT (MAE середн. ≈ 0,17, RMSE ≈ 0,30);

− варіант 3 (покращений з уточненням ролі моделі та формалізацією критеріїв) погіршив точність обох моделей (RMSE до 0,42 у GPT);

− варіант 4 (переклад третього варіанту англійською мовою) дав стабільний, але не найкращий результат.

Отже, надмірне ускладнення формулювання промпта знижує узгодженість роботи моделей. Найбільш надійним виявився другий варіант промпту (переклад першого варіанту англійською мовою), тому було вирішено використовувати його як остаточний промпт для оцінювання на основі декількох LLMs.

2.3.2.4 Алгоритм на основі декількох LLMs

Для підвищення точності, стійкості та об'єктивності автоматичного оцінювання SQL-скриптів у системі тестування знань студентів було розроблено алгоритм на основі декількох великих мовних моделей (LLM). В його основі лежить ідея мультиагентного аналізу, при якому дві незалежні моделі – Gemini і GPT-3.5-Turbo – паралельно оцінюють одне і те ж рішення, а потім їх результати агрегуються в єдину підсумкову оцінку. Такий підхід дозволяє компенсувати індивідуальні особливості кожної моделі, зменшити вплив випадкових помилок і підвищити довіру до підсумкових результатів.

Кожна LLM, незважаючи на загальні архітектурні принципи, має власні мовні та семантичні особливості:

– GPT-3.5-Turbo відрізняється високою стабільністю, точністю дотримання інструкцій і передбачуваністю структури висновку, що робить її надійною при формалізованих підказках і фіксованому форматі відповіді (JSON);

– Gemini демонструє краще розуміння контексту, гнучкість в інтерпретації та здатність помічати логічні невідповідності, особливо в текстах на природних мовах, але може бути менш суворим у форматі.

Використовуючи ці моделі було розроблено алгоритм:

Крок 1. На вхід обом моделям подаються однакові параметри:

– структура бази даних (databaseScript);

– тестові дані (dataScript);

– умова завдання (question);

– вимоги до використання операторів (requirements);

– еталонний SQL-запит (etalonAnswer);

– SQL-запит, написаний студентом (studentAnswer).

Всі текстові значення попередньо декодуються з Base64.

Крок 2. Використання заготовленого промпта.

Крок 3. Паралельна оцінка. Обидві моделі – Gemini і GPT-3.5-Turbo – отримують однаковий промпт і повертають відповідь у форматі JSON.

Крок 4. Нормалізація відповідей. Після отримання відповідей система перевіряє коректність JSON-структури і при необхідності виправляє дрібні синтаксичні помилки (наприклад, відсутні лапки або коми). Це дозволяє уникнути відмови при подальшій обробці.

Крок 5. Після нормалізації обидві відповіді аналізуються. Алгоритм обчислює фактор довіри (confidence) – обернено пропорційний розбіжності

$$confidence = 1 - |mark_{GPT} - mark_{Gemini}|. \quad (2.3)$$

Крок 6. Передача результату системі та користувачеві.

– якщо confidence > 0.8 – оцінка приймається автоматично;

– якщо confidence ≤ 0.8 – система позначає результат як «потребує перевірки» та повідомляє викладача.

Алгоритм, що використовує кілька LLM, виконує функцію взаємної верифікації результатів. Його впровадження вирішує відразу кілька завдань:

– усуває залежність від однієї моделі та знижує ризик систематичних помилок;

– підвищує достовірність оцінки за рахунок перехресної перевірки;

– дозволяє аналізувати, в яких випадках думки моделей розходяться – що особливо корисно для наукового аналізу та подальшого вдосконалення промптів.

2.4 Метод гнучкої оцінки на основі AI

2.4.1 Гнучке оцінювання на основі готових LLMs

Для підвищення точності та інформативності процесу автоматичного оцінювання SQL-скриптів у системі було реалізовано гнучке оцінювання на основі готових великих мовних моделей (LLM). На відміну від грубого оцінювання, де модель видає одну підсумкову оцінку за принципом 0 / 0,7 / 1, гнучка система дозволяє проводити детальний аналіз рішення студента за кількома критеріями, що робить результат перевірки більш об'єктивним і зрозумілим як для викладача, так і для студента.

Гнучке оцінювання побудовано на основі заздалегідь підготовлених промптів, спеціально адаптованих для завдань аналізу SQL-запитів. Ці промпти формують для LLM чітку інструкцію, де визначається контекст завдання (структура бази даних, умова, вимоги, еталонний і студентський запит), а також система критеріїв, за якими модель повинна виставити оцінки.

Кожен запит оцінюється за шістьма параметрами:

– syntax: відсутність синтаксичних помилок у коді;

– correct_table: правильність вибору таблиць;

– correct_attributes: коректність вибору полів;

– result: збіг результату виконання з еталонним запитом;

– requirements: виконання вимог, зазначених у завданні.

Кожен пункт оцінюється окремо в діапазоні від 0 до 1, а ключові критерії (наприклад, відповідність результату і виконання вимог) мають підвищену вагу – до 3 балів, що дозволяє моделі акцентувати увагу на найбільш важливих аспектах рішення.

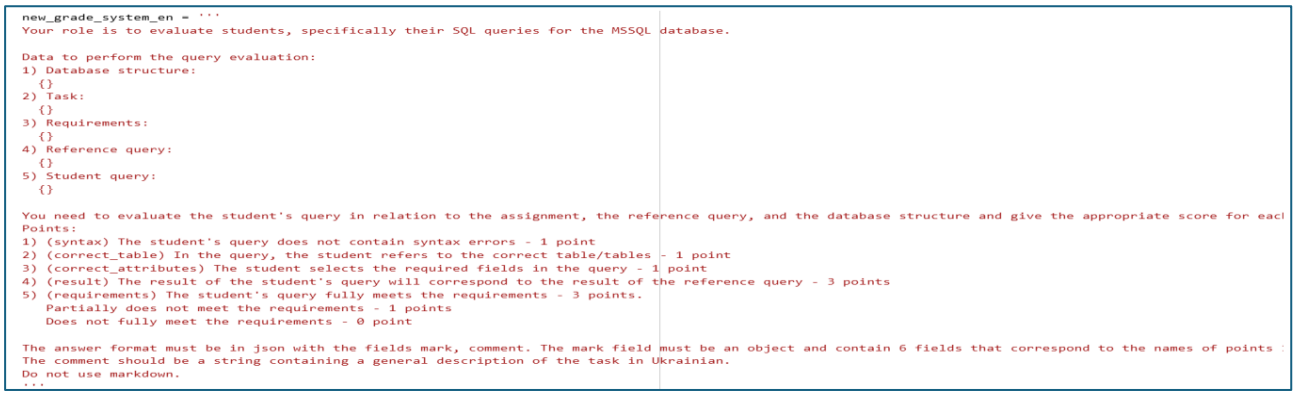
Завдяки такому формату система отримує структурований результат, який легко піддається автоматичній обробці та аналізу. Це також дозволяє викладачеві бачити не тільки кінцеву оцінку, але й те, в яких саме аспектах студент припустився помилок.

Промпти для гнучкого оцінювання були спеціально розроблені таким чином, щоб мінімізувати вплив випадкових інтерпретацій моделі та забезпечити відтворюваність відповідей. В інструкціях чітко вказано, що модель повинна відповідати строго у форматі JSON, не використовувати Markdown і формулювати коментар українською мовою. Це гарантує одноманітність структури висновку і полегшує інтеграцію результатів в автоматизовану систему тестування.

Використання гнучкого оцінювання дозволило перейти від простої перевірки результату запиту до багаторівневого аналізу, що включає синтаксичну, логічну і контекстну оцінку рішення. Таким чином, система не просто визначає, чи збігається результат з еталонним, але й виявляє характер помилок: неправильний вибір таблиці, відсутність потрібних операторів або часткова невідповідність вимогам.

Крім того, гнучке оцінювання дало можливість формувати персоналізований зворотний зв'язок для студента. Замість формальної оцінки студент отримує коментар, що пояснює, що саме в його запиті було виконано правильно, а які елементи вимагають виправлення. Це робить процес перевірки не тільки автоматизованим, але й навчальним.

За попередніми результатами грубого оцінювання було вирішено, що для гнучкого оцінювання найліпше підійде промпт англійської мовою, так як обидві моделі попередньо показали ліпші результати, коли їм задаєш промпт англійською мовою. Промпт для гнучкого оцінювання зображено на рисунку 2.6.



```
new_grade_system_en = '''
Your role is to evaluate students, specifically their SQL queries for the MSSQL database.

Data to perform the query evaluation:
1) Database structure:
  {}
2) Task:
  {}
3) Requirements:
  {}
4) Reference query:
  {}
5) Student query:
  {}

You need to evaluate the student's query in relation to the assignment, the reference query, and the database structure and give the appropriate score for eac
Points:
1) (syntax) The student's query does not contain syntax errors - 1 point
2) (correct_table) In the query, the student refers to the correct table/tables - 1 point
3) (correct_attributes) The student selects the required fields in the query - 1 point
4) (result) The result of the student's query will correspond to the result of the reference query - 3 points
5) (requirements) The student's query fully meets the requirements - 3 points.
   Partially does not meet the requirements - 1 points
   Does not fully meet the requirements - 0 point

The answer format must be in json with the fields mark, comment. The mark field must be an object and contain 6 fields that correspond to the names of points
The comment should be a string containing a general description of the task in Ukrainian.
Do not use markdown.
'''
```

Рисунок 2.6 – Промпт для гнучкої системи оцінювання

2.4.2 Аналіз результатів гнучкого оцінювання

Після успішного тестування промптів для строгого оцінювання SQL-запитів, у межах подальших експериментів було вирішено перевірити ефективність гнучкого алгоритму оцінювання. Для цього було використано ті самі 270 студентських відповідей, які застосовувалися в попередньому експерименті з промптами для грубого оцінювання.

Метою цього етапу було перевірити, наскільки великі мовні моделі можуть точно оцінювати студентські SQL-запити за декількома критеріями одночасно, а саме:

– наявність синтаксичних помилок (syntax);

– правильність використання таблиць (correct_table);

– правильність вибору атрибутів (correct_attributes);

– відповідність результату еталонному запиту (result);

– дотримання вимог завдання (requirements).

Для реалізації гнучкої системи оцінювання було застосовано ті ж дві моделі, які показали найкращі результати в попередніх дослідженнях – GPT-3.5-Turbo та Gemini-2.0-Flash.

Кожна модель отримувала у вхідних даних структуру бази даних, текст завдання, еталонний запит і студентський запит, після чого формувала JSON-відповідь із полями:

– mark: об’єкт, що містив оцінку за шістьма критеріями;

– comment: текстовий коментар українською мовою з описом виявлених недоліків.

Фінальна оцінка розраховувалася як середнє зважене значення усіх шести критеріїв (максимум 10 балів).

Однак, у ході практичного тестування виявилося, що моделі часто плуталися під час визначення ваги кожного критерію, що призводило до спотворення загальної оцінки.

Попри детальну структуру промпта і чітко визначені критерії, результати гнучкого оцінювання виявилися менш точними у порівнянні з моделями строгого оцінювання.

Основною проблемою було те, що моделі не завжди узгоджували часткові оцінки між собою – наприклад, могли виставити високий бал за «syntax», але низький за «result», навіть якщо студентський запит давав правильний вихід.

Також моделі демонстрували чутливість до довжини запиту: при складних SQL-конструкціях із підзапитами вони часто переоцінювали або недооцінювали окремі критерії.

У таблиці 2.4 наведено результати точності моделей GPT-3.5-Turbo та Gemini-2.0-Flash у режимі гнучкого оцінювання за метриками MAE і RMSE відносно оцінок викладача.

Таблиця 2.4 – Показники точності моделей при гнучкому оцінюванні

Модель	**MAE**	**RMSE**
GPT-3.5-Turbo	0,3	0,45
Gemini-2.0-flash	0,25	0,38

Як видно з таблиці, похибки MAE і RMSE є значно більшими, ніж при строгому оцінюванні (де MAE не перевищував 0.17–0.23, а RMSE залишався в

межах 0.28–0.36). Особливо велика варіативність спостерігалася у GPT-3.5-Turbo, яка в окремих випадках демонструвала надмірно «оптимістичні» оцінки навіть при очевидних помилках у запитах.

Після обчислення результатів для обох моделей було проведено агрегацію оцінок – шляхом усереднення результатів GPT і Gemini для кожної студентської відповіді (табл. 2.5). Цей метод дозволив дещо знизити коливання похибок, однак загальна точність залишалася нижчою, ніж при строгому оцінюванні.

Таблиця 2.5 – Середні результати комбінованого оцінювання

Тип оцінювання	**MAE**	**RMSE**	**Відсоток відхилень > 20%**
Гнучке оцінювання	0.24	0.40	32%
Грубе оцінювання	0.17	0.27	23%

Як показують дані таблиці, гнучке оцінювання спричинило збільшення середньої похибки приблизно на 40% порівняно зі строгим. Крім того, кількість випадків, коли оцінка моделі відрізнялася від оцінки викладача більш ніж на 20%, зросла з 23% до 32%.

Результати експерименту підтвердили, що гнучке оцінювання є значно складнішим завданням для сучасних великих мовних моделей. Попри потенційну перевагу у багатокритеріальності, така система має низку недоліків:

– висока чутливість до контексту завдання: моделі не завжди коректно визначали, які критерії є ключовими;

– схильність до плутанини у часткових оцінках: через нечітку вагу критеріїв підсумковий бал часто не відображав реальну якість SQL-запиту;

– зростання кількості відхилень: у понад 30% випадків оцінки моделей відрізнялися від експертної думки викладача більш ніж на 20%.

Таким чином, експерименти показали, що гнучке оцінювання потребує додаткового доопрацювання, тому було вирішено продовжити використання алгоритму грубого оцінювання.

2.5 Вибір остаточного алгоритму

На основі проведених досліджень, тестувань і порівняльного аналізу різних підходів було прийнято рішення використовувати комбінований алгоритм, який об'єднує грубе оцінювання на основі LLM-моделей і евристичний алгоритм структурного аналізу SQL-запитів.

Грубе оцінювання виконується за допомогою великих мовних моделей (GPT-3.5-Turbo і Gemini-2.0-Flash) і забезпечує контекстне розуміння завдання, інтерпретацію логіки запиту, а також формування осмислених коментарів природною мовою. Ці моделі здатні враховувати різні варіанти коректних рішень, визначати часткову правильність (оцінки 0 / 0,7 / 1) і виявляти помилки у формулюваннях, іменах полів і порядку вибірки даних.

Евристичний алгоритм, у свою чергу, виконує попередню перевірку SQL-скрипта: аналізує структуру запиту, коректність синтаксису, відповідність результату студентського скрипта та еталонного скрипта. Він служить фільтром для виключення технічних помилок (наприклад, відсутніх таблиць, синтаксичних збоїв, неправильних аліасів).

У підсумковому підході передбачено два етапи роботи:

Етап 1. Евристичний аналіз – швидка перевірка синтаксису та структури скрипта, формування первинної оцінки та виявлення потенційних помилок.

Етап 2. LLM-оцінювання – глибокий семантичний аналіз скрипта з урахуванням завдання, формування остаточної оцінки та коментаря.

Якщо результати двох методів збігаються (або відрізняються не більше ніж на 20 %), система приймає усереднене значення оцінки. При істотних

розбіжностях результат позначається як «невпевнений» і направляється викладачеві для перевірки.

Такий комбінований підхід забезпечує:

– стійкість до синтаксичних і логічних помилок;

– підвищення точності за рахунок врахування контексту завдання;

– зменшення кількості помилкових оцінок;

– можливість часткової автоматизації перевірки без втрати достовірності.

Завдяки поєднанню двох взаємодоповнюючих методів – евристичного та LLM-підходу – система здатна враховувати як формальні, так і контекстуальні аспекти SQL-запиту, що істотно знижує ризик пропуску критичних помилок. Евристичні правила виконують функцію «структурного фільтра», забезпечуючи виявлення помилок, пов’язаних із кількістю рядків, стовпців, відповідністю назв атрибутів або непослідовністю результатів, тоді як LLM-моделі дозволяють оцінити глибші логічні залежності, коректність задуму та відповідність сценарію виконання завданню.

Таким чином, остаточним рішенням для прототипу системи обрано дворівневий алгоритм, що поєднує евристичний і LLM-підходи до оцінювання SQL-запитів, що забезпечує баланс між швидкістю, точністю та інтерпретованістю результатів.

3 ВПРОВАДЖЕННЯ ЗАПРОПОНОВАНИХ АЛГОРИТМІВ У СИСТЕМУ ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ

3.1 Опис функціональності застосунка

Вебзастосунок, що розробляється, являє собою систему тестування знань студентів з мови SQL з можливістю автоматичного оцінювання відповідей на основі евристичних правил і великих мовних моделей (LLM), таких як GPT-3.5-Turbo і Gemini. Основна мета додатка – підвищити об'єктивність і швидкість перевірки практичних SQL-завдань, зменшивши навантаження на викладачів і забезпечивши прозорий зворотний зв'язок зі студентами.

Застосунок підтримує чотири типи користувачів:

– «Адміністратор» керує всіма аспектами системи, створює користувачів, тести та керує розкладом;

– «Викладач» формує тести, призначає їх студентам, аналізує результати та за необхідності коригує оцінки, якщо система позначила відповідь як «сумнівну»;

– «Студент» проходить тести (рис. 3.1), пише SQL-скрипти у відповідь на завдання і отримує автоматичні оцінки та коментарі;

– «Гість» (звичайний користувач) може виконувати публічно доступні тести без реєстрації.

Система підтримує режим автоматичного оцінювання: суворе оцінювання – орієнтоване на перевірку фактичної коректності запиту і може виконуватися як евристичним алгоритмом, так і LLM-моделлю за фіксованою шкалою 0 / 0,7 / 1. У цьому режимі оцінюється відповідність результату запиту завданню, збіг з еталоном і наявність необхідних операторів.

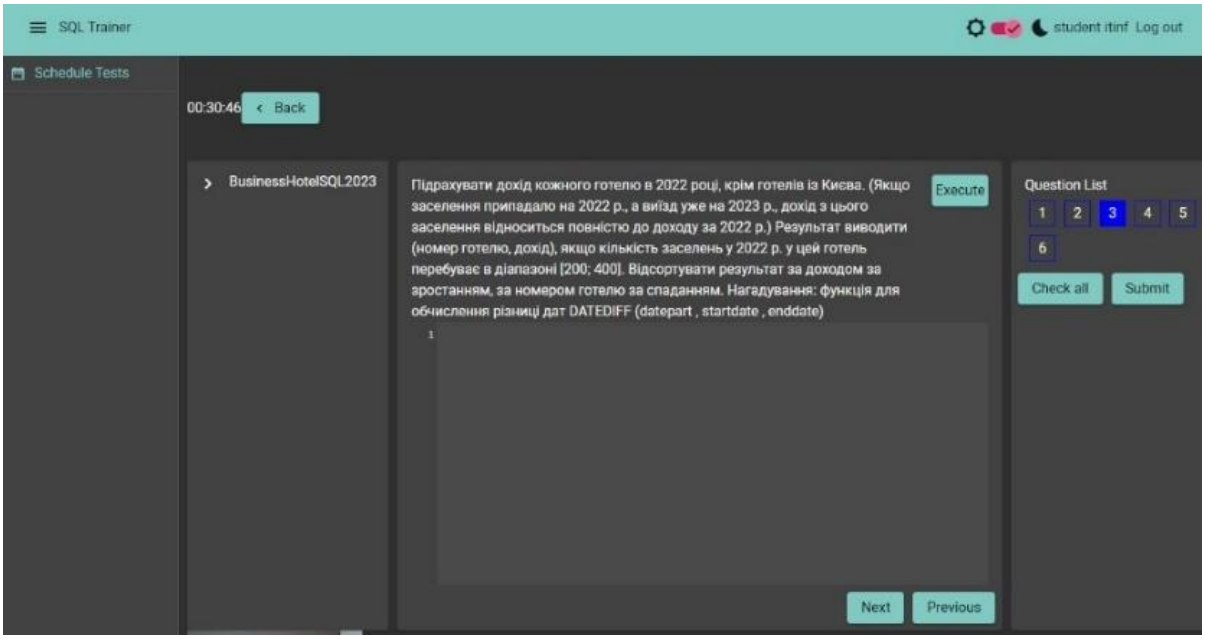


Рисунок 3.1 - Інтерфейс проходження тесту

3.2 Опис програмного забезпечення для застосунка

Застосунок побудований за клієнт-серверною архітектурою.

Клієнтська частина розроблена на Angular 15, забезпечуючи адаптивний та інтуїтивно зрозумілий інтерфейс, а серверна частина реалізована з використанням .NET 8 та Python 3.

В якості СУБД використовується PostgreSQL, а взаємодія мікросервісів здійснюється за протоколом gRPC.

Основні сервіси системи: Users Service, Test Infos Service, Subject Area Service, Evaluation Service.

Users Service відповідає за управління користувачами, ролями, та правами доступа.

Test Infos Service відповідає за зберігання тестів, питань і результатів, а також автоматична перевірка відповідей.

Subject Area Service відповідає виконання SQL-скриптів студентів та управління тестовими базами даних.

Evaluation Service це ключовий модуль перевірки, що включає евристичний та LLM-механізми.

Для інтеграції з мовними моделями реалізовано API-адаптери до OpenAI GPT-3.5-Turbo та Google Gemini 1.5 Pro.

Модуль Gemini доповнює GPT-модель у випадках складних синтаксичних і контекстних завдань, забезпечуючи порівняльний аналіз двох незалежних моделей.

Результати їх роботи оцінюються за критерієм узгодженості – якщо розбіжність перевищує 20 %, відповідь позначається як «сумнівна» і вимагає ручної перевірки викладача.

3.3 Архітектура застосунку

Архітектура розробленого застосунку побудована з урахуванням принципів модульності, масштабованості та гнучкості, що дозволяє легко інтегрувати нові механізми оцінювання та адаптувати систему під різні навчальні сценарії [1, 48–53]. Основна мета архітектурного рішення полягає в тому, щоб забезпечити надійне виконання SQL-запитів студентів, їх коректну обробку та автоматичне оцінювання із застосуванням як евристичних методів, так і великих мовних моделей (LLM) [54].

Система реалізована за клієнт-серверною архітектурою (рис. 3.2) і включає в себе кілька функціонально незалежних модулів, що взаємодіють через API-інтерфейси. Такий підхід забезпечує стійкість додатка і можливість його масштабування при збільшенні кількості користувачів і обсягу даних.

Frontend написаний на Angular це клієнтська частина, що забезпечує взаємодію з користувачем. Інтерфейс розроблений таким чином, щоб студент міг зручно виконувати завдання, вводити SQL-запити та отримувати результати оцінювання, а викладач – створювати тести, аналізувати результати та керувати процесом навчання.

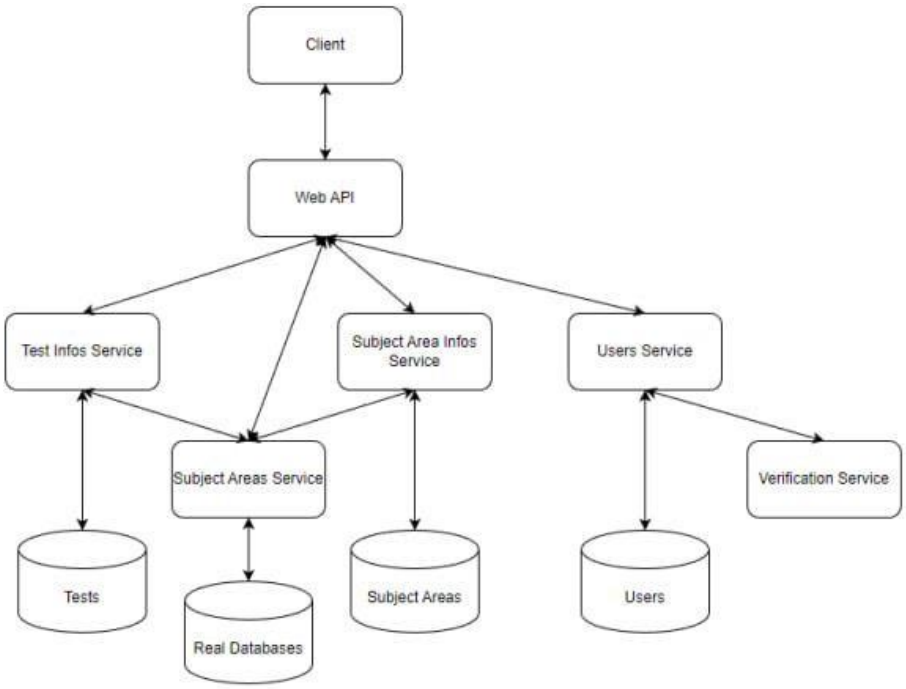


Рисунок 3.2 - Діаграма архітектури

Backend написаний на .NET 8 це серверна частина, що виконує роль центрального керуючого модуля [41]. Вона відповідає за маршрутизацію запитів, аутентифікацію користувачів, взаємодію з базою даних і виклик сервісів оцінювання.

Evaluation Service (Python/.NET) це модуль автоматичного оцінювання, що реалізує грубе оцінювання, що виконується як евристичним алгоритмом, так і за допомогою LLM (GPT і Gemini) за фіксованою шкалою 0 / 0,7 / 1.

LLM Integration Layer це проміжний шар для взаємодії із зовнішніми мовними моделями (OpenAI GPT-3.5-Turbo і Google Gemini 1.5 Pro). Він відповідає за формування промптів, обробку JSON-відповідей і контроль узгодженості результатів.

Database Layer (PostgreSQL) це рівень зберігання даних, що включає таблиці користувачів, тестів, завдань, структур баз даних, результатів виконання запитів і отриманих оцінок.

Взаємодія компонентів системи побудована наступним чином:

– клієнт відправляє запити на сервер з діями користувача (наприклад, виконання SQL-скрипта, перегляд результату, проходження тесту);

– сервер приймає запит, обробляє його і передає дані у відповідний сервіс. При виконанні SQL-завдання запит студента і еталонний запит направляються в Evaluation Service;

– сервіс оцінювання отримує вхідні дані – структуру бази даних, SQL-запити, вимоги та результати виконання скриптів;

– сервіс агрегування результатів порівнює оцінки від обох моделей. Якщо різниця між ними перевищує 20 %, то результат позначається як «сумнівний» і передається викладачеві для перевірки;

– після завершення аналізу оцінка і коментар зберігаються в базі даних і стають доступними користувачеві в інтерфейсі.

Архітектура додатка спроектована з можливістю інтеграції додаткових компонентів без значних змін існуючого коду.

Завдяки цьому передбачена можливість: додавання нових мовних моделей (наприклад, GPT-4 або Claude) через єдиний LLM Integration Layer, підключення інших алгоритмів аналізу SQL (наприклад, перевірка семантики або ефективності запиту) та інтеграції із зовнішніми освітніми платформами (через REST API або LTI).

Для обміну даними між сервісами застосовується протокол gRPC, що забезпечує високу швидкість і надійність передачі інформації.

Так як застосунок має мікросервісну архітектуру, то кожен сервіс має власну базу даних, що вимагає більш детального планування та дотримання правил нормалізації баз даних при проектуванні бази даних для кожного з вебсерверів, що складають вебзастосунок.

Users Service працює з базою даних користувачів, що складається з наступних таблиць: Users, Roles, Groups,

Users зберігає інформацію про користувачів, яка складається з унікального ідентифікатора, адреси електронної пошти, імені користувача, зашифрованого пароля, інформації про роль та групу користувача, якщо користувач має роль «Студент».

Roles зберігає інформацію про ролі користувачів у системі, яка складається з назви ролі та унікального ідентифікатора.

Groups зберігає інформацію про групи студентів, яка складається з унікального ідентифікатора групи та назви групи в системі.

Схема бази даних для Users Service зображена на рисунку 3.3.

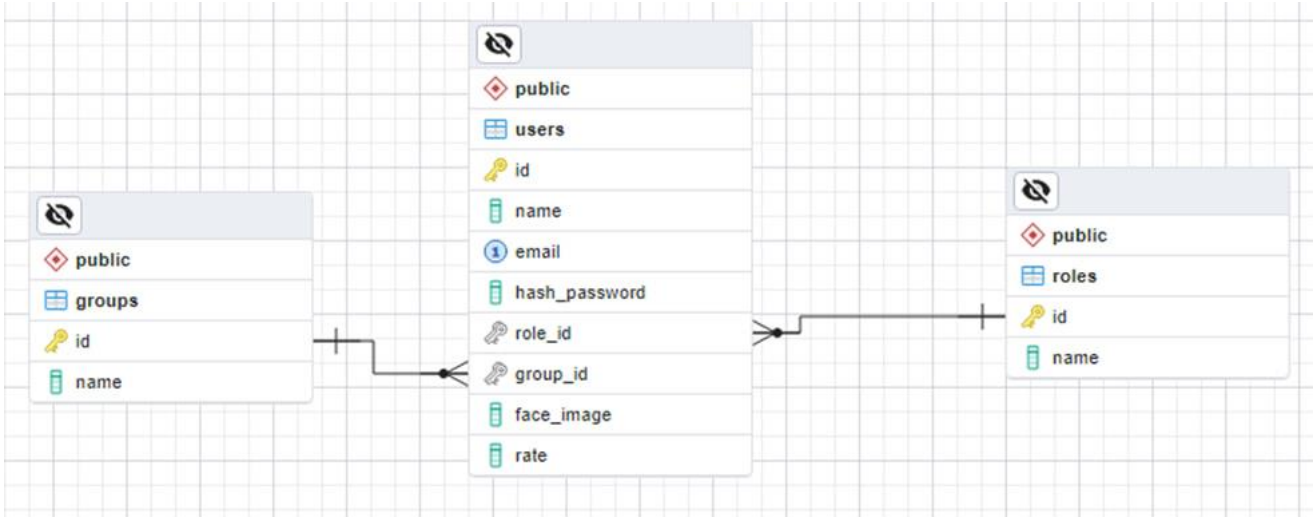


Рисунок 3.3 – Схема бази даних для Users Service

Сервіс Test Infos Service працює з базою даних тестів, яка включає наступні таблиці: Topics, Questions, Correct Answers, Tests, Test Questions, Schedule Tests, User Schedule Tests, User Answers.

Topics призначена для зберігання тем запитань; містить унікальний ідентифікатор теми та її назву.

Questions призначена для зберігання питань; включає унікальний ідентифікатор питання, текст самого питання та рівень його складності.

Correct Answers зберігає правильні відповіді на конкретні питання; містить унікальний ідентифікатор відповіді, текст відповіді та інформацію про те, до якого питання вона відноситься.

Tests відповідає за зберігання тестів; включає унікальний ідентифікатор тесту, його назву та дату створення.

Test Questions це проміжна таблиця між Questions і Tests, яка зберігає зв'язки між питаннями і конкретними тестами.

Schedule Tests зберігає інформації про заплановані тести; містить унікальний ідентифікатор розкладу, дату початку тесту, дату його завершення і посилання на відповідний тест.

User Schedule Tests зберігає відомості про те, який тест за розкладом призначений конкретному студенту; містить унікальний ідентифікатор запису, інформацію про тест та інформацію про студента.

User Answers призначена для зберігання даних про те, які відповіді студент дав на певні питання конкретного тесту за розкладом; містить інформацію про студента, про тест у розкладі, про питання, відповідь студента, а також оцінки системи та викладача.

Схема бази даних для Test Infos Service зображена на рисунку 3.4.

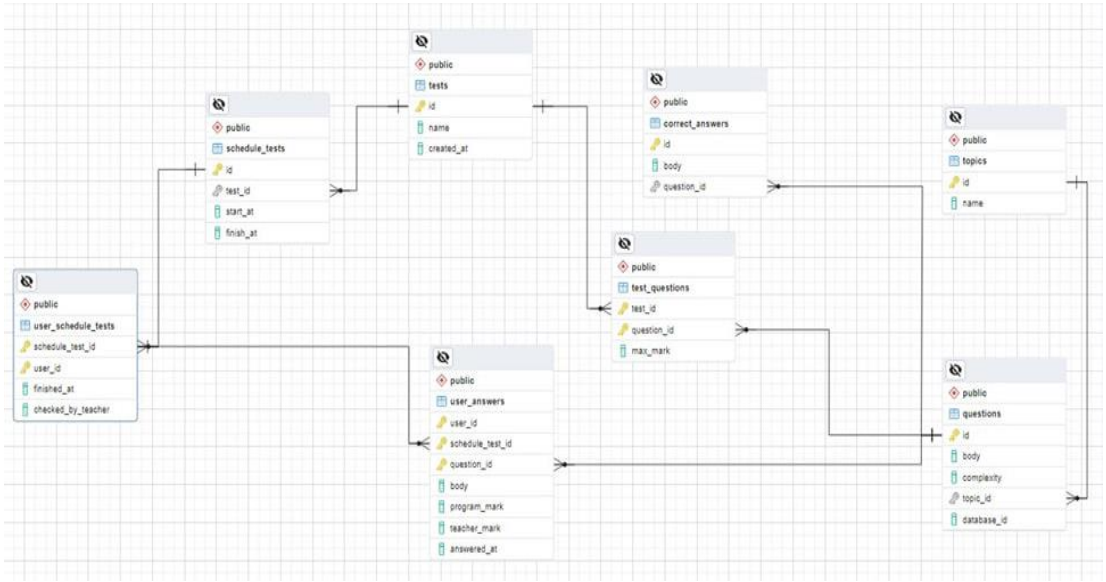


Рисунок 3.4 – Схема бази даних для Test Infos Service

Subject Area Infos Service працює з базою даних, яка зберігає інформацію про створені навчальні бази даних, всередині яких виконуються тестові завдання для студентів. База даних сервісу включає наступні таблиці: Languages, Databases, Tables, Attributes.

Languages – таблиця, призначена для зберігання відомостей про діалекти SQL. Містить унікальний ідентифікатор мови та її назву.

Databases – таблиця, яка зберігає інформацію про конкретні бази даних. Містить унікальний ідентифікатор бази даних, її назву, а також відомості про те, до якого SQL-діалекту вона належить.

Tables – таблиця для зберігання переліку таблиць, що входять до певної бази даних. Включає унікальний ідентифікатор таблиці, її назву та посилання на базу даних, до якої ця таблиця належить.

Attributes – таблиця, що містить інформацію про атрибути (стовпці) конкретної таблиці. У таблиці зберігаються унікальний ідентифікатор атрибута, його назва, тип даних, максимальна довжина (за необхідності), значення за замовчуванням та інформація про те, до якої таблиці належить даний атрибут.

Схема бази даних для Subject Area Infos Service зображена на рисунку 3.5.

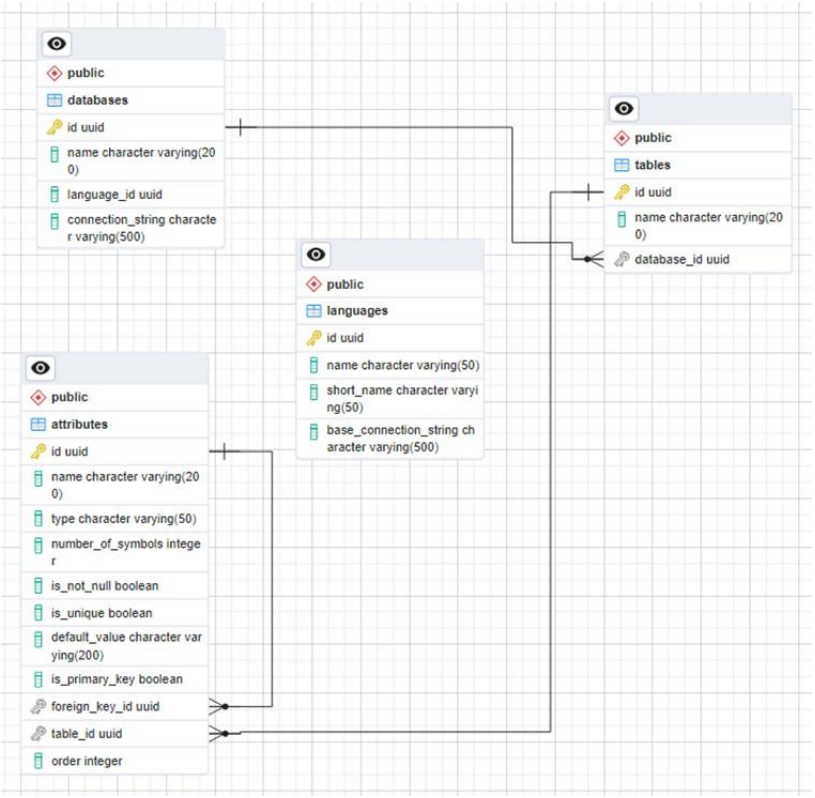


Рисунок 3.5 – Схема бази даних для Subject Area Infos Service

3.4 Ілюстрація роботи застосунка

Розглянемо роботу застосунку на основі створенні груп студентів, створенні тем та завдань, створенні тестів, проходження та перегляд результатів тесту.

3.4.1 Створення груп студентів

Для створення тесту та майбутнього його проходження необхідно створити групу для студентів та створити користувачів із роллю студент та закріпити цього користувача за створеною групою.

Створену групу під назвою «test group» зображено на рисунку 3.6.

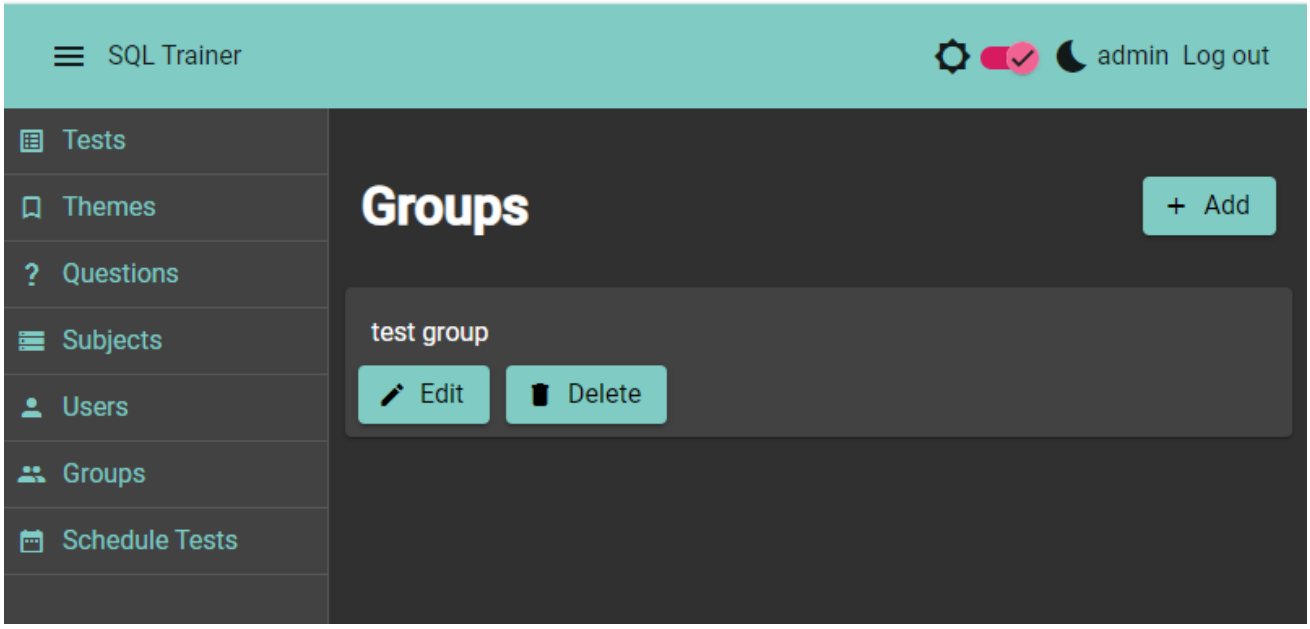


Рисунок 3.6 – Створена група «test group»

Створеного користувача, за котрого ми будемо проходити запланований вчителем тест, з ім’ям «test student» та роллю «Студент» зображено на рисунку 3.7.

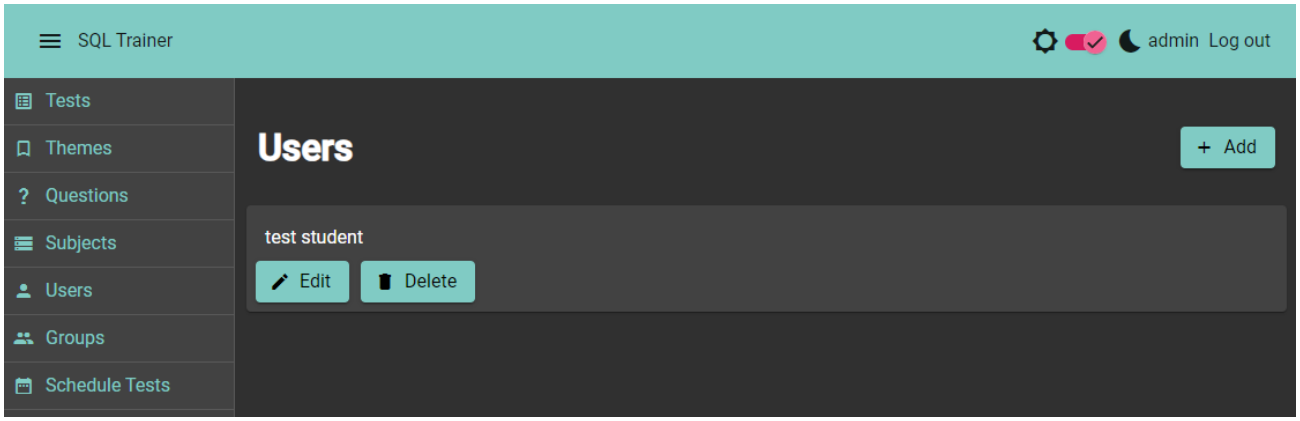


Рисунок 3.7 – Створений користувач з ім’ям «test student» та роллю Студент

3.4.2 Створення тем, завдань та тестів

Для створення тесту за розкладом необхідно створити питання для цього тесту та прив’язати питання до відповідної теми.

Створена тема під назвою «test theme» зображена на рисунку 3.8.

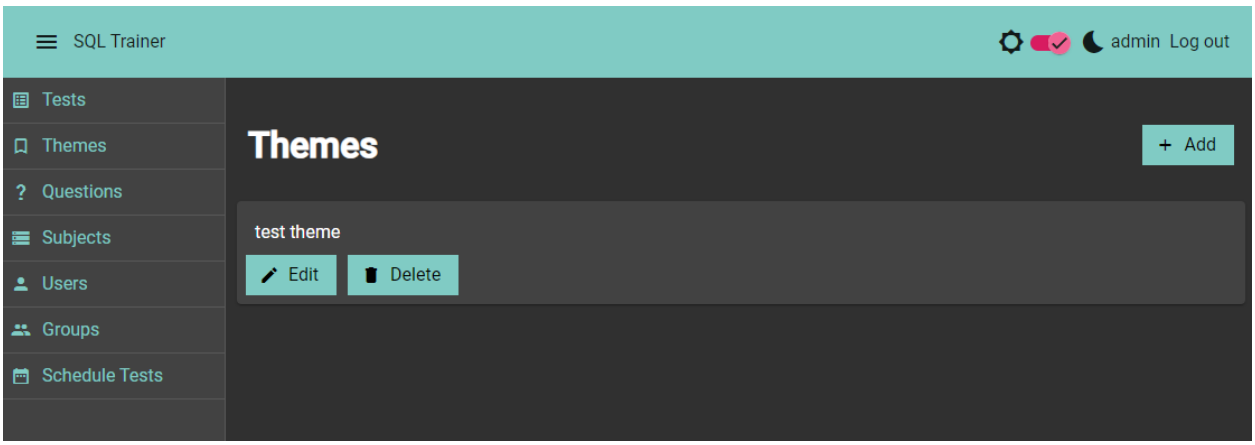


Рисунок 3.8 – Створена тема під назвою «test theme»

Створимо запитання та закріпимо їх за цієї темою. Для створення запитання необхідно створити базу даних, таблиці у цій базі даних та заповнити ці таблиці тестовими даними.

Створена база даних, як буде використована у тесті, з назвою «test_database» зображена на рисунку 3.9.

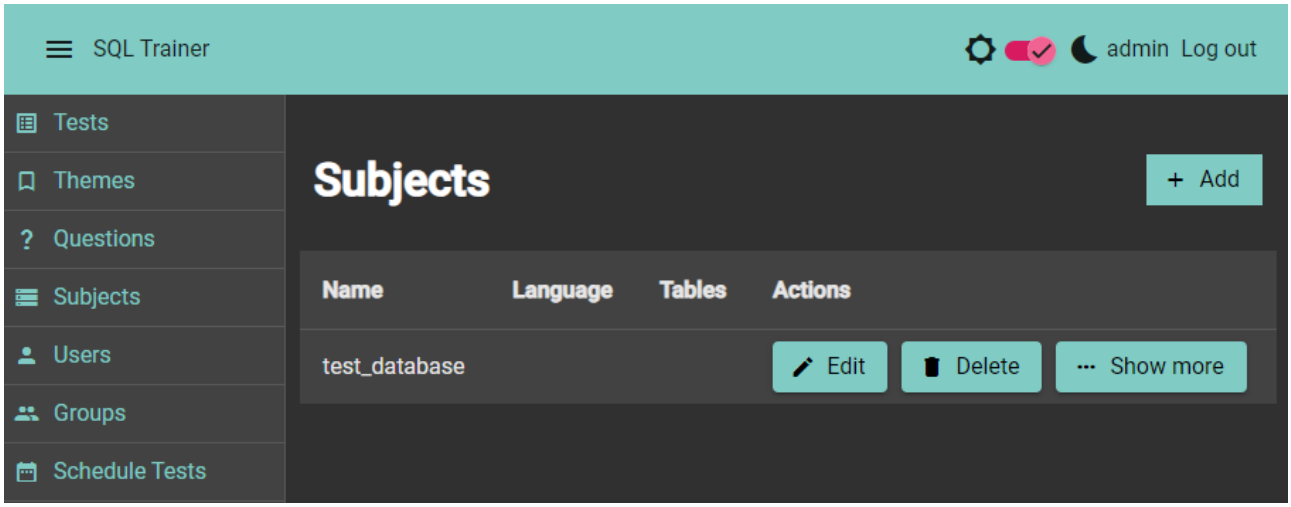


Рисунок 3.9 – Створена база даних з назвою «test_database»

Створені таблиці для цієї бази даних зображені на рисунку 3.10

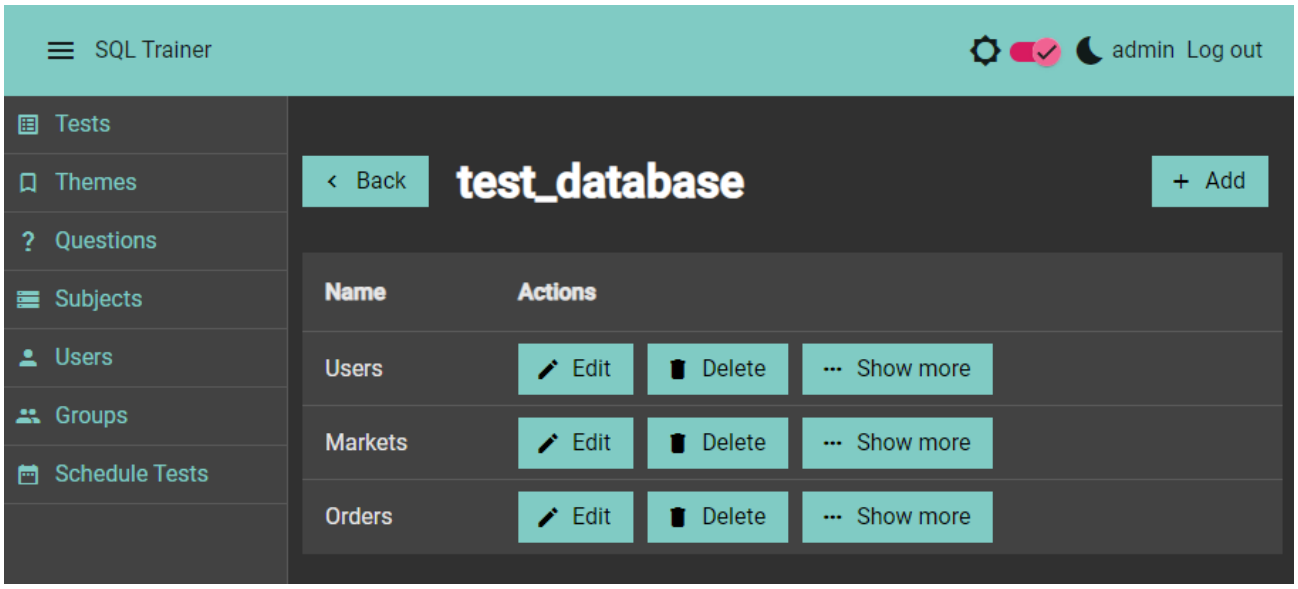


Рисунок 3.10 – Створені таблиці для бази даних

Таблиця «Users» (рис. 3.11) буде зберігати у собі інформацію про користувачів бази даних. Вона складається з полів Id (унікальний ідентифікатор користувача), Name (ім’я користувача, яке було зазначено при оформленні замовлення), LastOrderDate (дата останнього замовлення, зробленого цим користувачем), HasDiscount (булева колонка, яка відповідає за те, чи має цей користувач скидку на замовлення).

Рисунок 3.11 – Наповнення таблиці «Users»

Таблиця «Markets» (рис. 3.12) зберігає у собі інформацію про магазини бази даних. Складається з полів Id (ідентифікатор), Name (назва магазину).

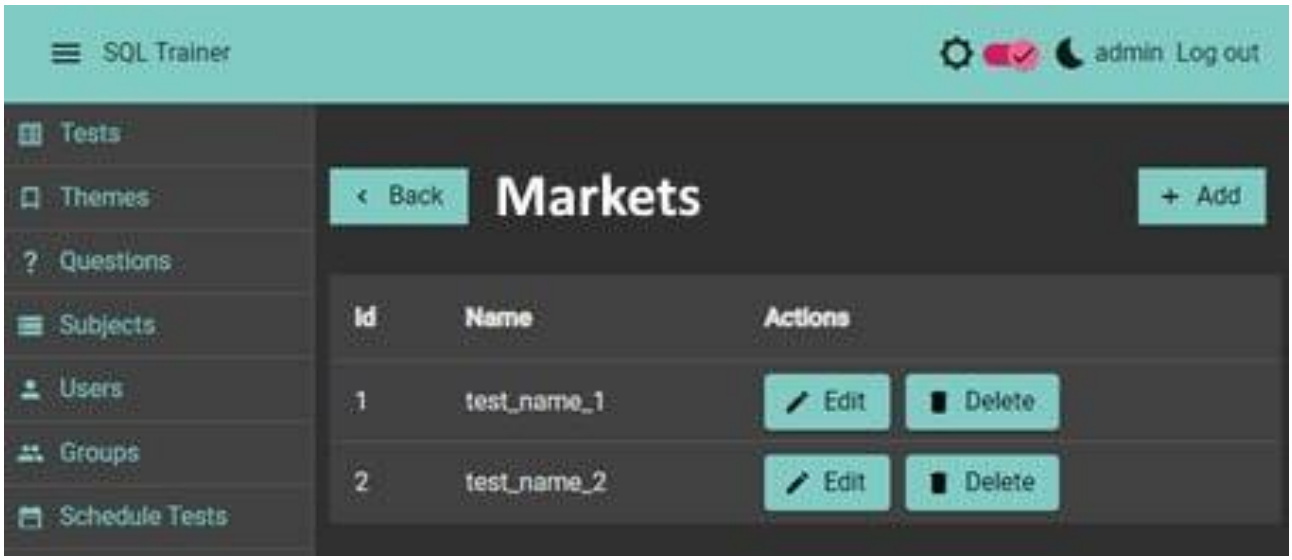


Рисунок 3.12 – Наповнення таблиці «Markets»

Таблиця «Orders» (рис. 3.13) зберігає у собі інформацію про замовлення бази даних. Складається з полів Id (ідентифікатор), UserId (ідентифікатор користувача, який зробив замовлення), OrderDate (дата створення замовлення), TotalPrice (ціна замовлення), PaymentMethod (метод оплати замовлення, котрий було обрано користувачем).

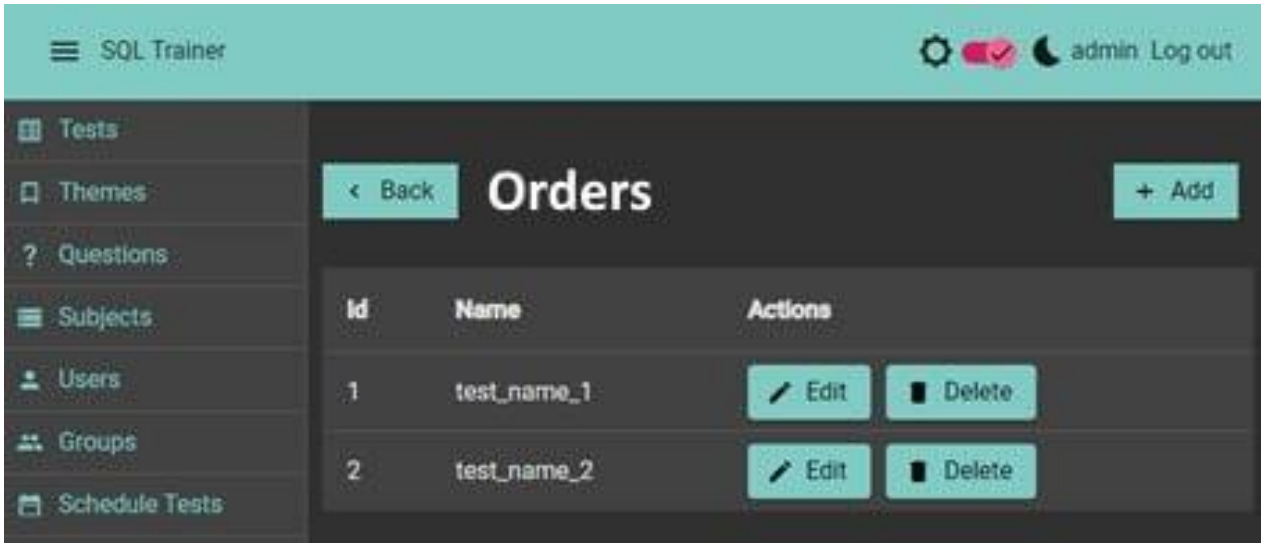


Рисунок 3.13 – Наповнення таблиці «Orders»

Після створення бази даних, таблиць та заповнення таблиць даними, перейдемо до створення запитань, які будуть використані у тесті. Створені запитання зображені на рисунку 3.14.

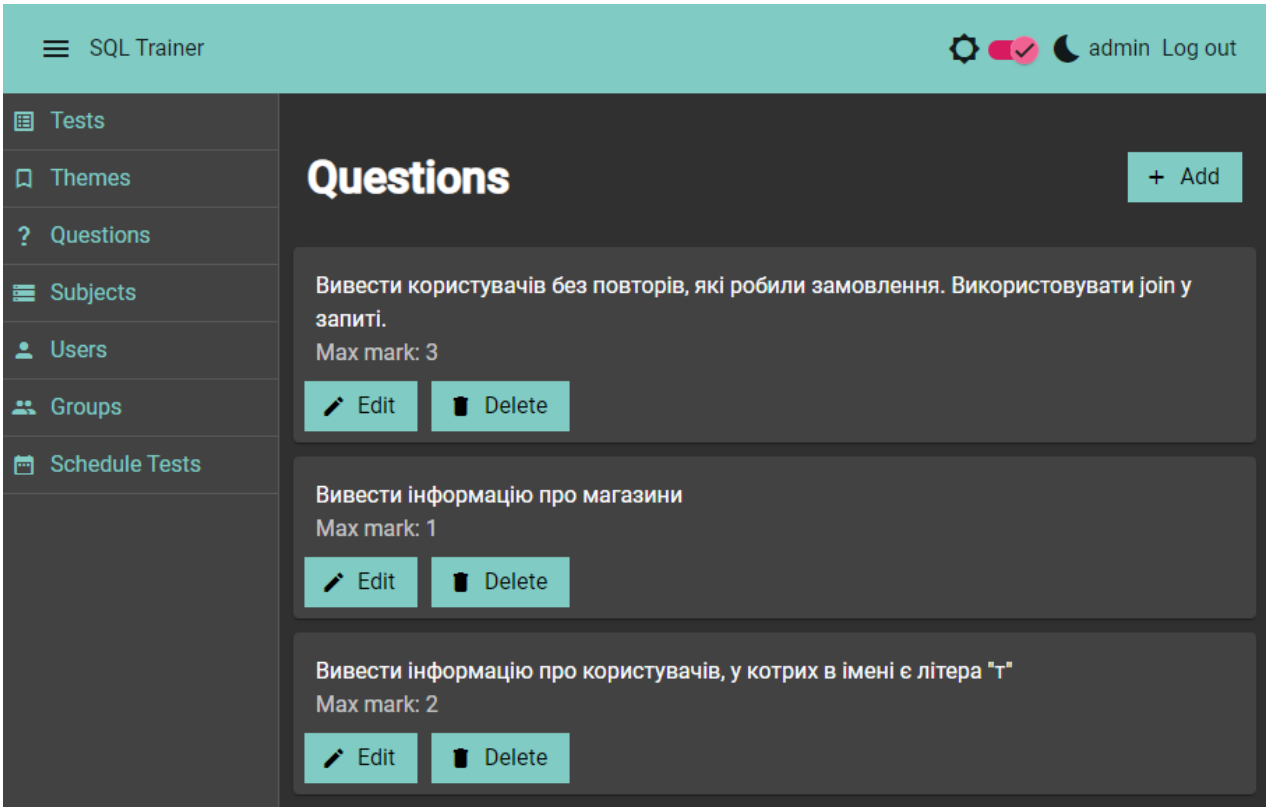


Рисунок 3.14 – Створені запитання

Після створення запитань можна створити тест за розкладом та назначити його початковий час та кінцевий і назначити цей тест групам студентів. Створений тест за розкладом зображено на рисунку 3.15.

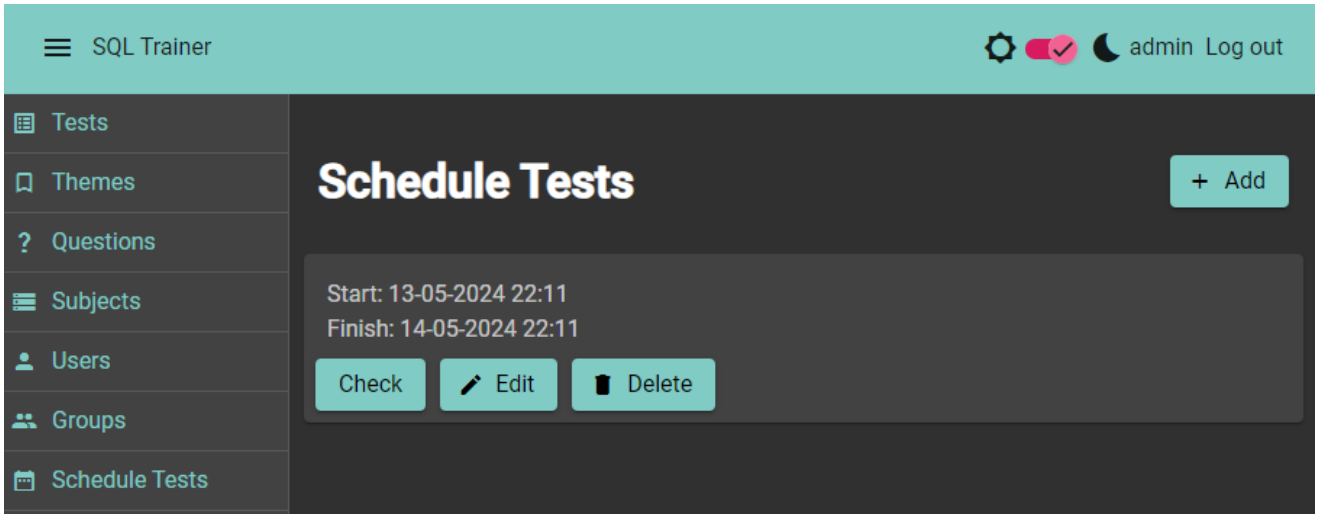


Рисунок 3.15 – Створений тест за розкладом

Тепер студенти, яким назначили цей тест можуть пройти його з 13 травня по 14 травня.

Для проходження цього тесту на найвищу оцінку студенту необхідно правильно відповісти на наступні запитання:

– вивести інформацію про користувачів;

– вивести користувачів, які народилися після 2000 року;

– вивести унікальні назви товарів, що є в таблиці Product;

– вивести імена користувачів, які зробили хоча б одне замовлення (використати JOIN);

– вивести кількість товарів у кожному замовленні;

– вивести середню суму замовлення по кожному користувачу;

– вивести користувачів, які робили замовлення у магазині з ідентифікатором 3;

– вивести магазини, у яких ще не було замовлень (використати NOT IN);

– вивести користувачів, які зробили замовлення на суму більшу за середню (підзапит);

– вивести користувачів, які робили замовлення у всіх магазинах (використати NOT EXISTS);

– вивести трьох користувачів з найбільшою кількістю замовлень;

– вивести рейтинг користувачів за кількістю замовлень, використовуючи віконну функцію RANK().Правильні відповіді на запитання, перелічені вище представленні лістингах з 3.1 по 3.12.

Лістинг 3.1 Правильна відповідь на питання «Вивести інформацію про користувачів»:

```
SELECT *
FROM Users;
```

Лістинг 3.2 Правильна відповідь на питання «Вивести користувачів, які народилися після 2000 року»:

```
SELECT *
FROM Users
WHERE BirthDate > '2000-01-01';
```

Лістинг 3.3 Правильна відповідь на питання «Вивести унікальні назви товарів, що є в таблиці Product»:

```
SELECT DISTINCT ProductName
FROM Product;
```

Лістинг 3.4 Правильна відповідь на питання «Вивести імена користувачів, які зробили хоча б одне замовлення (JOIN)»:

```
SELECT DISTINCT u.Name
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId;
```

Лістинг 3.5 Правильна відповідь на питання «Вивести кількість товарів у кожному замовленні»:

```
SELECT o.Id AS OrderId, COUNT(oi.ProductId) AS ProductCount
FROM Orders o
INNER JOIN OrderItems oi ON o.Id = oi.OrderId
GROUP BY o.Id;
```

Лістинг 3.6 Правильна відповідь на питання «Вивести середню суму замовлення по кожному користувачу»:

```
SELECT u.Name, AVG(o.TotalAmount) AS AvgOrderAmount
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId
GROUP BY u.Name;
```

Лістинг 3.7 Правильна відповідь на питання «Вивести користувачів, які робили замовлення у магазині з ідентифікатором 3»:

```
SELECT DISTINCT u.Name
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId
WHERE o.MarketId = 3;
```

Лістинг 3.8 Правильна відповідь на питання «Вивести магазини, у яких ще не було замовлень (NOT IN)»:

```
SELECT *
FROM Markets
WHERE Id NOT IN (
   SELECT DISTINCT MarketId
   FROM Orders
   WHERE MarketId IS NOT NULL
);
```

Лістинг 3.9 Правильна відповідь на питання «Вивести користувачів, які зробили замовлення на суму більшу за середню (підзапит)»:

```
SELECT DISTINCT u.Name
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId
WHERE o.TotalAmount > (SELECT AVG(TotalAmount) FROM Orders);
```

Лістинг 3.10 Правильна відповідь на питання «Вивести користувачів, які робили замовлення у всіх магазинах (NOT EXISTS)»:

```
SELECT u.Name
FROM Users u
WHERE NOT EXISTS (
   SELECT *
   FROM Markets m
   WHERE m.Id NOT IN (
      SELECT o.MarketId
      FROM Orders o
      WHERE o.UserId = u.Id ));
```

Лістинг 3.11 Правильна відповідь на питання «Вивести трьох користувачів з найбільшою кількістю замовлень»:

```
SELECT TOP 3 u.Name, COUNT(o.Id) AS OrderCount
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId
GROUP BY u.Name
ORDER BY OrderCount DESC;
```

Лістинг 3.12 Правильна відповідь на питання «Вивести рейтинг користувачів за кількістю замовлень (віконна функція RANK())»:

```
SELECT
   u.Name,
   COUNT(o.Id) AS OrderCount,
   RANK() OVER (ORDER BY COUNT(o.Id) DESC) AS UserRank
FROM Users u
INNER JOIN Orders o ON u.Id = o.UserId
GROUP BY u.Name
ORDER BY UserRank;
```

3.4.3 Проходження тестів та перегляд результатів

Під час проходження тесту система періодично кожні 10 секунд проводить верифікацію обличчя студента. Якщо правила верифікації порушено тричі, відповіді студента фіксуються, і тест вважається невдалим, при цьому за кожне питання ставиться оцінка 0.

Під час проходження тесту користувач має навігаційну панель питань, для того щоб переключатись між запитаннями, структуру бази даних, для

розуміння структури бази даних, кнопку «Execute» для того, щоб виконати запит, написаний користувачем, та бути впевненим, що запит відпрацьовує правильно. При правильному відпрацюванні запиту користувач побачить дані, які повернув його запит. Екран проходження тесту зображено на рисунку 3.16.

Після проходження тесту користувач буде в змозі переглянути результати пройденого тесту, коли час для тесту закінчився. Користувач буде в змозі подивитися свою відповідь на запитання та оцінку з коментарями від. Екран перегляду результатів тесту зображено на рисунку 3.17.

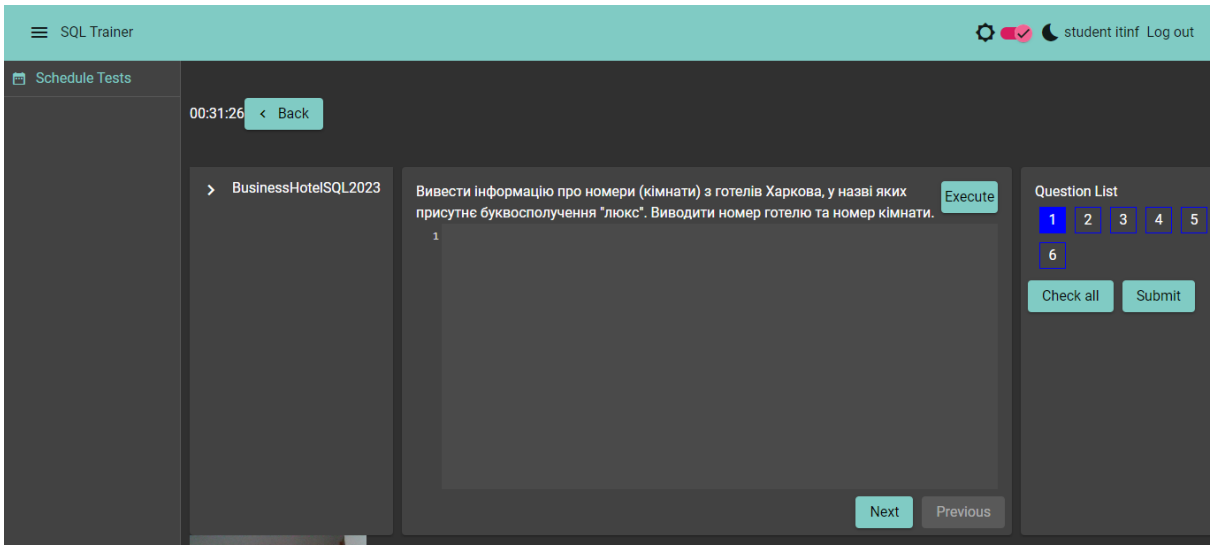


Рисунок 3.16 – Екран проходження тесту

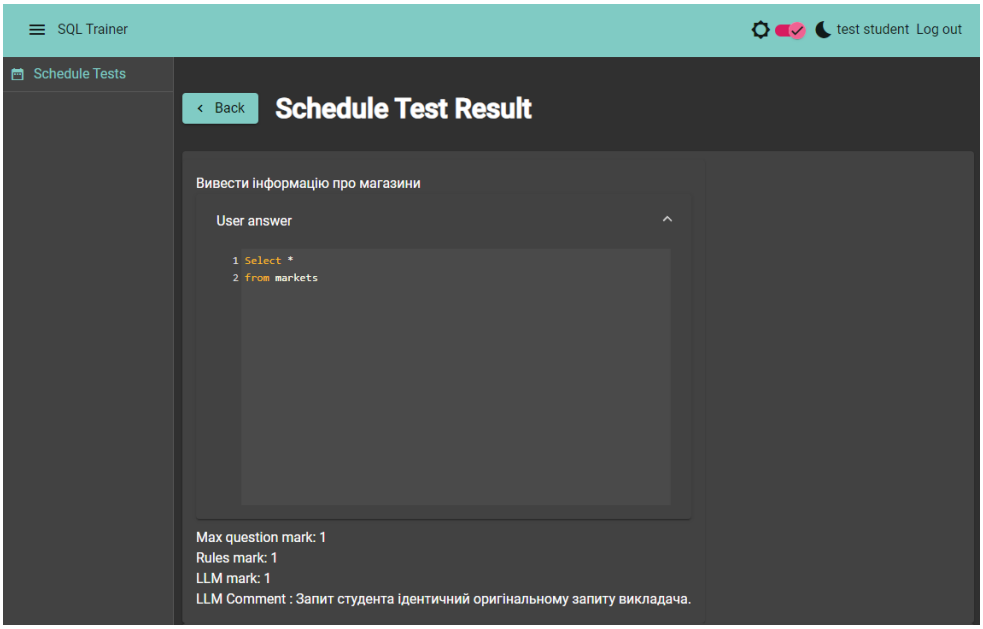


Рисунок 3.17 – Екран перегляду результатів тесту

3.5 Оцінка роботи застоснку

Для перевірки ефективності та практичної придатності розробленої системи автоматичного оцінювання SQL-скриптів було проведено детальне експериментальне тестування в умовах, максимально наближених до реального навчального процесу.

У тестуванні взяли участь 20 студентів, які виконали 10 різних SQL-завдань. Завдання охоплювали широкий спектр складності: від простих вибірок (SELECT, WHERE) до запитів із використанням агрегаційних функцій, підзапитів, JOIN-операторів, віконних функцій та групування.

Після цього оцінки двох моделей та евристичного алгоритму комбінувалися за алгоритмом усереднення результатів, що використовує механізм взаємного контролю між LLM-моделями (якщо розбіжність ≤ 20 %, обчислюється середнє значення; якщо > 20 % – результат позначається як «невпевнений» і підлягає ручній перевірці).

У таблиці 3.1 наведено приклад перших семи завдань із набору тестів, де відображені оцінки, надані моделями GPT-3.5-Turbo та Gemini-2.0-Flash, а також експертна оцінка викладача.

Таблиця 3.1 – Результати оцінювання моделей GPT-3.5 та Gemini у порівнянні з викладачем

№ запитання	**Оцінка GPT**	**Оцінка Gemini**	**Оцінка викладача**
1	0,7	1	1
2	0	0,7	0,7
3	1	0,7	1
4	0,7	0,7	0,7
5	0	0	0
6	1	1	1
7	0,7	1	1

Як видно з таблиці, обидві моделі демонструють близькі результати, проте іноді оцінки відрізняються. Наприклад, у питаннях 1 та 7 модель GPT

занижує бал, тоді як Gemini точніше відтворює логіку викладача. У питанні 3 навпаки – Gemini недооцінила результат, а GPT оцінила правильно.

Для підвищення достовірності результатів було використано алгоритм усереднення оцінок від двох моделей. Середнє значення розраховувалося як арифметичне середнє між оцінками GPT і Gemini (табл 3.2).

Таблиця 3.2 – Результати оцінювання моделей GPT-3.5 та Gemini у порівнянні з викладачем

№ запитання	**Середня оцінка між моделями**	**Оцінка викладача**
1	0,85	1
2	0,35	0,7
3	0,85	1
4	0,7	0,7
5	0	0
6	1	1
7	0,85	1

Як видно, середнє значення дозволяє згладити індивідуальні похибки кожної моделі. Наприклад, для запитань 1, 3 та 7 похибка зменшилася вдвічі порівняно з окремими оцінками GPT. Точність системи підвищується завдяки компенсації помилок різних моделей – там, де одна недооцінює результат, інша його коригує.

Для кількісної оцінки якості роботи системи було обчислено дві базові метрики (табл 3.3).

Таблиця 3.3 – Метрики точності окремих моделей

Модель	**MAE**	**RMSE**	**Відсоток відхилень > 20**
GPT-3.5-Turbo	0,18	0,32	22 %
Gemini-2.0-Flash	0,14	0,27	19%

Як видно з таблиці, Gemini-2.0-Flash продемонструвала трохи кращу стабільність і меншу похибку при оцінюванні результатів студентів.

Однак обидві моделі показують рівень похибки в межах 15–20 %, що є прийнятним для автоматизованого тестування, але потребує уточнення механізмів перевірки.

Для подальшого підвищення надійності система була доповнена евристичним алгоритмом, який виконує формальну перевірку структури SQL-скрипта та набору даних, який повернув скрипт. В нашому випадку евристичний алгоритм не замінює LLMs, він використовується у парі з моделями. Такий підхід дозволяє знизити кількість «викидень» так як обидва алгоритми «підстраховують» один одного у випадках, коли моделі можуть пропустити помилку в скрипті, так як не можуть перевірити остаточний результат, котрий скрипт повернув, і чи є цей скрипт синтаксично правильним відносно структури бази даних та синтаксису, або коли евристичний алгоритм вважає відповідь студента коректною, так як скрипт студента повертає такий самий набір даних, як і еталонний скрипт, проте по факту завдання не є коректно виконаним, на допомогу приходять моделі, котрі можуть повноцінно проаналізувати скрипт.

Приклад випадку коли евристичний алгоритм допомагає не пропустити помилку у скрипті описано нижче.

Завдання: «Вивести з Booking заселення в 3 перші дати (виводити всі поля з Booking). Дата заселення знаходиться у полі dateB. В одну дату може бути багато заселень.». Правильна відповідь та відповідь студента описані у лістингах 3.13 та 3.14.

Лістинг 3.13 Правильна відповідь на питання:

```
SELECT *
FROM booking
WHERE dateb IN (SELECT DISTINCT TOP 3 dateb
FROM booking
ORDER BY dateb);
```

Лістинг 3.14 Відповідь студента

```
SELECT *
FROM booking
WHERE dateb IN (SELECT TOP 3 dateb
FROM booking
ORDER BY dateb);
```

Обидва скрипти по факту є коректиними, але є випадок коли дата може бути однакова і через відсутність оператора «*DISTINCT*» запит студента вертає неправильний набір даних. Такий випадок не можуть побачити моделі, проте може побачити евристичний алгоритм.

Приклад коли LLMs бачать, що завдання не є коректно виконаним, незважаючи на те, що скрипт повернув правильний набір даних описано нижче.

Завдання: «Вивести готелі, які відвідали в 2022 р. (dateB дорівнює 2022) всі клієнти нашої мережі (всі клієнти з відношення Guest). Виводити номер готелю та назву. Відсортувати за назвою за зростанням.». Правильна відповідь та відповідь студента описані у лістингах 3.15 та 3.16.

Лістинг 3.13 Правильна відповідь на питання:

```
SELECT Booking.hotelNo, hotelName
FROM Booking inner join Hotel on Booking.hotelNo=Hotel.hotelNo
WHERE Year(dateB) = 2022
GROUP BY Booking.hotelNo, hotelName
HAVING COUNT(DISTINCT Booking.guestNo) = (SELECT COUNT(*) FROM Guest)
ORDER BY hotelName;
```

Лістинг 3.14 Відповідь студента

```
select hotelNo, hotelName
from Hotel
where hotelNo=6
```

Далі було проведено розрахунок метрик для комбінованого алгоритму усереднених оцінок (табл 3.4).

Таблиця 3.4 – Метрики точності алгоритмів

Алгоритм	**MAE**	**RMSE**	**Відсоток відхилень > 20**
GPT + Gemini	0,11	0,23	14%
Евристичний алгоритм	0,17	0,29	21%
Дворівневий алгоритм	0,09	0,15	12%

Отримані результати демонструють, що усереднення оцінок двох моделей дозволяє знизити похибку приблизно на 30 % у порівнянні з найкращою окремою моделлю (Gemini-2.0-Flash). Кількість випадків, коли оцінка системи відрізнялася від викладацької більш ніж на 20 %, зменшилася до 14 %.

За результатами тестування можна зробити такі висновки:

– дворівневий алгоритм оцінювання дозволив автоматизувати близько 90% процесу перевірки SQL-завдань. Тобто у більшості випадків викладачеві не потрібно було вручну перевіряти відповіді студентів, адже середнє значення оцінки збігалося або відрізнялося незначно (до 0.1 балу);

– використання двох незалежних моделей різних виробників забезпечило ефект взаємної перевірки: GPT-3.5 частіше робила помилки у структурних елементах SQL-запиту, тоді як Gemini точніше оцінювала логіку виконання завдання. Поєднання їхніх результатів зменшило вплив випадкових похибок;

Незважаючи на загальне покращення, у 14 % випадків система все ще демонструвала відхилення понад 20 % від оцінки викладача.

Такі помилки переважно виникали:

– при запитах із вкладеними підзапитами та складною логікою умов (HAVING, EXISTS, CASE);

– при неправильному трактуванні незначних помилок у синтаксисі;

– у випадках, коли моделі неоднаково розуміли текстове формулювання завдання.

Для підвищення точності роботи алгоритму доцільно:

– доопрацювати механізм вагового оцінювання критеріїв – різним аспектам запиту (синтаксис, логіка, результат) можна надати різні коефіцієнти ваги;

– проводити контекстне порівняння результатів виконання запиту, а не лише аналіз тексту SQL-коду;

– додати механізм самоадаптації – алгоритм може навчатися на помилках, порівнюючи власні оцінки з остаточними оцінками викладача;

– розширити датасет, включивши приклади складних SQL-структур (CTE, UNION, віконні функції), щоб моделі краще розуміли логіку запитів.

Отже, результати експериментального тестування підтвердили, що розроблений застосунок є ефективним інструментом автоматизованого оцінювання SQL-скриптів. Алгоритм, який поєднує результати двох моделей та оцінку евристичного алгоритму забезпечує високий рівень точності (MAE ≈ 0,09, RMSE ≈ 0,15), що дозволяє зменшити навантаження на викладача приблизно на 80%.

Водночас система ще потребує удосконалення механізмів аналізу складних запитів і динамічного налаштування ваг оцінювання, що дозволить у перспективі досягти точності, максимально наближеної до людської оцінки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено та досліджено вебзастосунок для автоматичного оцінювання SQL-скриптів студентів із використанням великих мовних моделей (LLM).

Розроблена система поєднує можливості штучного інтелекту, евристичних методів аналізу та класичних алгоритмів порівняння для забезпечення об’єктивної, пояснюваної та автоматизованої перевірки SQL-запитів.

Проведене дослідження охоплювало як аналітичну, так і практичну частини: було створено методологію перевірки, розроблено прототип системи, проведено серію експериментів із використанням моделей GPT-3.5-Turbo та Gemini-2.0-Flash, а також здійснено порівняльний аналіз результатів автоматичного оцінювання з оцінками викладача.

У процесі роботи були виконані такі завдання:

– проведено аналіз існуючих підходів та додатків для навчання та перевірки SQL-запитів, визначено їх сильні та слабкі сторони, включаючи системи з використанням ШІ (наприклад, DataLemur, SQL Judge, AI Tutor) та традиційні платформи без інтелектуальної перевірки;

– досліджено можливості сучасних великих мовних моделей (GPT, Gemini, Claude, Mistral) для задач аналізу коду та автоматичного тестування, оцінено їхню ефективність при роботі з SQL-запитами;

– розроблено методологію представлення SQL-скриптів і еталонних рішень, яка передбачає побудову тестових наборів, сценаріїв виконання запитів та чітко визначені критерії оцінювання;

– створено метод автоматичної інтерпретації та оцінювання SQL-запитів із використанням LLM, що включає:

– перевірку коректності результату виконання запиту;

– аналіз структури та синтаксису SQL-коду;

– виявлення неефективних конструкцій і логічних помилок;

– формування пояснення та рекомендацій для студента природною мовою;

– реалізовано прототип системи автоматичного оцінювання SQL-скриптів, інтегрований у навчальну платформу з можливістю використання API для взаємодії з мовними моделями;

– проведено експериментальне дослідження роботи алгоритмів, під час якого виконано оцінювання понад 270 студентських рішень; результати порівнювалися з викладацькими оцінками за метриками MAE та RMSE, що дозволило об’єктивно оцінити точність;

– сформульовано рекомендації щодо подальшого використання великих мовних моделей у навчальних системах, включаючи аспекти достовірності, узгодженості оцінок та прозорості прийняття рішень.

Розроблений застосунок може бути використаний у системах електронного тестування знань (зокрема у ВНЗ, коледжах та онлайн-платформах), навчальних курсах з баз даних та програмування, середовищах підготовки та сертифікації спеціалістів у сфері Data Science та SQL-аналітики, автоматизованих тренажерах для самостійного навчання студентів.

Алгоритм оцінювання продемонстрував високу ефективність і точність: середня абсолютна похибка (MAE) при використанні комбінованого підходу GPT + Gemini склала 0.11, а середньоквадратична похибка (RMSE) – 0.23. Це дозволило автоматизувати приблизно 80–85 % процесу перевірки SQL-запитів, що значно зменшує навантаження на викладача і підвищує об’єктивність оцінювання.

Подальший розвиток роботи передбачає:

– розширення системи за рахунок алгоритму гнучкого оцінювання з ваговими коефіцієнтами для різних критеріїв;

– інтеграцію додаткових моделей (наприклад, GPT-4 або Gemini 2.0 Pro) для підвищення якості аналізу;

– впровадження підсистеми пост-обробки результатів, що виявлятиме сумнівні оцінки та подаватиме їх на ручну перевірку;

– створення адаптивного навчального модуля, який би формував рекомендації студенту залежно від типових помилок.

Отже, у результаті виконаної роботи було створено, реалізовано та експериментально перевірено метод автоматичного оцінювання SQL-скриптів з використанням великих мовних моделей.

Розроблений підхід продемонстрував здатність ефективно інтерпретувати та оцінювати студентські запити, формувати коментарі українською мовою та забезпечувати узгодженість результатів із людськими оцінками.

Система має високий потенціал для практичного застосування у закладах освіти та онлайн-платформах, де потрібна масштабована, об’єктивна і зрозуміла автоматична перевірка завдань з SQL. Подальше вдосконалення алгоритмів та додавання підтримки гнучкого оцінювання дозволить досягти ще більшої точності та адаптивності, наблизивши роботу системи до рівня експертної перевірки викладача.

Результати роботи апробовано у вигляді 2 тез доповідей під час XXIX Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» [41] та 11-ої Міжнародної науково-практичної конференції «World science: problems, issues and prospects for development» [47].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shelest V., Yakovleva O. (2024, November 5-8) Research on selecting web application architecture based on the analysis of applied requirements. Proceedings of the X International Scientific and Practical Conference «Computerintegrated technologies of automation of technological processes». Hamburg, Germany, pp. 46-54.

2. DB Fiddle – SQL Database Playground. URL: https://www.db-fiddle.com/ (дата звернення 08.11.2025).

3. SQL Fiddle – Online SQL Compiler for learning & practice. URL: https://sqlfiddle.com/ (дата звернення 08.11.2025).

4. SQLite Online. URL: https://sqliteonline.com/ (дата звернення 08.11.2025).

5. SQLZoo. URL: https://sqlzoo.net/ (дата звернення 08.11.2025).

6. SQLBolt – Learn SQL. URL: https://sqlbolt.com/ (дата звернення 08.11.2025).

7. DataCamp – Data and AI. URL: https://www.datacamp.com/ (дата звернення 08.11.2025).

8. Codecademy.com. URL: https://www.codecademy.com/ (дата звернення 08.11.2025).

9. LeetCode Database. URL: https://leetcode.com/problemset/database/ (дата звернення 08.11.2025).

10. Beaver Community | Free Open-Source Database. URL: https://dbeaver.io/ (дата звернення 08.11.2025).

11. pgAdmin - PostgreSQL Tools. URL: https://www.pgadmin.org/ (дата звернення 08.11.2025).

12. Docker: Accelerated Container Application Development. URL: https://www.docker.com/ (дата звернення 08.11.2025).

13. Visual Studio Code - The open source AI code editor. URL: https://code.visualstudio.com/ (дата звернення 08.11.2025).

14. SoloLearn. URL: https://www.sololearn.com/ (дата звернення 08.11.2025).

15. Mimo: Learn to Code in Python, JavaScript, HTML, CSS, & more. URL: https://mimo.org/ (дата звернення 08.11.2025).

16. Grasshopper: Learn to Code. URL: https://grasshopper-yz.netlify.app/ (дата звернення 08.11.2025).

17. Enki | Learn 10x faster: coding, no-code, data. URL: https://www.enki.com/ (дата звернення 08.11.2025).

18. Programming Hub: Learn Programming, Coding Online. URL: https://programminghub.io/ (дата звернення 08.11.2025).

19. Dodona: Home. URL: https://dodona.be/ (дата звернення 08.11.2025).

20. LightBot. URL: https://lightbot.com/ (дата звернення 08.11.2025).

21. ScratchJr – Home. URL: https://www.scratchjr.org/ (дата звернення 08.11.2025).

22. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., & Hudáková M. (2025) Image description compression in classification structural methods, IEEE Access, vol. 13, pp. 43631-43641.

23. Yakovleva, O., & Nikolaieva, K. (2020). Research Of Descriptor Based Image Normalization And Comparative Analysis Of SURF, SIFT, BRISK, ORB, KAZE, AKAZE Descriptors. Advanced Information Systems, 4(4), 89-101.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no. 1, pp. 113-125.

25. Gorokhovatskyi , O., & Yakovleva , O. (2024). Medoids as a packing of ORB image descriptors. Advanced information systems, 8(2), pp. 5–11.

26. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385.

27. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. CEUR Workshop Proceedings Vol. 3403. pp. 69-86.

28. Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Public Administration and Regional Development, 19(1), pp. 161–186.

29. Yakovleva O., Matúšová S., Tvoroshenko I., Isaiev Y. (2024). Visitor counting based on video stream analysis from surveillance cameras. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing», vol. 20, no. 1, pp. 67–87.

30. Lyashenko, V., Kobylin, O., & Selevko, O. (2020). Wavelet analysis and contrast modification in the study of cell structures images.

31. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. IEEE Access, 9, 92964-92973.

32. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, International Journal of Academic and Applied Research, 9(10), pp. 249-263.

33. Cherednichenko, Olga, Ivashchenko, Oksana, Cibák, Ľuboš and Lincenyi, Marcel. "Item Matching Model in E-Commerce: How Users Benefit" Economics and Culture, vol.20, no.1, 2023, pp.77-90.

34. Cherednichenko, O., Ivashchenko, O., Lincényi, M., Kováč, M. 2023. Information technology for intellectual analysis of item descriptions in e-commerce, Entrepreneurship and Sustainability Issues 11(1): 178-190.

35. Yakovleva, O., Matúšová, S., Liubchenko, V., Maksimov, H. (2025). Innovative solutions based on AI in education, on the example of generating

multimodal lecture notes. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing», vol. 21, no. 1, pp.140–163.

36. Yanholenko, O., Grinchenko, M., Rohovyi, M., Yakovleva, O., Rogovyi, A. (2025). The model and method of intelligent planning of IT project team work. PhD Workshop on Artificial Intelligence in Computer Science at 9th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2025). CEUR Workshop Proceedings Vol. 3403. pp. 134-149.

37. Kuzomin, O., & Lyashenko, V. (2022). Agent-Based Model as a Research Tool.

38. Yakovleva, O., Matúšová, S., & Talakh, V. (2025, February 14). Gradio and Hugging capabilities for developing research AI applications. Proceedings of the VII International Scientific and Practical Conference «Scientific practice: modern and classical research methods», Boston, USA, pp. 202-205.

39. Yakovleva, O., Matúšová, S., & Koshel, V. (2025, February 21). Implementation of AI approaches in current tools for managing image collections to improve the search capabilities. Proceedings of the IV Correspondence International Scientific and Practical Conference «Science in motion: classic and modern tools and methods in scientific investigations» in Periodical International scientific journal «Grail of science». Vinnytsia, Ukraine - Vienna, Austria. Vol. 49. pp. 752–755.

40. Науменко В.В., Яковлева О.В. (2024). Розробка методу автоматичного оцінювання відповідей студентів з використанням GPT-моделей від OpenAI для веб-застосунку з тестування знань мови SQL. 28-ий міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті».

41. Науменко В.В. (2025). Аналіз можливостей LLMs щодо допомоги у вивченні мови SQL. Радіоелектроніка і молодь у XXI столітті: Тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.). Харків: ХНУРЕ. Т. 7, с. 107–109.

42. GPT API Pricing. URL: https://openai.com/api/pricing/ (дата звернення 08.11.2025).

43. Gemini Developer API Pricing. URL: https://ai.google.dev/gemini-api/docs/pricing (дата звернення 09.11.2025).

44. Gemini 2.0 flash vs GPT-3.5 Turbo - Detailed Performance & Feature Comparison. DocsBot AI. (2025). URL: https://docsbot.ai/models/compare/gemini-2-0-flash/gpt-3-5-turbo (дата звернення 07.11.2025).

45. Yakovleva, O., Matúšová, S., Táncošová, J. (2024, December 16-18). Investigation of LLMs for generating answers based on user-provided content to support educational and organizational processes. Abstracts of XVI International Scientific and Practical Conference «Modern and new technical trends that help humanity». Thessaloniki, Greece, Pp. 289-295.

46. Yakovleva, O., Nebeský, L., Kirichenko, A. (2023). Using the GPT models for responses based on custom content to develop neural consultant for university applicants. Abstracts of V International Scientific and Practical Conference «The world of modern technologies and inventions» Madrid, Spain. Pp. 172-178.

47. Yakovleva, O., Matúšová, S., Naumenko, V. Developing prompts for automated evaluation of SQL scripts with Large Language Models for educational use. Proceedings of the XI International Scientific and Practical Conference. Sofia, Bulgaria. 2025. Pp. 29-38.

48. Naumenko V., Shelest V., & Yakovleva O. (2024). Combination of .Net technology and Angular framework to develop application for testing SQL language knowledge. Proceedings of the XVth International Scientific and Practical Conference «Free And Open Source Software», Ukraine, Kharkiv, February 13-14, 2024. pp.63-66.

49. Cherednichenko, O., Vovk, M., Yanholenko, O., & Yakovleva, O. (2020). Towards the Technology of Employers' Requirements Collection Development. In Integrated Computer Technologies in Mechanical Engineering (pp. 228-239). Springer, Cham.

50. Дацок, Є., Дацок, О., & Руденко, Д. (2025). Аналіз ефективності використання ORM DAPPER та принципів SOLID у проєктуванні

інформаційної системи медичного призначення. Вісник Національного технічного університету" ХПІ". Серія: Інформатика і моделювання, 1(2 (14)), 157-171.

51. Bohdan, N., Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin, O. (2025). Development of a hybrid method to enhance context memory for a chatbot application based on large language models.

52. Танянський, О., & Руденко, Д. (2018). Порівняльний аналіз популярних JavaScript-фреймворків та бібліотек для front-end розробки.

53. Skeet, J. (2019). C# in Depth. Simon and Schuster.

54. Маренич, В. В., & Руденко, Д. (2025). Дослідження методів розробки комп’ютерних застосунків з інтегрованою API.