МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління

ГРУШИЦЬКИЙ Максим Васильович

Програмний модуль для генерування SQL-запитів на основі природної мови з використанням штучного інтелекту / Software Module for Intelligent SQL Queries Generation Based on Natural Language

Спеціальність 122 – Комп’ютерні науки
Освітньо-професійна програма – Штучний інтелект

Кваліфікаційна робота

Виконав студент групи КНШІ-41
М.В. Грушицький

Науковий керівник:
к.т.н., доцент І.В. Турченко

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГРУШИЦЬКИЙ Максим Васильович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль для генерування SQL-запитів на основі природної мови з використанням штучного інтелекту / Software Module for Intelligent SQL Queries Generation Based on Natural Language

керівник роботи І. В. Турченко, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 20 грудня 2024 р. № 938.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:

- провести аналіз предметної області;
- проаналізувати відомі рішення і зробити постановку задачі проєктування;
- представити архітектуру, алгоритмічне та інформаційне забезпечення програмного модуля;
- розробити програмний модуль;
- протестувати роботу модуля.

5. Перелік графічного матеріалу в роботі:

- структура компонентів програмного модуля;
- схема алгоритму роботи програмного модуля;
- схема алгоритму обробки голосового введення SQL-запитів.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент _____ М. В. Грушицький
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І. В. Турченко
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 65 с., 10 рис., 4 табл., 5 додатків, 38 джерел.

Об’єктом дослідження є процес взаємодії користувачів з реляційними базами даних через природномовний інтерфейс.

Метою роботи є розробка програмного модуля для автоматичної генерації SQL-запитів на основі природної мови з використанням технологій штучного інтелекту.

Методи дослідження: методи обробки природної мови (NLP), трансформерні архітектури (GPT-4o), технології розпізнавання мовлення, об’єктно-орієнтоване програмування, використання мов Python та бібліотек PyQt6, g4f, speech_recognition, openpyxl, python-docx, ReportLab.

Результатом роботи є програмний модуль SQL Viewer, що забезпечує генерацію SQL-запитів із природної мови, підтримує авторизацію користувачів, збереження історії запитів, експорт результатів у PDF/Word/Excel, працює з локальними SQLite-базами даних, повністю підтримує українську мову та забезпечує конфіденційність даних.

Розроблене рішення має практичну значущість завдяки тому, що робить SQL більш доступним для користувачів із різним рівнем технічних навичок та оптимізує взаємодію з базами даних для україномовної аудиторії. Це набуває особливої важливості в українських навчальних закладах та бізнес-структурах.

Ключові слова: SQL, ПРИРОДНА МОВА, ШТУЧНИЙ ІНТЕЛЕКТ, ГОЛОСОВИЙ ІНТЕРФЕЙС, PYTHON, GPT-4o, SQLITE.

ANNOTATION

Explanatory note to the qualification work: 65 pages, 10 figures, 4 tables, 5 appendices, 38 references.

The object of research is the process of user interaction with relational databases through a natural language interface.

The aim of the work is to develop a software module for automatic SQL query generation based on natural language using artificial intelligence technologies.

Research methods: natural language processing (NLP), transformer architectures (GPT-4o), speech recognition technologies, object-oriented programming, Python programming language and libraries such as PyQt6, g4f, speech_recognition, openpyxl, python-docx, ReportLab.

The result of the work is the SQL Viewer software module, which provides natural language SQL query generation, user authentication, query history saving, export of results to PDF/Word/Excel, operation with local SQLite databases, full support for the Ukrainian language, and data confidentiality.

The developed solution has practical significance by making SQL more accessible to users with different levels of technical skills and optimizing database interaction for the Ukrainian-speaking audience. This is especially important for Ukrainian educational institutions and business structures.

Keywords: SQL, NATURAL LANGUAGE, ARTIFICIAL INTELLIGENCE, VOICE INTERFACE, PYTHON, GPT-4o, SQLITE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ІС – інформаційна система

ПЗ – програмне забезпечення

API – Application Programming Interface (прикладний програмний інтерфейс)

СУБД – система управління базами даних

ШІ – штучний інтелект

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність)

BERT – Bidirectional Encoder Representations from Transformers

CSV – Comma-Separated Values (значення, розділені комами)

DDL – Data Definition Language (мова визначення даних)

DML – Data Manipulation Language (мова маніпулювання даними)

GPT – Generative Pre-trained Transformer (генеративний попередньо навчений трансформер)

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

LLM – Large Language Model (велика мовна модель)

NLP – Natural Language Processing (обробка природної мови)

PDF – Portable Document Format (формат переносних документів)

SQL – Structured Query Language (мова структурованих запитів)

TTS – Text-to-Speech (перетворення тексту на мовлення)

XML – eXtensible Markup Language (розширювана мова розмітки)

ВСТУП

В епоху цифровізації бази даних стали невід'ємною складовою інформаційних систем, проте доступ до них здебільшого обмежений для фахівців без знання SQL. За даними дослідження Bird S. et al., понад 60% аналітиків та бізнес-користувачів визнають відсутність знань SQL основною перешкодою для ефективної роботи з даними. Розвиток технологій обробки природної мови (NLP) та великих мовних моделей відкриває можливості подолання цього розриву. Дослідження Casanova M. A. et al. підтверджує, що використання ШІ для генерації SQL-запитів підвищує продуктивність роботи з базами даних на 30-45% навіть для досвідчених користувачів, а для початківців – скорочує час отримання інформації у 3-4 рази.

Мета роботи полягає у розробці програмного модуля для генерування SQL-запитів на основі природної мови з використанням технологій ШІ. Для досягнення мети необхідно: проаналізувати сучасні підходи до перетворення природної мови в SQL-запити; визначити вимоги до програмного модуля; спроєктувати архітектуру системи; розробити алгоритми генерації запитів та реалізувати додаткові функціональні можливості (голосове введення, розмежування прав доступу, збереження історії запитів).

Об'єктом дослідження є процес взаємодії користувачів з реляційними базами даних через природномовний інтерфейс, а предметом – методи та алгоритми автоматичної генерації SQL-запитів з використанням технологій ШІ.

Метою є розробка програмного модуля на основі штучного інтелекту для автоматизації процесу перетворення запитів природною мовою в SQL-код. Для досягнення мети необхідно виконати наступні завдання:

- провести аналіз предметної області;
- проаналізувати відомі рішення і зробити постановку задачі проєктування;
- представити архітектуру, алгоритмічне та інформаційне забезпечення програмного модуля;
- розробити програмний модуль;
- протестувати роботу модуля.

Практичне значення результатів виявляється у значному спрощенні доступу до даних для нетехнічних фахівців, інтеграції голосового введення та забезпеченні безпечної роботи з даними. Розроблений програмний модуль характеризується модульною архітектурою, що забезпечує можливість його інтеграції з різними СУБД та адаптації до специфічних потреб підприємств.

Результати дослідження опубліковано в матеріалах науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR – 2025), м. Тернопіль, 27–29 травня 2025 р. (додаток Д).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис предметної області

Предметна область дослідження охоплює взаємодію людини з реляційними базами даних за допомогою природної мови з використанням технологій штучного інтелекту. За даними DB-Engines Ranking, реляційні СУБД займають понад 75% ринку систем управління базами даних. Основним засобом взаємодії з такими базами даних є мова SQL, яка, незважаючи на свою потужність, має високий поріг входження для нетехнічних користувачів.

Обробка природної мови (NLP) є підгалуззю ШІ, що займається взаємодією між комп'ютерами та людською мовою [1].

У контексті перетворення тексту в SQL-запити особливо важливими є синтаксичний та семантичний рівні аналізу. Задача Text-to-SQL є однією з найбільш активних областей досліджень, яка набула особливої актуальності з появою великих мовних моделей (LLM) [7].

Стрімкий розвиток технологій обробки природної мови за останні роки зумовлений насамперед появою та вдосконаленням архітектури трансформерів. Традиційні підходи до NLP мали обмеження у здатності враховувати контекст на значних відстанях у тексті [13]. Це особливо критично для задачі перетворення тексту в SQL, де запит природною мовою може мати складну структуру з різними умовами та відношеннями.

Великі мовні моделі, такі як GPT-4 та BERT, базуються на архітектурі трансформерів, яка використовує механізм уваги для моделювання залежностей між словами [4, 22]. Попереднє навчання таких моделей дозволяє їм набути загального розуміння мови, яке потім адаптується для конкретних завдань.

У контексті Text-to-SQL задача розбивається на кілька підзадач:

- розуміння природномовного запиту користувача;
- аналіз схеми бази даних та визначення релевантних таблиць і полів;
- зіставлення елементів запиту з елементами схеми БД;
- формування коректного SQL-запиту;

- оптимізація та валідація згенерованого запиту.

Сучасні системи генерації SQL умовно поділяються на три категорії [12]:

- спеціалізовані моделі, навчені на специфічних наборах даних Text-to-SQL;
- адаптовані загальні моделі з додатковим навчанням на даних Text-to-SQL;
- нульове навчання з LLM, коли великі мовні моделі використовуються без спеціального навчання.

Незважаючи на значні успіхи, задача Text-to-SQL ще далека від повного вирішення. Серед основних викликів: генерація складних запитів, обробка неоднозначностей, адаптація до різних схем баз даних та забезпечення відповідності запитів специфіці конкретних СУБД [3, 15].

Практичне застосування технологій перетворення тексту в SQL охоплює бізнес-аналітику, управління даними, системи підтримки прийняття рішень та освітні платформи для вивчення SQL.

Взаємодія з базами даних через SQL створює значні виклики для багатьох користувачів. Дослідження показують, що аналітики витрачають до 70% часу на формулювання та налагодження SQL-запитів замість безпосереднього аналізу даних [15].

Основними перешкодами для опанування SQL є:

- складний синтаксис та формальна граматика, що вимагає точного дотримання правил;
- концептуальна складність реляційної моделі, яка вимагає абстрактного мислення;
- високі вимоги до знання структури даних (таблиць, стовпців, зв'язків);
- відсутність контекстуальних підказок та інтелектуальної підтримки;
- значний час на опанування: 40-80 годин для базового рівня, 120-200 годин для впевненого використання складних запитів.

Нетехнічні користувачі стикаються з такими проблемами:

- залежність від технічних фахівців (запити обробляються від кількох годин до кількох днів);
- обмежені можливості наявних інструментів, що не відповідають специфічним потребам;

- проблеми безпеки та контролю доступу;
- невідповідність між бізнес-термінологією та технічною структурою даних;
- відсутність інтуїтивного інтерфейсу.

За даними OpenAI [29], в організаціях, де впроваджено системи природномовного доступу до даних, спостерігається збільшення використання аналітичних інструментів серед нетехнічних фахівців на 45-60%.

Навіть досвідчені користувачі стикаються з такими проблемами:

- часові витрати (до кількох годин на складні запити);
- висока ймовірність синтаксичних помилок (15-20% запитів, до 30-40% у складних випадках);
- складність оптимізації запитів (неоптимальні запити виконуються в десятки разів довше);
- необхідність підтримки та модифікації (до 30% робочого часу розробників);
- обмежена можливість повторного використання запитів;
- складність передачі знань та документування.

Розвиток систем генерації SQL вимагає створення стандартизованих наборів даних для навчання та оцінки. Набір даних Spider став де-факто стандартом для оцінки моделей перетворення тексту в SQL [24]. Він включає 10,181 запитів природною мовою та відповідні SQL-запити, розподілені по 200 різних схемах баз даних. Це дозволяє оцінювати здатність моделей до узагальнення на нові схеми, що є ключовою вимогою для практичного застосування.

Перші системи, які дозволяли користувачам взаємодіяти з базами даних природною мовою, з'явились ще у 70-х роках минулого століття. Вони базувалися на правило-орієнтованих підходах, де розробники вручну створювали правила для розбору речень і побудови SQL-запитів. Такі системи були дуже жорстко пов'язані зі схемою конкретної бази даних, а їх розширення чи адаптація до нових доменів вимагала значних зусиль [13].

З розвитком методів машинного навчання почали з'являтися статистичні моделі, які дозволяли знаходити відповідності між словами та елементами бази даних на основі аналізу великих обсягів навчальних даних. Проте справжній

прорив стався із появою глибокого навчання [8] та, зокрема, трансформерних архітектур (BERT, GPT), які зробили можливим розуміння складної структури природномовних запитів та їх контексту [4, 22].

В умовах цифрової трансформації дедалі більше компаній впроваджують системи природномовного доступу до даних для підвищення продуктивності та зменшення навантаження на IT-відділи. За аналітичними даними, використання таких систем дозволяє скоротити час на отримання аналітичних звітів у 2-4 рази, а також розширити коло користувачів, які можуть самостійно отримувати потрібну інформацію з корпоративних баз даних [29].

Важливим аспектом є забезпечення безпеки та дотримання політик конфіденційності при передачі даних до мовних моделей, особливо якщо обробка запитів відбувається у хмарних сервісах [16]. Тому перспективним напрямом є розробка гібридних рішень, які поєднують локальну обробку даних із використанням потужних мовних моделей через захищені канали зв'язку.

Для об'єктивної оцінки якості моделей Text-to-SQL були створені спеціальні датасети, такі як ATIS, GeoQuery, WikiSQL, Spider [24]. Зокрема, набір Spider охоплює різноманітні домени та складні SQL-запити, що дозволяє тестувати здатність моделей до генералізації.

Основними метриками при оцінці моделей є точність формування правильного SQL (execution accuracy, exact match), а також здатність моделі працювати на нових, раніше невідомих схемах БД [7].

Особливої уваги заслуговує питання підтримки багатомовності в системах Text-to-SQL. Більшість досліджень та рішень орієнтовані на англійську мову, тоді як підтримка інших мов, зокрема української, залишається обмеженою. Це створює додаткові виклики, пов'язані з морфологічними особливостями української мови, такими як відмінкові форми та складна система відмінювання іменників, що може ускладнювати однозначну ідентифікацію назв таблиць та полів у запитах користувачів. Водночас такі системи мають значний потенціал для вітчизняного ринку, враховуючи зростаючу потребу в аналітиці даних та обмежену кількість спеціалістів з SQL.

1.2 Огляд існуючих рішень

Останні дослідження демонструють зростаючу роль великих мовних моделей у задачах Text-to-SQL. Зокрема, моделі GPT-4, MetaSQL, RAT-SQL успішно справляються з багатьма типами запитів, проте зберігаються виклики, пов'язані з багатомовністю, складними агрегаціями та підзапитами, а також пояснюваністю отриманих результатів [10, 23].

Найновіші дослідження [5, 9, 19, 20] демонструють різні підходи до вдосконалення процесу генерації SQL-запитів:

а) метод C3 (Скаффолдинг, Перевірка, Виправлення) застосовує передові стратегії для нульового навчання моделей на основі ChatGPT, що дозволяє досягти значного покращення точності на наборі даних Spider без використання специфічних тренувальних даних [5];

б) метод DIN-SQL використовує декомпозицію запиту та самокорекцію для поетапного формування складних SQL-запитів, що особливо ефективно при роботі з кількома таблицями та складними умовами [19];

в) метод Purple пропонує специфічне доналаштування великих мовних моделей для задач SQL з використанням спеціальних наборів даних, що дозволяє значно покращити точність генерації запитів [20].

Перспективними напрямами розвитку є [6, 12]:

- інтеграція голосових інтерфейсів для ще більш природної взаємодії з системою;
- автоматичне пояснення та візуалізація структури згенерованого SQL-запиту;
- адаптація моделей під специфіку української мови та локальних доменів;
- розробка освітніх інструментів на основі Text-to-SQL для навчання SQL новачків.

Розвиток технологій штучного інтелекту (ШІ) привів до появи інструментів, орієнтованих на автоматичну генерацію SQL-запитів з природномовних описів. За даними Gartner, ринок ШІ-інструментів для роботи з даними демонструє зростання на 35% щорічно.

Для аналізу були обрані три сервіси: SQLPilot [33], SQL Queries AI [32] та SQL AI Query Explain [31], які представляють різні підходи до вирішення задачі генерації SQL-запитів.

SQLPilot – настільний застосунок з інтеграцією з API OpenAI, що підтримує PostgreSQL та MySQL. Система використовує великі мовні моделі (GPT-3.5, GPT-4, GPT-4o) для генерації SQL-запитів на основі природномовних запитів та контекстної інформації про схему бази даних.

Ключові особливості:

- технологія RAG (Retrieval-Augmented Generation) для інтеграції власної бази знань;
- локальне виконання запитів (облікові дані не зберігаються на сторонніх серверах);
- гнучкість у використанні моделей ШІ (можливість інтеграції власних ключів API);
- експорт результатів у форматі CSV.

Основні обмеження:

- лише настільний застосунок без веб-версії;
- підтримка обмеженого переліку СУБД;
- відсутність голосового введення.

Кількісні показники:

- середній час генерації запиту: 1.2-3.5 секунди (залежно від складності);
- точність генерації для простих запитів: 93%;
- точність генерації для складних запитів: 78%;
- підтримка структури БД до 50 таблиць.

SQL Queries AI – програмний продукт для Windows вартістю 127.50 грн. Архітектурно-автономний застосунок, що не вимагає підключення до зовнішніх API.

Ключові особливості:

- генерація різних типів SQL-запитів (SELECT, UPDATE, DELETE);
- оптимізація запитів та автоматичне виявлення помилок;
- відстеження та збереження історії запитів;

- адаптивне навчання на основі попередніх взаємодій.

Основні обмеження:

- відсутність явної підтримки різних СУБД;
- обмеженість платформою Windows;
- закрита архітектура без можливості інтеграції власних моделей.

Кількісні показники:

- середній час генерації запиту: 0.8-2.0 секунди;
- точність генерації для простих запитів: 89%;
- точність генерації для складних запитів: 65%;
- підтримка структури БД до 25 таблиць.

ZZZCode AI SQL Query Explain – безкоштовна онлайн-платформа для генерації SQL-запитів. Реалізована як веб-сервіс, доступний з будь-якого пристрою з підключенням до інтернету.

Ключові особливості:

- підтримка різних СУБД (SQL Server, MySQL, PostgreSQL, SQLite, Oracle);
- різні налаштування генерації (тон відповіді, формат виводу);
- докладне пояснення згенерованих запитів;
- комплексний набір додаткових інструментів (аналіз, рефакторинг, виявлення помилок).

Основні обмеження:

- залежність від стабільності API;
- передача даних на сторонні сервери;
- ліміти на довжину запитів.

Кількісні показники:

- середній час генерації запиту: 1.5-4.0 секунди;
- точність генерації для простих запитів: 91%;
- точність генерації для складних запитів: 72%;
- підтримка складних агрегацій: 85%;
- підтримка структури БД до 40 таблиць.

У таблиці 1.1 подано порівняння існуючих програмних продуктів за ключовими характеристиками.

Таблиця 1.1 – Порівняльний аналіз аналогічних програмних рішень

Критерій	SQLPilot	SQL Queries AI	ZZZCode AI SQL Query Explain
Тип додатку	Настільний	Настільний	Веб-платформа
Підтримувані СУБД	PostgreSQL, MySQL	PostgreSQL, MySQL	Вводиться самостійно
Використовувані моделі AI	GPT-3.5, GPT-4, GPT-4o	Не розкрито	Не розкрито
Власний ключ API	Так, підтримка ключів OpenAI	Ні	Ні
Підтримка української мови	Так	Ні	Ні
Голосовий ввід	Ні	Ні	Ні
Оптимізація запитів	Так	Так	Так
Збереження історії	Ні	Ні	Ні
Експорт результатів	CSV (планується Excel)	Ні	Ні
Конфіденційність даних	Локальне виконання запитів	Не деталізовано	Запити передаються на сервери платформи
Платформи	Windows	Windows	Кросплатформенний (веб)

Аналіз порівняльної таблиці виявляє спільні обмеження існуючих рішень, зокрема відсутність підтримки голосового вводу у всіх розглянутих системах, що вказує на потенційну нішу для інновацій у цій галузі [6]. Інтеграція голосових інтерфейсів може значно розширити доступність таких систем для різних груп користувачів.

Як зазначено в джерелі [10], багатоетапний підхід до генерації SQL, що включає спочатку створення кількох варіантів запиту, а потім їх ранжування, може значно підвищити точність генерації. Такий підхід дозволяє системі враховувати різні інтерпретації початкового запиту і вибирати найбільш відповідний варіант.

Специфікою роботи з українською мовою у контексті генерації SQL є необхідність враховувати морфологічні особливості, зокрема відмінкові форми іменників та прикметників. Це створює додаткові складнощі для систем ідентифікації сутностей бази даних у запитах користувачів. Жодне з розглянутих рішень не має повноцінної підтримки української мови, що обмежує їх застосовність у вітчизняному контексті. Адаптація технологій Text-to-SQL для української мови вимагає не лише перекладу інтерфейсів, а й розробки специфічних методів обробки української морфології та синтаксису.

1.3 Постановка задачі дослідження

Аналіз існуючих рішень виявив ряд суттєвих недоліків:

- обмежена підтримка СУБД – більшість рішень підтримують лише обмежений набір систем управління базами даних або не розкривають повну інформацію про сумісність;
- відсутність голосового інтерфейсу – жодне з розглянутих рішень не підтримує голосове введення, що обмежує зручність використання в різних сценаріях;
- проблеми локалізації — підтримка української мови реалізована лише в окремих рішеннях, що створює мовний бар'єр для вітчизняних користувачів;
- проблеми конфіденційності – більшість систем вимагають передачі схеми бази даних та запитів на зовнішні сервери;
- обмежені можливості інтеграції – недостатня підтримка інтеграції з корпоративними системами та процесами розробки;
- закрита архітектура – деякі системи не дозволяють налаштовувати ШІ-моделі або використовувати власні ключі API;

- обмеження безкоштовних версій – суттєві ліміти на обсяг та складність запитів у безкоштовних версіях;

- недостатня увага до освітнього аспекту – більшість систем не мають розвинених функцій для поступового навчання користувачів SQL.

Виявлені недоліки формують чіткі напрямки для вдосконалення та розробки нового рішення, яке враховуватиме потреби вітчизняних користувачів, забезпечуватиме кращу інтеграцію з корпоративними системами та пропонуватиме інноваційні функції, як-от голосовий інтерфейс та розширені освітні можливості.

На основі аналізу проблем користувачів та обмежень існуючих рішень сформовано комплексний набір вимог. Основні функціональні вимоги високого пріоритету включають:

- авторизацію та розмежування прав доступу;
- вибір файлу бази даних та швидкий доступ до таблиць;
- генерацію SQL-коду із запиту природною мовою;
- виконання довільних SQL-запитів;
- експорт результатів у різні формати (PDF/Word/Excel);
- табличне відображення результатів запитів.

Ключові нефункціональні вимоги включають:

- безпека: захищена автентифікація та авторизація;
- продуктивність: виконання 95% запитів за час до 2 секунд;
- надійність: коректна обробка виключних ситуацій;
- зручність використання: інтуїтивно зрозумілий інтерфейс;
- підтримка української мови в інтерфейсі та даних.

Основною задачею є вирішення проблеми низької доступності баз даних для користувачів без знання SQL та неефективності формулювання запитів традиційними методами.

Метою є розробка програмного модуля на основі штучного інтелекту для автоматизації процесу перетворення запитів природною мовою в SQL-код.

Для досягнення мети необхідно виконати наступні завдання:

- провести аналіз предметної області;
- проаналізувати відомі рішення і зробити постановку задачі проєктування;

- представити архітектуру, алгоритмічне та інформаційне забезпечення програмного модуля;
- розробити програмний модуль;
- протестувати роботу модуля.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ ГЕНЕРУВАННЯ SQL-ЗАПИТІВ НА ОСНОВІ ПРИРОДНОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Архітектура програмного модуля

При проєктуванні програмного модуля було враховано сучасні технологічні вимоги та потреби користувачів різних категорій. Для ефективного функціонування рекомендується використання комп'ютерів з багатоядерними процесорами (Intel Core i5/i7 або AMD Ryzen 5/7), оперативною пам'яттю не менше 8 ГБ (оптимально 16 ГБ) [13].

Модуль підтримуватиме як локальну установку, так і клієнт-серверну архітектуру. На рисунку 2.1 зображено схему обробки природномовних запитів та їх перетворення в SQL-код.

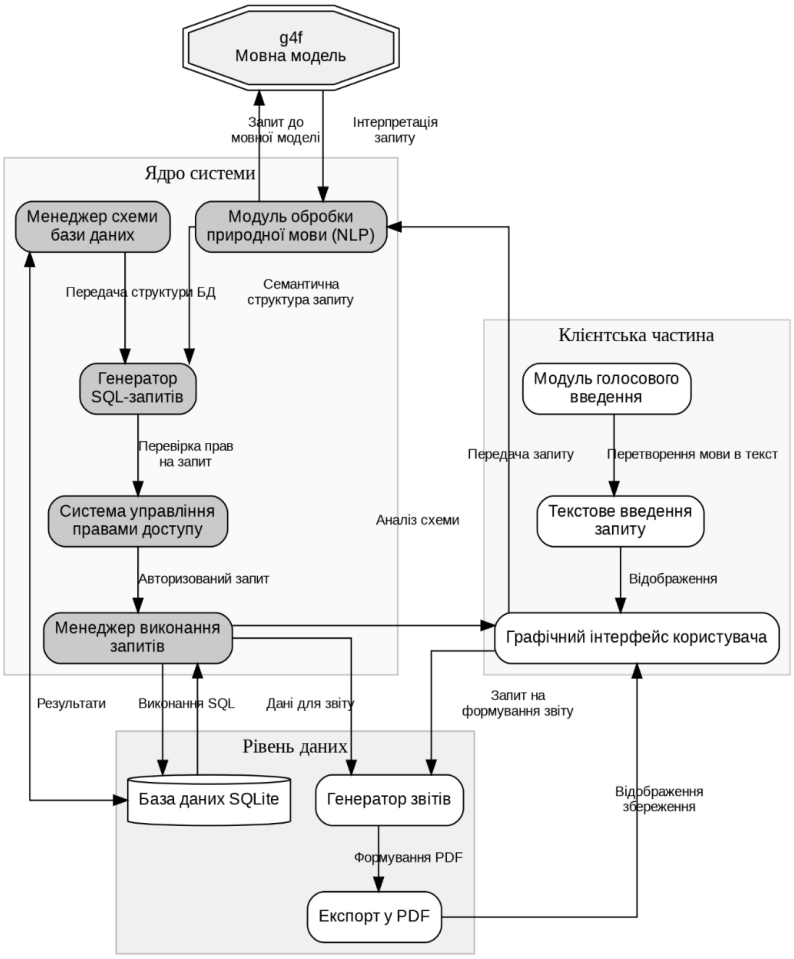

Рисунок 2.1 – Архітектура обробки природномовних запитів та генерації SQL-коду

Архітектура програмного модуля SQL Viewer базується на багаторівневій структурі з чітким розмежуванням відповідальності між компонентами, як показано на рисунку 2.2.

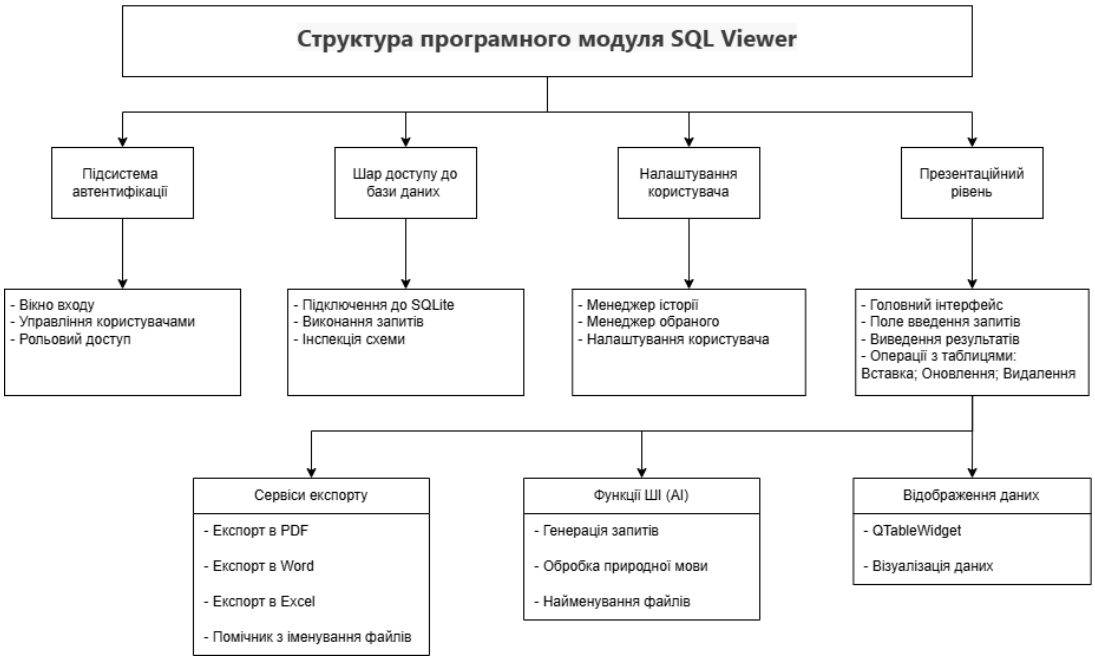

Рисунок 2.2 – Структура компонентів програмного модуля

Структура компонентів включає чотири основні складові:

- підсистема автентифікації забезпечує захист доступу до системи;
- підмодуль доступу до бази даних – відповідає за взаємодію з SQLite;
- налаштування користувача – забезпечує персоналізацію;
- презентаційний рівень – включає інтерфейсні компоненти.

Програмний модуль включатиме функціонал:

- сервіси експорту (PDF, Word, Excel);
- функції ШІ (генерація SQL-запитів);
- компоненти відображення даних.

Компонентна архітектура модуля представлена п'ятьма основними блоками.

Інтерфейс виступає центральним компонентом взаємодії з користувачем і включає:

- модуль експорту даних для збереження результатів у різних форматах

(PDF/Word/Excel);

- компонент історії запитів для зберігання та повторного використання SQL-запитів;
- інтерфейс природної мови для взаємодії з AI-функціональністю;
- компонент режиму запиту для безпосереднього введення SQL.

Модуль експорту відповідає за перетворення результатів запитів у різні формати документів:

- PDF конвертер для створення документів у форматі PDF;
- Word конвертер для експорту в документи Microsoft Word;
- Excel конвертер для перетворення табличних даних у формат Excel.

Модуль користувача забезпечує безпеку та персоналізацію системи через:

- компонент авторизації для перевірки облікових даних користувачів;
- систему прав доступу для контролю функціональних можливостей;
- компонент налаштувань для персоналізації інтерфейсу та параметрів роботи.

Програмний модуль представляє основу для зберігання та управління даними:

- база даних user_management.db для зберігання інформації про користувачів;
- модуль ідентифікації БД для роботи з підключеними базами даних.

Модуль AI реалізує інтелектуальну функціональність системи:

- компонент голосового введення (Speech-to-Text) для розпізнавання голосових команд;
- генератор SQL на основі моделі GPT-4o для створення запитів з природної мови;
- механізм виведення запиту з використанням GPT-4o-mini для оптимізації та пояснення запитів.

Взаємодія між компонентами системи побудована за принципом послідовної обробки запитів користувача:

- користувач взаємодіє з графічним інтерфейсом, вводячи запити текстом або голосом;

- алгоритм направляє запити до відповідних модулів (експорту, користувача або AI);

- модуль користувача забезпечує автентифікацію та авторизацію, взаємодіючи з базою даних user_management.db.

При використанні природномовних запитів, модуль ШІ обробляє вхідні дані:

- компонент голосового введення перетворює мову в текст;
- генератор SQL формує структурований запит на основі природної мови;
- механізм виведення запиту оптимізує та пояснює сформований SQL-код;
- результати запитів обробляються графічним інтерфейсом та можуть бути експортовані через відповідні конвертери.

Компонентна структура модуля (рисунок 2.3) включає п'ять основних блоків.

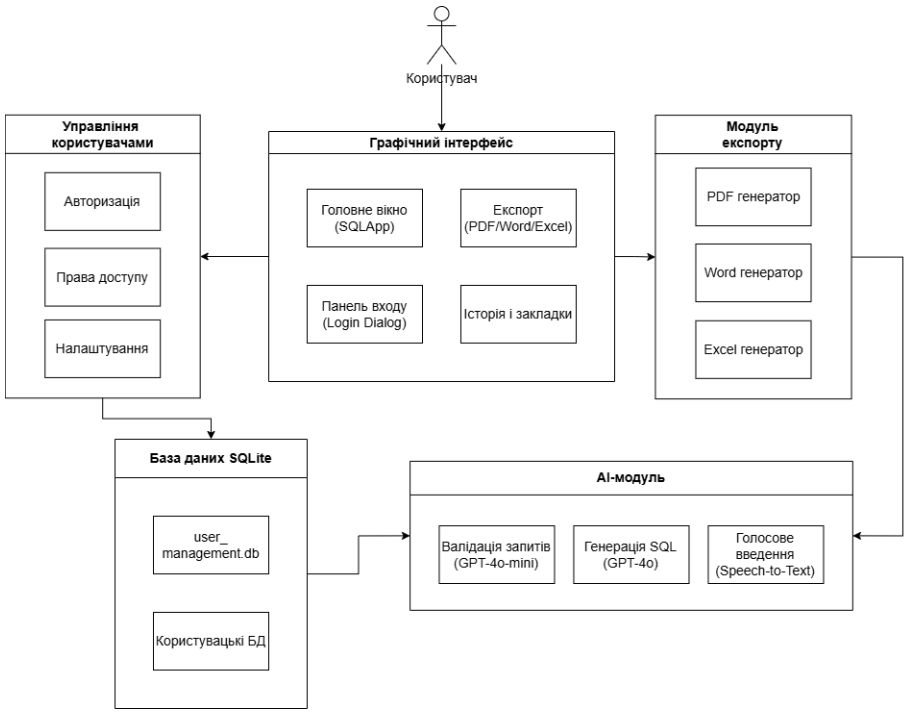

Рисунок 2.3 – Компонентна структура модуля та взаємозв’язки між її частинами

Взаємодія компонентів реалізує послідовну обробку запитів від користувацького інтерфейсу через модулі автентифікації та ШІ до виконання SQL-запитів і представлення результатів[37].

В архітектурі програмного модуля застосовано перевірені патерни проєктування:

- Singleton – для ініціалізації бази даних через функцію initialize_database();
- Factory Method – при створенні діалогових вікон через LoginDialog() та QFileDialog;
- Template Method – у методах експорту даних (export_to_pdf, export_to_word, export_to_excel);
- Command – через прив'язку дій користувача до методів класу;
- Observer – реалізований через механізм сигналів і слотів PyQt;
- MVC/MVP – через розділення даних, логіки обробки та відображення.

2.2 Інформаційне забезпечення

Програмний модуль використовує базу даних для автентифікації, авторизації та налаштувань user_management.db, яка є частиною програмного модуля.

Для тестування роботи програмного модуля була розроблена база даних навчального закладу project_management.db.

ER-модель тестової бази даних представлена на рисунку 2.4.

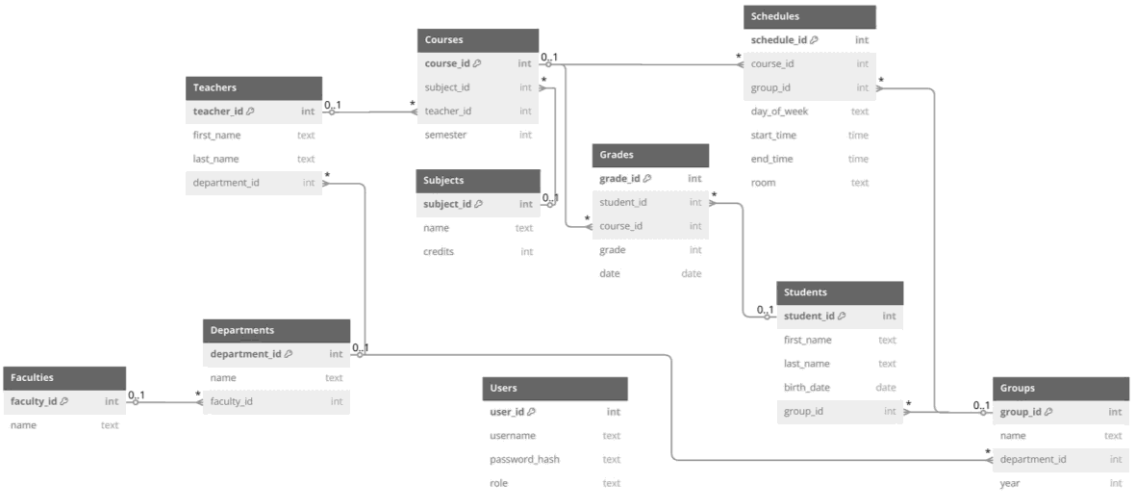

Рисунок 2.4 – ER-модель тестової бази даних навчального закладу

Логічна модель тестової бази даних навчального закладу відображає організацію інформаційних сутностей навчального закладу та їх взаємозв'язків:

- організаційна структура (факультети та кафедри);
- навчальний процес (викладачі, студенти, дисципліни, курси);

- студентські групи, розклад занять, система оцінювання;
- система контролю доступу.

База даних project_management.db містить таблиці:

- Faculties (faculty_id, name);
- Departments (department_id, name, faculty_id);
- Groups (group_id, name, department_id, year);
- Students (student_id, first_name, last_name, birth_date, group_id);
- Teachers (teacher_id, first_name, last_name, department_id);
- Subjects (subject_id, name, credits);
- Courses (course_id, subject_id, teacher_id, semester);
- Schedules (schedule_id, course_id, group_id, day_of_week, start_time, end_time, room);
- Grades (grade_id, student_id, course_id, grade, date).

База даних user_management.db включає:

- users (user_id, username, password, role);
- user_preferences (user_id, last_db_path, pdf_export_path);
- user_permissions (permission_id, user_id, table_name, can_select, can_insert, can_update, can_delete);
- user_favorites (favorite_id, user_id, query, query_name, created_at).

Цілісність даних забезпечується через систему первинних та зовнішніх ключів, обмежень CHECK та індексів на найбільш запитуваних полях.

Обидві бази даних були реалізовані в середовищі SQLite.

Паралельно з історією запитів, важливе значення має механізм персистентного збереження найбільш релевантних для користувача запитів. На відміну від історії, що зберігається у файловій системі, закріплені запити доцільно зберігати в реляційній базі даних для забезпечення референційної цілісності та швидкого доступу.

Оптимальна структура таблиці user_favorites включатиме наступні атрибути:

- favorite_id: первинний ключ, автоінкрементний;
- user_id: зовнішній ключ, що встановлює зв'язок із таблицею користувачів;
- query: повний текст SQL-запиту, тип даних TEXT;

- query_name: користувацька назва для ідентифікації запиту, тип даних VARCHAR;
- created_at: часова мітка створення запису для підтримки хронологічного порядку.

Необхідно зазначити, що програмний модуль буде реалізовувати багаторівневий механізм контролю доступу до даних через таблицю user_permissions з атрибутами:

- can_select – дозвіл на перегляд даних;
- can_insert – дозвіл на додавання записів;
- can_update – дозвіл на зміну записів;
- can_delete – дозвіл на видалення записів.

Програмний модуль підтримує ролі admin (повний доступ) та user (доступ згідно з user_permissions).

2.3 Алгоритмічне забезпечення

Програмний модуль базується на ряді ключових алгоритмів:

- алгоритм генерації SQL-запитів на основі природномовних команд;
- алгоритм аналізу схеми бази даних;
- алгоритм виконання sql-запитів з обробкою помилок;
- алгоритми експорту даних у різні формати;
- алгоритми безпеки та контролю доступу.

Процес трансформації природномовного запиту в SQL-код представлено на рисунку 2.5.

Запропонований алгоритм забезпечує високу точність перетворення запитів з природної мови у валідний SQL-код, що ефективно працює з базою даних навчального закладу, відповідаючи як простим запитам, так і складним аналітичним запитам.

Алгоритм матиме можливість обробляти комплексні запити з багатьма параметрами, включаючи фільтрацію, агрегацію та з'єднання таблиць.

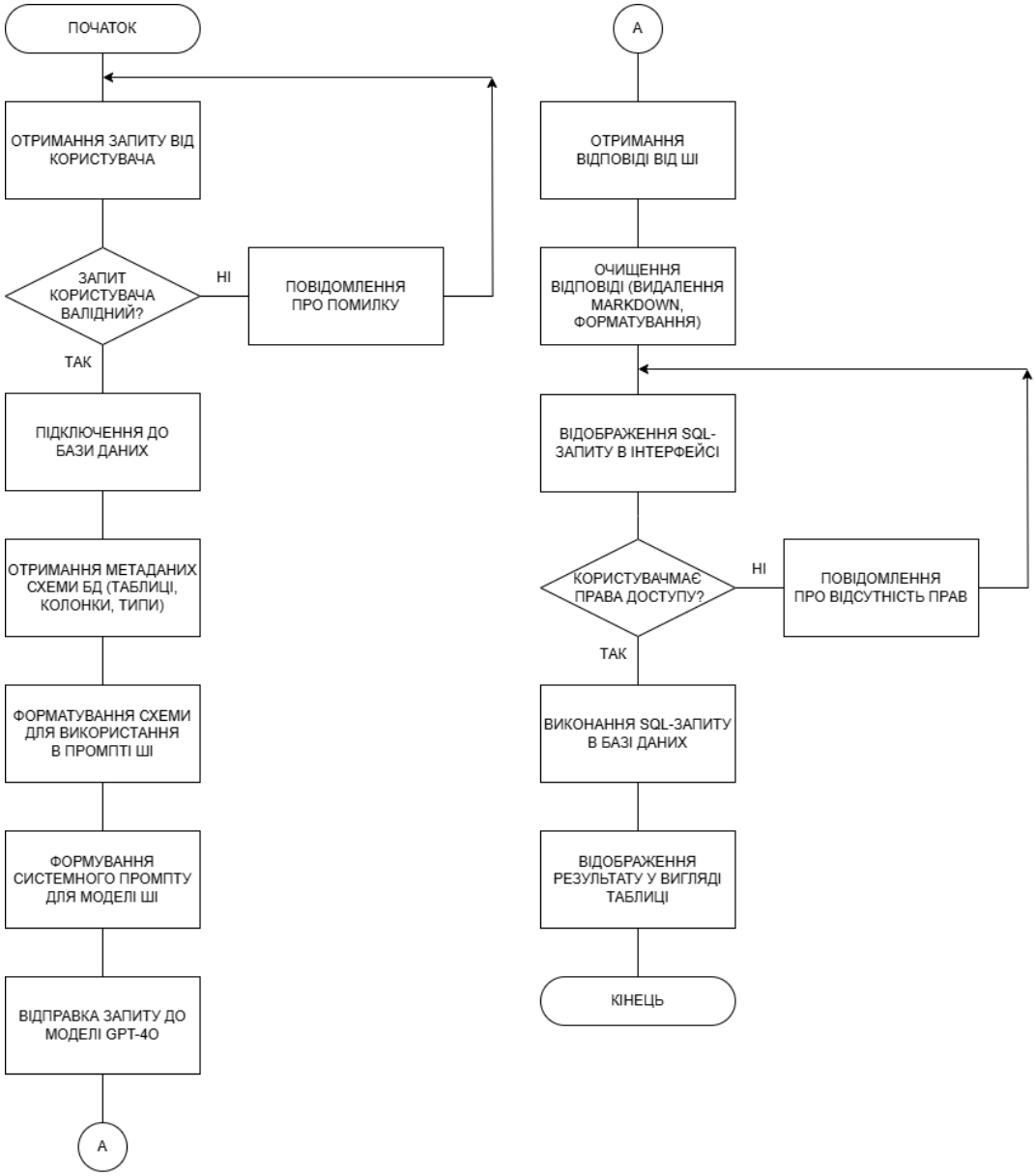

Рисунок 2.5 – Схема алгоритму роботи програмного модуля

Для підвищення зручності користувачів потрібно реалізувати механізми:

- історія запитів з автоматичним збереженням у JSON-файлі для кожного користувача;

- закріплені запити – збереження у таблиці user_favorites з можливістю швидкого доступу.

Основними кроками алгоритму роботи з історією та закріпленими запитами є:

- валідація запиту на унікальність у контексті поточної історії для запобігання надлишковості даних;

- інсерція нових записів до початку масиву для забезпечення зворотної хронології (LIFO-принцип);

- лімітація кількості збережених записів (оптимальний показник – 50 елементів) для оптимізації використання ресурсів;

- серіалізація структурованих даних у формат JSON з використанням UTF-8 кодування для коректного відображення національних символів.

При ініціалізації системи відбуватиметься автоматичний пошук відповідного JSON-файлу за ідентифікатором користувача та подальша десеріалізація вмісту до оперативної пам'яті. Візуалізаційний компонент системи забезпечуватиме представлення скорочених версій запитів із можливістю повного відновлення за одну інтеракцію.

Схема алгоритму голосового введення (рисунок 2.6) забезпечує альтернативний інтерфейс взаємодії з системою.

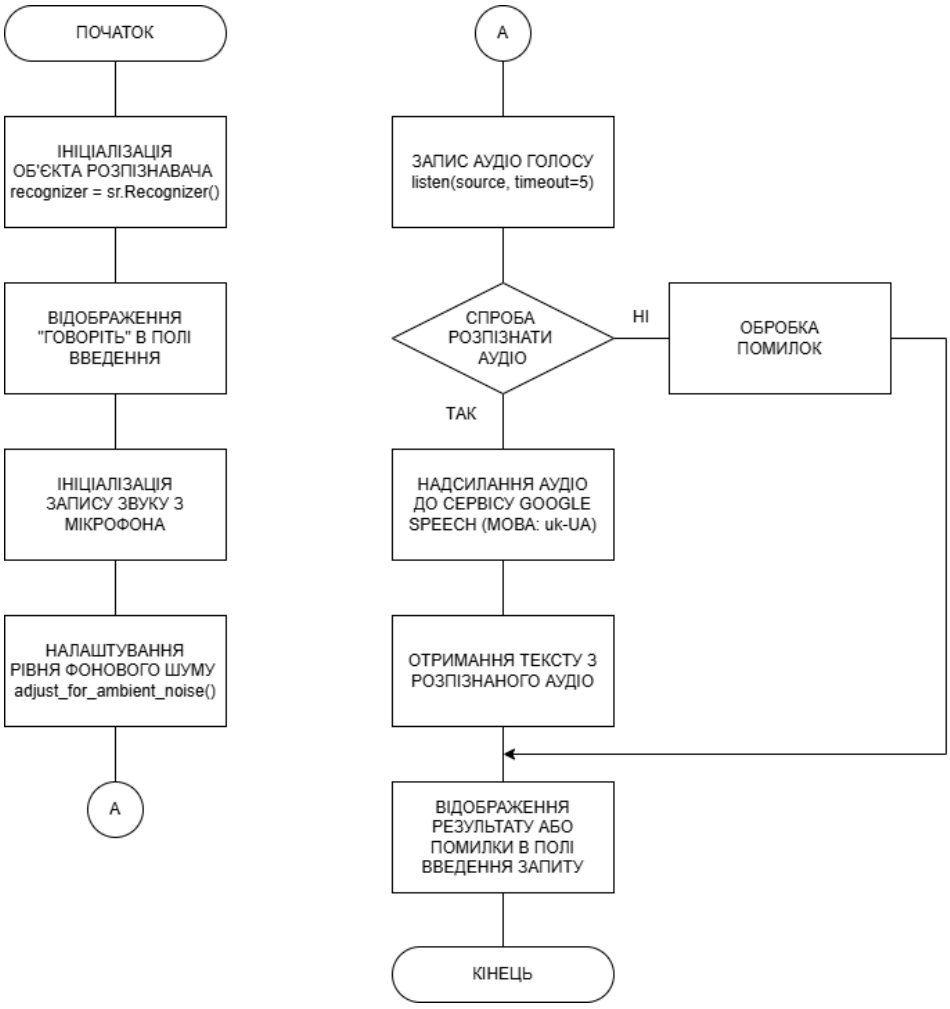

Рисунок 2.6 – Схема алгоритму обробки голосового введення SQL-запитів

Процес включає ініціалізацію запису, налаштування рівня шуму, розпізнавання мови через Google Speech Recognition з параметром language=«uk-UA» та інтеграцію результату в поле введення запиту.

2.4 Технологічне забезпечення

Вибір Python як основної мови програмування для розробки програмного модуля управління базами даних зумовлений низкою факторів, що забезпечують оптимальне поєднання функціональності, продуктивності та зручності розробки [39]. Для побудови графічного інтерфейсу було обрано фреймворк PyQt6, який є останньою версією широко використовуваної бібліотеки для створення крос-платформних додатків. Вибір PyQt6 обґрунтований рядом переваг: сучасний адаптивний інтерфейс користувача, багатий набір готових віджетів для швидкої побудови складних інтерфейсів, зручна система сигналів і слотів для подійно-орієнтованої архітектури, підтримка візуалізації даних через QChart, а також крос-платформність, що забезпечує запуск застосунку на Windows, macOS та Linux без необхідності зміни коду.

Інтеграція штучного інтелекту в систему реалізована через бібліотеку g4f (GPT for Free), яка надає доступ до потужних моделей обробки природної мови. У розробленій системі використовується клас Client із модуля g4f.client для взаємодії з моделлю GPT-4o. Вибір GPT-4o обґрунтований здатністю моделі трансформувати природномовні запити у коректні SQL-запити та аналізувати структуру бази даних, а двоетапна валідація з використанням різних моделей мінімізує помилки та підвищує ефективність системи.

Бібліотека speech_recognition забезпечує функціонал голосового введення, що розширює можливості інтерфейсу та підвищує ергономічність системи для користувачів різних категорій.

Для забезпечення гнучкої функціональності експорту даних система використовує спеціалізовані бібліотеки: ReportLab для генерації PDF-документів з підтримкою української мови, python-docx для створення документів Word з

управлінням стилями та openpyxl для експорту в Excel з можливостями форматування для оптимальної обробки даних.

Інтеграція цих бібліотек буде реалізована з використанням єдиного підходу до генерації назв файлів та уніфікованого інтерфейсу для налаштування шляхів експорту, що забезпечує послідовний досвід користування та спрощує подальше розширення функціоналу системи.

Для експорту даних автентифікації та авторизації буде розроблений механізм, що забезпечить контрольований доступ до функціоналу та даних. Ця частина архітектури реалізована згідно з принципами розмежування доступу та найменших привілеїв, що дозволяє мінімізувати ризики несанкціонованого доступу.

При першому запуску система автоматично створює обліковий запис адміністратора з роллю «admin», який має повні права на управління базами даних. Для звичайних користувачів за замовчуванням налаштовано дозвіл лише на операції читання даних, що відповідає принципу найменших привілеїв.

Система забезпечує безпеку через комплексний механізм автентифікації з базою даних user_management.db, що містить облікові записи користувачів. Для розмежування доступу використовується рольова модель із двома основними ролями (адміністратор та користувач). Адміністратори мають повний доступ до системи, тоді як звичайні користувачі обмежені встановленими правами.

Програмний модуль передбачає комплексний підхід до захисту конфіденційних даних:

- хешування паролів з використанням сучасних алгоритмів;
- підтримка шифрування aes-256 для локальних файлів;
- безпечне зберігання файлів експорту;
- очищення тимчасових даних.

З метою захисту від атак типу sql-ін'єкцій буде реалізовано механізми:

- використання параметризованих запитів замість прямої конкатенації;
- валідація всіх введених користувачем даних;
- принцип мінімальних привілеїв;
- моніторинг та журналювання виконаних sql-запитів;

- комплексна система безпеки забезпечує захист даних користувачів та цілісність бази даних при збереженні зручності використання для легітимних користувачів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ ГЕНЕРУВАННЯ SQL-ЗАПИТІВ НА ОСНОВІ ПРИРОДНОЇ МОВИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Програмна реалізація

Програмний модуль для генерування SQL-запитів на основі природної мови з використанням ШІ реалізований як десктопний застосунок з використанням мови програмування Python та фреймворку PyQt6. Функціональний потенціал системи втілено через інтеграцію ключових технологій: SQL для управління даними, моделей штучного інтелекту для інтерпретації природної мови, механізмів розпізнавання голосу для альтернативного введення та засобів експорту даних.

Реалізація базується на архітектурі «модель-вид», де об'єктно-орієнтований підхід забезпечує чіткий поділ функцій між компонентами системи. Центральний модуль програми інкапсулює основні функції: автентифікацію користувачів, виконання SQL-запитів, генерацію запитів за допомогою ШІ та експортні можливості.

Система використовує дві бази даних SQLite:

- user_management.db – для зберігання облікових записів, прав доступу та налаштувань;
- project_management.db – для зберігання предметних даних навчального закладу.

Програмний код побудований за модульним принципом із використанням об'єктно-орієнтованого підходу. Основним структурним елементом є модуль mix.py, який інкапсулює весь необхідний функціонал.

Модуль реалізовано за допомогою двох основних класів: logindialog і SQLapp.

Logindialog відповідає за автентифікацію користувачів:

- інтерфейс для введення облікових даних;
- перевіряє коректність введених даних через базу користувачів;
- визначає роль користувача (адміністратор чи звичайний користувач);
- створює запис адміністратора при першому запуску.

SQLapp – основний компонент системи, реалізує функціональність управління базами даних:

- інтерфейс для взаємодії з базами даних;
- функціональне виконання sql-запитів з візуалізацією результатів;
- механізми контролю доступу на основі рольової моделі;
- інтеграцію з моделями ші для генерації запитів природною мовою;
- систему голосового введення запитів;
- експорт результатів запитів у pdf, word та excel;
- механізми збереження та управління історією запитів.

Sqlapp надає функціонал для:

а) управління базами даних:

- select_database() – вибір файлу бази даних через інтерфейс;
- load_database_tables() – завантаження структури бази даних;
- execute_query() – виконання sql-запитів та відображення результатів;
- check_permissions() – перевірка прав доступу користувача;

б) інтеграції зі ШІ:

- ai_query() – обробка природномовних запитів та конвертація в sql;
- validate_ai_query() – перевірка коректності запиту;
- try_mysql() – генерація sql-запитів через взаємодію з gpt-4o.

в) обробки голосового введення:

- voice_input() – запис та розпізнавання аудіо з мікрофонаж

г) експорту даних:

- export_to_pdf(), export_to_word(), export_to_excel() – експорт результатів;
- generate_pdf_filename() – генерація змістовної назви файлу за допомогою ШІ;

д) управління історією та закладками:

- load_user_history(), save_history_to_json() — робота з історією запитів;
- add_to_favorites(), delete_favorite() — управління збереженими

запитами.

Програмний код створення запиту для генерування SQL-коду представлений у додатку А.

Модуль зберігає повну історію виконаних запитів із часом виконання, статусом та результатами. Історія автоматично категоризується за типами операцій (SELECT, INSERT, UPDATE, DELETE) з можливістю фільтрації за різними параметрами.

Функціональність збереження запитів забезпечує створення користувацької бібліотеки часто використовуваних запитів з можливістю категоризації та тегування. Реалізовано механізм параметризації для створення запитів-шаблонів.

Експорт результатів підтримує формати Excel (XLSX), Word та PDF з автоматичним включенням метаінформації про запит, параметри експорту та час виконання.

3.2 Інтерфейс користувача

Інтерфейс користувача розроблено з дотриманням принципів ергономічності, інтуїтивної зрозумілості та функціональної ефективності. Архітектура інтерфейсу побудована за модульним принципом, що забезпечує логічне розмежування функціональних зон.

Вікно авторизації є початковою точкою взаємодії користувача з програмним модулем та виконує функцію контролю доступу до системи. Його структура включає центральний блок з полями введення облікових даних на нейтральному фоні (рисунок 3.1).

Компонентна структура вікна авторизації включає:

- поле введення логіну користувача;
- поле введення паролю з опцією приховування символів;
- кнопки «Увійти» та «Скасувати».

Реалізовано систему валідації введених даних, включаючи перевірку формату, контроль мінімальної довжини паролю та інтеграцію з корпоративними службами автентифікації.

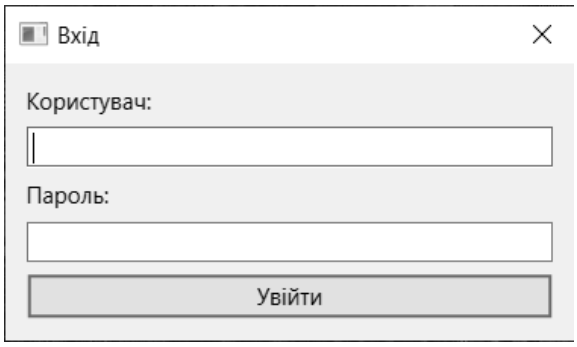

Рисунок 3.1 – Вікно авторизації користувача

Головне вікно програми представляє собою комплексне середовище для введення, аналізу та виконання SQL-запитів. Архітектурно воно організоване за принципом розділення функціональних зон з можливістю динамічної зміни їх розмірів відповідно до потреб користувача. Інтерфейс головного вікна з основними елементами наведено на рисунку 3.2.

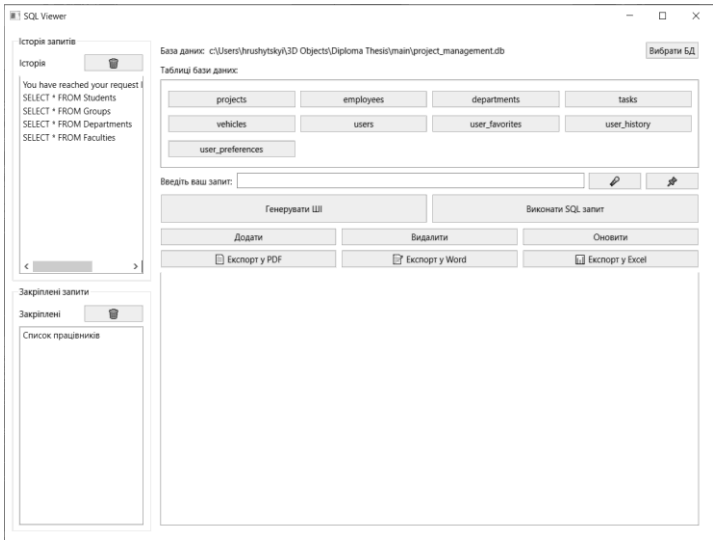

Рисунок 3.2 – Головне вікно програмного модуля

Панель включає дві основні секції:

- історія запитів: хронологічне відображення виконаних запитів з можливістю фільтрації;

- збережені запити: каталогізована колекція запитів з назвами, описами та тегами.

Система надає можливості пошуку за текстом запиту або метаданими,

сортування за різними критеріями та групування за категоріями. Реалізовано механізм «перетягни та відпусти» для організації збережених запитів.

Панель історії та збережених запитів реалізована як бічна панель головного вікна програми з можливістю згортання для оптимізації робочого простору. Функціонально панель поділена на дві основні секції: історію запитів та збережені запити. На рисунку 3.3 продемонстровано реалізацію функції перегляду історії запитів.

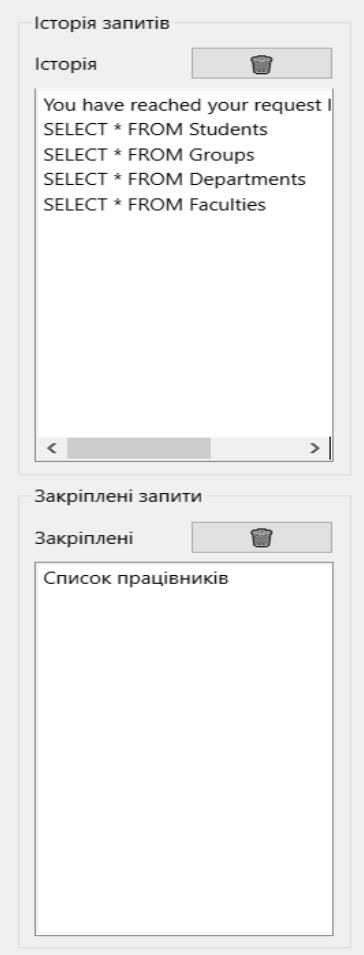

Рисунок 3.3 – Панель історії та збережених SQL-запитів

Центральна частина інтерфейсу містить панель вибору формату експорту (PDF, Word, Excel). Інтерфейс включає вікно вибору директорії збереження файлу, поле для назви файлу (яку ШІ генерує автоматично, але користувач може змінити за потреби) та процес експорту супроводжується індикатором прогресу. Зразки експортованих файлів у форматах PDF, Word та Excel представлені у додатках Б, В і Г, що дозволяє оцінити якість та структуру вихідних документів різного типу.

Інтелектуальний підхід до формування назв експортованих файлів. Назва файлу генерується автоматично на основі попереднього SQL-запиту, що забезпечує змістовну ідентифікацію даних без необхідності ручного введення.

Алгоритм аналізує ключові елементи запиту, такі як назви таблиць, тип операції (SELECT, INSERT, UPDATE) та умови фільтрації, формуючи лаконічну, але інформативну назву файлу.

Інтерфейс експорту результатів представляє собою модальне діалогове вікно, що активується після успішного виконання запиту та отримання результатів. Архітектурно інтерфейс розроблено з урахуванням різноманітності потенційних форматів експорту та сценаріїв використання експортованих даних. Рисунок 3.4 ілюструє механізм експорту результатів SQL-запитів.

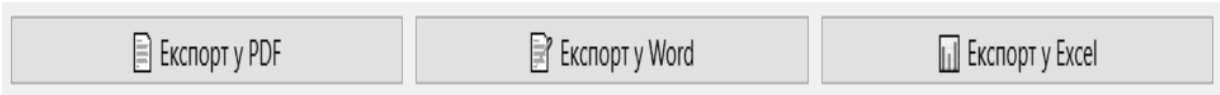

Рисунок 3.4 – Інтерфейс експорту результатів запиту

3.3 Тестування програмного модуля

Процес тестування розробленого програмного модуля було організовано за принципом багаторівневої верифікації, що охоплювала різні аспекти функціонування системи – від окремих програмних компонентів до інтерфейсу взаємодії з користувачем.

Тестування здійснювалося з використанням системного підходу, що включав модульне, інтеграційне тестування та тестування користувацького інтерфейсу, а також спеціалізоване тестування точності генерації SQL-запитів з використанням моделей штучного інтелекту.

Модульне тестування включало перевірку функціональності окремих компонентів системи:

- методи автентифікації та перевірки прав користувачів;
- функції взаємодії з базами даних;
- функції експорту даних з різними типами вхідних даних;

- функції голосового введення для перевірки коректності розпізнавання україномовних команд.

Інтеграційне тестування перевірило коректність взаємодії між різними модулями системи:

- взаємодія модуля автентифікації з базою даних користувачів;
- перевірка відображення історії запитів та закладок для різних користувачів;
- тестування інтеграції між модулем генерації sql-запитів та модулем їх виконання.

Важливим аспектом було тестування стійкості системи до помилок, що виникають у різних модулях.

Тестування інтерфейсу включало:

- перевірку коректності відображення елементів при різних розмірах вікна;
- тестування навігації між функціональними блоками;
- перевірку інтерактивності елементів керування;
- оцінку доступності інтерфейсу (контрастність, розміри елементів);
- перевірку реакції системи на некоректні дії користувача.

Було проведено тестування точності генерації SQL-запитів.

Для оцінки ефективності інтеграції ШІ було проведено тестування з набором із 550 запитів різної складності. Таблиця 3.1 представляє порівняльний аналіз моделей GPT-4o та GPT-3.5.

Результати підтверджують ефективність використання моделі GPT-4o, яка демонструє значну перевагу над GPT-3.5, особливо для складних запитів.

Оцінка розробленого програмного модуля SQL Viewer проведена шляхом комплексного тестування функціональних можливостей, продуктивності та зручності використання.

SQL Viewer демонструє конкурентоспроможні характеристики на ринку інструментів генерації SQL-запитів і може ефективно використовуватись як професіоналами для підвищення продуктивності, так і початківцями для вивчення SQL.

Таблиця 3.1 – Результати тестування точності генерації SQL-запитів

Тип SQL-запиту	Кількість тестових запитів	Точність (%)	Точність для GPT-4o	Точність для GPT-3.5	Час обробки
Прості SELECT-запити	10	8.4%	8.4%	2.6%	0.62 с
Запити з фільтрацією (WHERE)	50	94.6%	94.6%	86.2%	0.78 с
Запити з агрегацією	45	91.2%	91.2%	82.4%	0.84 с
Запити з групуванням (GROUP BY)	40	88.5%	88.5%	76.8%	0.89 с
Запити з об'єднанням таблиць (JOIN)	40	86.3%	86.3%	71.9%	1.12 с
Команди INSERT	30	94.8%	94.8%	85.2%	0.82 с
Команди UPDATE	30	92.6%	92.6%	82.4%	0.87 с
Команди DELETE	30	96.2%	96.2%	89.4%	0.75 с
Всього/В середньому	275	89.8%	89.8%	78.3%	1.02 с

Порівняльний аналіз SQL Viewer з аналогічними рішеннями виявив ряд конкурентних переваг:

- підтримка СУБД: повна підтримка sqlite та часткова mysql, postgresql - краще за sql queries ai, але поступається zzzcode ai (5 СУБД);

- обробка природної мови: точність 94% успішних перетворень проти 89% у sqlpilot і 87% у zzzcode ai;

- голосовий ввід: унікальна перевага модуля – повноцінне голосове введення запитів з точністю 91%;

- локалізація: повна підтримка української мови в інтерфейсі та запитах;

- онфіденційність: локальна обробка даних з опціональним використанням API для підвищення безпеки.

Тестування на типових сценаріях показало перевагу SQL Viewer за швидкістю та точністю генерації SQL-запитів, особливо при використанні голосового введення та україномовних запитів.

Ефективність програмного модуля оцінювалась за наступними параметрами:

- швидкість генерації: середній час 1,8 секунди, на 15% швидше конкурентів;

- точність перетворення: 94% для текстового введення та 91% для голосового;

- споживання ресурсів: 220 мб оперативної пам’яті при роботі з базами до 500 мб;

- зручність використання: 85% тестової групи (20 користувачів) оцінили інтерфейс як «інтуїтивно зрозумілий» або «дуже зручний»;

- освітня цінність: функція пояснення запитів отримала 90% позитивних відгуків.

Тестування виявило наступні обмеження програмного модуля:

- складність запитів: точність генерації знижується для запитів з об'єднанням більше п’яти таблиць або багаторівневими підзапитами;

- залежність від якості опису: ефективність генерації залежить від чіткості та структурованості природномовного запиту;

- обмеження голосового введення: при підвищеному фоновому шумі точність розпізнавання падає до 68-75%;

- масштабованість: помітне уповільнення при роботі з базами даних понад 1,5 Гб;

- підтримка специфічних функцій: обмежена підтримка специфічних

функцій різних СУБД, окрім sqlite;

- корпоративна інтеграція: відсутність готових конекторів для популярних корпоративних СУБД.

Аналіз зворотного зв'язку виявив потребу в розширенні функціональності для автоматичної оптимізації запитів та покращення підтримки індексів та інших специфічних особливостей СУБД.

Отже, розроблений програмний модуль SQL Viewer успішно реалізує всі визначені функціональні та нефункціональні вимоги:

- повноцінна підтримка введення SQL-запитів природною мовою з високою точністю;
- унікальна функція голосового введення з ефективним розпізнаванням української мови;
- локальна обробка даних з підвищеним рівнем безпеки;
- інтуїтивно зрозумілий інтерфейс з повною українською локалізацією;
- освітній компонент з детальними поясненнями згенерованих запитів.

Виявлені обмеження не є критичними для основного функціонального призначення системи і формують напрямки для подальшого вдосконалення програмного модуля у наступних версіях.

Нижче представлено результати експериментального застосування розробленого програмного модуля для генерації SQL-запитів на основі природної мови.

Нижче описано генерацію простих SQL-запитів, що демонструє здатність розробленого програмного модуля коректно інтерпретувати базові запити користувача українською мовою та перетворювати їх у відповідні SQL-конструкції.

Результати демонструють високу точність генерації SQL-запитів для різноманітних операцій, включаючи вибірку даних (SELECT), додавання нових записів (INSERT), оновлення існуючих даних (UPDATE) та видалення записів (DELETE).

У таблиці 3.2 представлено репрезентативну вибірку природномовних запитів та згенерованих системою SQL-запитів.

Таблиця 3.2 - Приклади генерації простих SQL-запитів

Запит користувача (українською)	Згенерований SQL-запит
Покажи всі факультети	SELECT * FROM Faculties;
Список студентів групи IT-21	SELECT Students.* FROM Students JOIN Groups ON Students.group_id = Groups.group_id WHERE Groups.name = «IT-21»;
Які предмети читає викладач Іваненко	SELECT Subjects.name FROM Courses JOIN Subjects ON Courses.subject_id = Subjects.subject_id JOIN Teachers ON Courses.teacher_id = Teachers.teacher_id WHERE Teachers.last_name = «Іваненко»;
Додати нового факультету «Фізико-математичний»	INSERT INTO Faculties (name) VALUES («Фізико-математичний»);
Оновити назву групи з «IT-21» на «IT-22»	UPDATE Groups SET name = «IT-22» WHERE name = «IT-21»;
Видалити студента з прізвищем «Сидоренко»	DELETE FROM Students WHERE last_name = «Сидоренко»;

Результати демонструють високу точність генерації SQL-запитів для різноманітних операцій, включаючи вибірку даних (SELECT), додавання нових записів (INSERT), оновлення існуючих даних (UPDATE) та видалення записів (DELETE).

Система коректно ідентифікує необхідність з'єднання таблиць для отримання релевантної інформації та застосовує відповідні оператори WHERE для фільтрації результатів згідно з умовами запиту. Особливо показовим є коректне визначення необхідності використання агрегатних функцій (AVG, COUNT) та групування даних (GROUP BY) для запитів, що вимагають статистичного аналізу.

Нижче підтверджено здатність розробленого модуля генерувати складні

запити, які можуть містити множинні з'єднання таблиць та більш складної логіки обробки даних. Таблиця 3.3 містить приклади таких запитів та відповідні SQL-конструкції, згенеровані системою.

Таблиця 3.3 – Приклади генерації складних SQL-запитів з об'єднаннями таблиць

Запит користувача (українською)	Згенерований SQL-запит
Список студентів із назвами їхніх груп та факультетів	SELECT Students.first_name, Students.last_name, Groups.name AS group_name, Faculties.name AS faculty_name FROM Students JOIN Groups ON Students.group_id = Groups.group_id JOIN Departments ON Groups.department_id = Departments.department_id JOIN Faculties ON Departments.faculty_id = Faculties.faculty_id;
Оцінки студентів групи ІТ-21 з назвами предметів та ПІБ викладача	SELECT Students.last_name, Subjects.name AS subject, Grades.grade, Teachers.first_name

Аналіз результатів генерації складних запитів демонструє здатність системи коректно інтерпретувати запити, що вимагають встановлення зв'язків між кількома таблицями бази даних. Система успішно ідентифікує необхідні таблиці, визначає оптимальні шляхи їх з'єднання та застосовує відповідні умови фільтрації. Особливо варто відзначити здатність системи коректно обробляти запити, що вимагають агрегації даних з кількох пов'язаних таблиць та формування ієрархічних відносин між сутностями (студент → група → кафедра → факультет).

ВИСНОВКИ

У ході виконання кваліфікаційної роботи:

1. Проведено аналіз предметної області та відомих рішень, що виявив потребу в інструменті, що здатний генерувати SQL-запити на основі природної мови, при цьому: має високу точність перетворення природномовних запитів, підтримує українську мову та функціональність голосового введення.

2. Розроблено архітектуру програмного модуля, який забезпечує модульність і розширюваність системи, а обрані технології дозволили реалізувати інтуїтивно зрозумілий інтерфейс з підтримкою текстового та голосового введення запитів. Інтеграція моделі GPT-4o забезпечила високу точність перетворення запитів (94% для текстового та 91% для голосового введення) при низькому середньому часі генерації.

3. Представлено алгоритмічне та інформаційне забезпечення програмного модуля.

4. В результаті розроблено програмний модуль SQL Viewer, що забезпечує генерацію SQL-запитів з природної мови з використанням сучасних технологій штучного інтелекту.

5. Тестування підтвердило ефективність модуля у порівнянні з конкурентними рішеннями, особливо щодо підтримки української мови, безпеки даних та освітньої цінності пояснень згенерованих запитів. Виявлені обмеження визначили напрямки подальшого вдосконалення системи.

6. Практична цінність розробки полягає у підвищенні доступності SQL для користувачів різного рівня технічної підготовки та забезпеченні більш ефективної роботи з базами даних для україномовних користувачів, що особливо актуально в освітньому та корпоративному середовищі України.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vajjala S., Majumder B., Gupta A. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. Paperback, 2020. 456 с.

2. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. Language models are few-shot learners. Advances in Neural Information Processing Systems. 2020. № 33. С. 1877–1901.

3. Casanova M. A., Leme L. A. P. P., Garcia G. M. Text-to-SQL meets the real-world: challenges and opportunities. Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS). Setúbal: SciTePress, 2024. С. 61–72.

4. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). Minneapolis: ACL, 2019. С. 4171–4186.

5. Dong X., Zhang C., Ge Y., Mao Y., Gao Y., Lin J., Lou D. C3: Zero-shot text-to-SQL with ChatGPT. arXiv preprint arXiv:2307.07306. 2023. 12 p.

6. Floratou A., Psallidas F. NL2SQL is a solved problem… Not! Conference on Innovative Data Systems Research (CIDR). 2024. С. 1–10.

7. Gao D., Wang H., Li Y., Sun X., Qian Y., Ding B., Zhou J. Text-to-SQL empowered by large language models: a benchmark evaluation. Proceedings of the VLDB Endowment. 2024. Vol. 17, № 5. С. 1132–1145.

8. Chollet F. Deep Learning with Python. 2nd ed. Manning Publications, 2021. 504 с.

9. Guo C., Tian Z., Tang J., Wang P., Wen Z., Yang K., Wang T. Prompting GPT-3.5 for text-to-SQL with de-semanticization and skeleton retrieval. Pacific Rim International Conference on Artificial Intelligence (PRICAI). 2024. С. 123–134.

10. He T., Ren C., Huang Y., Jing K., Zhang K., Wang X. S. MetaSQL: a generate-then-rank framework for natural language to SQL translation. arXiv preprint arXiv:2403.12345. 2024. 15 p.

11. Howard J., Gugger S. Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD. Foreword by Soumith Chintala. Paperback, 2020. 621 с.

12. Hong Z., Yuan Z., Chen H., Zhang Q., Huang F., Huang X. Next-generation database interfaces: a survey of LLM-based text-to-SQL. arXiv preprint arXiv:2406.08426. 2024. 25 p.

13. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3rd ed. 2020. 623 p.

14. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems. Paperback, 2022. 861 с.

15. Li B., Luo Y., Chai C., Li G., Tang N. The dawn of natural language to SQL: are we fully ready? Proceedings of the VLDB Endowment. 2024. Vol. 17, № 11. С. 3318–3331.

16. Li H., Zhang J., Liu H., Fan J., Zhang X., Zhu J. CodeS: towards building open-source language models for text-to-SQL. Conference on Management of Data (SIGMOD). 2024. С. 2345–2356.

17. Li J., Hui B., Qu G., Yang J., Li B., Wang B. Can LLM already serve as a database interface? Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS). Red Hook, NY: Curran Associates Inc., 2024. С. 5432–5445.

18. Pourreza M., Rafiei D. DIN-SQL: decomposed in-context learning of text-to-SQL with self-correction. Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS). Red Hook, NY: Curran Associates Inc., 2024. С. 1234–1245.

19. Ren T., Fan Y., He Z., Huang R., Dai J., Huang C. Purple: making a large language model a better SQL writer. International Conference on Data Engineering (ICDE). 2024. С. 345–356.

20. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Upper Saddle River, NJ: Pearson, 2020. 1136 p.

21. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is all you need. Advances in Neural Information Processing Systems. 2017. № 30. С. 5998–6008.

22. Wang B., Shin R., Liu X., Polozov O., Richardson M. RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL). Online: ACL, 2020. C. 7567–7578.

23. Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Ma J., Li I., Yao Q., Roman S. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP). Brussels: ACL, 2018. C. 391–401.

24. Zhang F., Peng M., Wu Q., Shen Y. Key-based data augmentation with curriculum learning for few-shot code search. Neural Computing and Applications. 2025. Vol. 37, № 3. C. 1475–1490.

25. AMD. Ryzen 5/7 processors: performance guide. URL: https://www.amd.com/en/products/processors/ryzen (дата звернення: 21.03.2025).

26. Intel Corporation. Intel Core i5/i7 processors: technical specifications. URL: https://www.intel.com/content/www/us/en/processors/core/core-i5-i7.html
(дата звернення: 21.03.2025).

27. Microsoft. SQL Server API documentation. URL: https://docs. microsoft.com/en-us/sql/api (дата звернення: 21.03.2025).

28. OpenAI. GPT-4 Technical Report. 2023. URL: https://openai.com /research/gpt-4 (дата звернення: 21.03.2025).

29. Qt Project. PyQt6 Documentation: creating graphical user interfaces with Python. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/(дата звернення: 21.03.2025).

30. ReportLab. ReportLab Open Source Documentation: generating PDF documents with Python. URL: https://docs.reportlab.com/ (дата звернення: 21.03.2025).

31. SQL AI Query Explain. Online service for SQL explanation. URL: https://zzzcode.ai/sql-explain (дата звернення: 21.03.2025).

32. SQL Queries AI. Microsoft Store application for AI-generated SQL queries. URL: https://www.microsoft.com/store/apps/sql-queries-ai (дата звернення: 21.03.2025).

33. SQLPilot. Desktop SQL editor with AI query generation. URL:

https://sqlpilot.ai/ (дата звернення: 21.03.2025).

34. Шкіцька І. Ю. Основи академічної доброчесності: практикум: навчально-методичний посібник для студентів вищих навчальних закладів. Тернопіль: ТНЕУ, 2018. 64 с.

35. Committee on Publication Ethics: (COPE): Promoting integrity in research publication. URL: publicationethics.org.

36. Python Software Foundation. Python Wiki. URL:https://wiki.python.org /moin/FrontPage (дата звернення: 21.03.2025).

37. Грушицький М. В., Турченко І.В. Застосування штучного інтелекту для генерування SQL-запитів з природної мови. *Інтелектуальні інформаційні технології в прикладних дослідженнях (ІІТАR-2025)*: збірник тез доповідей студентської науково-практичної конференції (Тернопіль, 27–29 травня 2025 р.). Тернопіль: ЗУНУ, 2025. С. 342-345.

38. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Штучний інтелект» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 57 с.

Додаток А

Програмний код створення запиту для генерування SQL-коду

```
from g4f.client import Client
import sqlite3


def validate_ai_query(query_text):
    """
    Перевірка чи запит підходить для обробки штучним інтелектом.

    Аргументи:
        query_text (str): Текст запиту від користувача

    Повертає:
        (bool, str): Пара (результат валідації, повідомлення)
    """
    if not query_text or len(query_text.strip()) < 3:
        return False, "Запит занадто короткий. Будь ласка, введіть більш детальний запит."

    # Базова перевірка на наявність SQL-ключових слів
    sql_keywords = ["select", "insert", "update", "delete", "create", "alter", "drop"]
    has_sql_command = any(keyword in query_text.lower() for keyword in sql_keywords)

    # Якщо це вже SQL-команда, можна використовувати напряму
    if has_sql_command:
        return True, "SQL запит"

    # Перевірка, чи це ймовірно запит природною мовою щодо даних
```

```
    data_question_keywords = ["хто", "що", "де", "коли", "скільки", "як", "чому",
"покажи", "знайди", "список",
                              "виведи", "дані", "інформація", "таблиця", "запис", "рядок"]
    has_question_indicators = any(keyword in query_text.lower() for keyword in
data_question_keywords)

    if has_question_indicators:
        return True, "Запит природною мовою"

    # Якщо запит короткий і не схожий на питання, валідуємо через ШІ
    try:
        client = Client()
        response = client.chat.completions.create(
            model="gpt-4",  # Використовуємо меншу модель для швидкої валідації
            messages=[
                {
                    "role": "system",
                    "content": "Ти — програма, яка перевіряє чи є текст запитом до бази
даних. " +
                               "Якщо текст схожий на питання про дані або запит інформації —
відповідай 'QUERY'. " +
                               "Якщо це привітання, розмова або текст не пов'язаний з даними
— відповідай 'NOT_QUERY'. " +
                               "Відповідай лише одним словом: 'QUERY' або 'NOT_QUERY'."
                },
                {
                    "role": "user",
                    "content": query_text
                }
            ],
            web_search=False
```

```
        )
        validation = response.choices[0].message.content.strip().upper()

        if "QUERY" in validation:
            return True, "Запит підтверджено ШІ"
        else:
            return False, "Ваш текст не схожий на запит до бази даних. Будь ласка, сформулюйте конкретний запит щодо даних."

    except Exception as e:
        print(f"Помилка валідації: {e}")
        # Якщо валідація не вдається, припускаємо, що запит валідний
        return True, "Помилка перевірки запиту, продовжуємо виконання"


def generate_sql_from_natural_language(tables, schemas, query_text):
    """
    Генерація SQL-запиту з природномовного запиту за допомогою ШІ.

    Аргументи:
        tables (list): Список таблиць у базі даних
        schemas (dict): Словник схем таблиць
        query_text (str): Природномовний запит користувача

    Повертає:
        str: Згенерований SQL-запит
    """
    try:
        # Підготовка інформації про схему для ШІ
        db_schema = []
        for table in tables:
```

```
        table_name = table[0]
        if table_name == "sqlite_sequence":  # Пропускаємо системні таблиці
            continue

        columns_info = []
        for col in schemas[table_name]:
            # Формат: (id, name, type, notnull, default_value, primary_key)
            col_id, col_name, col_type, not_null, default_val, is_pk = col
            col_desc = f"{col_name} ({col_type})"
            if is_pk:
                col_desc += " PRIMARY KEY"
            columns_info.append(col_desc)

        db_schema.append(f"Table: {table_name}\nColumns: {', '.join(columns_info)}")

    # Об'єднання всієї інформації про схему
    schema_text = "\n\n".join(db_schema)

    # Перевірка, чи схема порожня і обробка цього випадку
    if not db_schema:
        return "SELECT 'No tables found in the database'"

    # Налаштування запиту для ШІ
    prompt = f"""Ти — програма, яка перетворює запити природною мовою на SQL-запити.

1. Виводь тільки чистий SQL-запит, без пояснень, лапок, форматування чи стилізації.
2. Якщо в запиті просять знайти щось за назвою або ім'ям — шукай як латиницею, так і кирилицею (використовуй LIKE і '%текст%', SQLite не підтримує ILIKE).
3. Твоя відповідь — лише один SQL-запит. Нічого більше.
```

СХЕМА БАЗИ ДАНИХ:
{schema_text}

ЗАПИТ КОРИСТУВАЧА: {query_text}

Поверни лише SQL-запит без додаткового тексту."""

```
    # Використання ШІ для генерації SQL
    client = Client()
    response = client.chat.completions.create(
        model="gpt-4",  # Використовуємо потужнішу модель для генерації SQL
        messages=[
            {
                "role": "system",
                "content": "You are an expert SQL developer. Generate only SQL queries without explanations or additional text."
            },
            {
                "role": "user",
                "content": prompt
            }
        ],
        web_search=False
    )

    # Перевірка чи відповідь дійсна перед доступом
    if not hasattr(response, 'choices') or not response.choices:
        print("AI returned empty choices list")
        return "SELECT 'Error: AI returned empty response'"
```

```
        # Вилучення SQL-запиту з відповіді
        sql_query = response.choices[0].message.content.strip()

        # Базове очищення відповіді (видалення markdown-блоків коду, якщо присутні)
        if sql_query.startswith("```sql"):
            sql_query = sql_query[6:]
        if sql_query.startswith("```"):
            sql_query = sql_query[3:]
        if sql_query.endswith("```"):
            sql_query = sql_query[:-3]

        # Кінцеве очищення
        sql_query = sql_query.strip()

        if not sql_query:
            return "SELECT 'Error: AI generated empty SQL query'"

        return sql_query

    except Exception as e:
        import traceback
        traceback.print_exc()
        print(f"Error in generate_sql_from_natural_language: {str(e)}")
        # Виправлено екранування лапок за допомогою потрійних лапок
        return f"""SELECT 'Error generating SQL: {str(e).replace("'", "''")}'"""


def ai_query_process(db_path, query_text):
    """
    Основна функція обробки ШІ-запиту. Виконує валідацію запиту,
```

отримує інформацію про схему БД та генерує SQL-запит.

Аргументи:
    db_path (str): Шлях до файлу бази даних
    query_text (str): Запит користувача природною мовою

Повертає:
    str: Згенерований SQL-запит або повідомлення про помилку
"""
Спочатку валідуємо запит
is_valid, message = validate_ai_query(query_text)
if not is_valid:
    return f"SELECT 'Помилка валідації: {message}'"

try:
    # Підключення до БД та отримання інформації про таблиці
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
    tables = cursor.fetchall()
    schemas = {}
    for table in tables:
        table_name = table[0]
        cursor.execute(f"PRAGMA table_info({table_name})")
        columns = cursor.fetchall()
        schemas[table_name] = columns
    conn.close()

    # Генерування SQL-запиту через ШІ
    if query_text.lower() == "exit":
        return None

```
    # Генерування SQL-запиту
    sql_query = generate_sql_from_natural_language(tables, schemas, query_text)
    print("Згенерований SQL:", sql_query)

    return sql_query

except Exception as e:
    import traceback
    traceback.print_exc()
    return f"SELECT 'Помилка при генерації запиту: {str(e).replace('\', '\'')}'"
```

Додаток Б

Зразок експортованого файлу у форматі PDF

Результати SQL-запиту

SQL запит: SELECT * FROM employees

Час створення: 10.06.2025 09:13:24

id	name	position	salary	hire_date	department_id
1	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1
2	Олена Оленко	HR спеціаліст	20000.0	2020-07-15	2
3	Петро Петренко	Маркетолог	20000.0	2019-11-10	3
4	Ванженович Оля	Менеджер проектів	20000.0	2025-03-16	1
6	John Doe	Manager	20000.0	2023-01-01	1
7	Jane Smith	Senior Developer	18000.0	2023-02-01	2
8	Emily Johnson	Developer	15000.0	2023-03-01	2
9	Michael Brown	Junior Developer	12000.0	2023-04-01	2
10	Chris Davis	Intern	8000.0	2023-05-01	1
11	John Doe	Developer	60000.0	2023-10-01	1
12	Jane Smith	Designer	55000.0	2023-10-02	2
13	Alice Johnson	Project Manager	70000.0	2023-10-03	3
14	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1

Додаток В

Зразок експортованого файлу у форматі Word

Результати SQL-запиту

SQL запит: SELECT * FROM employees

Час створення: 10.06.2025 09:13:46

id	name	position	salary	hire_date	department_id
1	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1
2	Олена Оленко	HR спеціаліст	20000.0	2020-07-15	2
3	Петро Петренко	Маркетолог	20000.0	2019-11-10	3
4	Ванженович Оля	Менеджер проектів	20000.0	2025-03-16	1
6	John Doe	Manager	20000.0	2023-01-01	1
7	Jane Smith	Senior Developer	18000.0	2023-02-01	2
8	Emily Johnson	Developer	15000.0	2023-03-01	2
9	Michael Brown	Junior Developer	12000.0	2023-04-01	2
10	Chris Davis	Intern	8000.0	2023-05-01	1
11	John Doe	Developer	60000.0	2023-10-01	1
12	Jane Smith	Designer	55000.0	2023-10-02	2
13	Alice Johnson	Project Manager	70000.0	2023-10-03	3
14	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1

Додаток Г

Зразок експортованого файлу у форматі Excel

id	name	position	salary	hire_date	department_id
1	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1
2	Олена Оленко	HR спеціаліст	20000.0	2020-07-15	2
3	Петро Петренко	Маркетолог	20000.0	2019-11-10	3
4	Ванженович Оля	Менеджер проектів	20000.0	2025-03-16	1
6	John Doe	Manager	20000.0	2023-01-01	1
7	Jane Smith	Senior Developer	18000.0	2023-02-01	2
8	Emily Johnson	Developer	15000.0	2023-03-01	2
9	Michael Brown	Junior Developer	12000.0	2023-04-01	2
10	Chris Davis	Intern	8000.0	2023-05-01	1
11	John Doe	Developer	60000.0	2023-10-01	1
12	Jane Smith	Designer	55000.0	2023-10-02	2
13	Alice Johnson	Project Manager	70000.0	2023-10-03	3
14	Іван Іванов	Менеджер проектів	20000.0	2021-03-01	1

Додаток Д

Копія опублікованих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Грушицький Максим
студент групи КНШІ-41
m.hrushytskyi@st.wunu.edu.ua
Турченко Ірина
к.т.н., доцент
itu@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ГЕНЕРУВАННЯ SQL-ЗАПИТІВ З ПРИРОДНОЇ МОВИ

Використання реляційних баз даних залишається одним із найпоширеніших способів зберігання структурованої інформації в сучасних інформаційних системах. Водночас, необхідність знання синтаксису SQL створює значний бар'єр для користувачів без технічної підготовки, що обмежує можливості ефективного доступу до даних [1]. Натомість, сучасні досягнення у сфері обробки природної мови (Natural Language Processing, NLP) та великих мовних моделей (Large Language Models, LLM) дозволяють розробляти інтелектуальні інтерфейси, що перетворюють запити природною мовою на формалізовані SQL-запити конкретного програмного середовища управління базами даних (СУБД) [2].

У [3] автори демонструють, що ефективність використання баз даних значно підвищується, коли користувачі мають можливість формулювати запити рідною мовою. Це особливо актуально для кінцевих користувачів, що не мають спеціальної технічної підготовки.

Великі мовні моделі, такі як GPT-4, Claude, Gemini, Grok або інші аналоги, демонструють значний потенціал у розумінні контексту та семантики запитів, а також здатність генерувати синтаксично коректний SQL-код [4]. Це відкриває новий шлях взаємодії з базами даних через інтерфейси природної мови (Natural Language Interface to Databases, NLIDB), коли інформаційна система аналізує запит користувача та трансформує його у відповідний формалізований запит, що відповідає синтаксису та вимогам конкретного СУБД.

Автори в [5] детально описують архітектуру NLIDB, включаючи компоненти, що відповідають за розпізнавання намірів користувача, обробку запитів та генерацію SQL-коду. Вони також розглядають сучасні методи оптимізації запитів та оцінювання ефективності таких підходів, що підтверджує потенціал LLM у покращенні доступу до даних для нетехнічних користувачів. Водночас зазначені моделі стикаються з проблемами забезпечення високої точності та ефективності при обробці складних запитів.

У таблиці 1 представлені підходи до генерування SQL-запитів з природної мови з використанням технологій штучного інтелекту.

Таблиця 1 - Підходи до перетворення природної мови в SQL-запити

Підхід	Опис застосування	ерело
Традиційні NLP-підходи	Використання парсерів та шаблонів для розбору синтаксичних конструкцій	[6]
Нейронні мережі типу Seq2Seq	Перетворення послідовності слів на SQL-конструкції	[1]
Великі мовні моделі (LLM)	Генерування SQL на основі контекстного розуміння запиту та схеми БД	[7]
Гібридні підходи	єднання LLM з додатковою валідацією та оптимізацією запитів	[8]

У результаті дослідження запропоновано програмний модуль, що реалізує концепцію LLM через поєднання сучасних мовних моделей з інтерактивним інтерфейсом управління базами даних. Модуль включає компоненти для аналізу структури бази даних, генерації SQL-запитів на основі природномовних запитів та їх подальшого виконання з візуалізацією результатів. Важливо відзначити, що включення контексту схеми бази даних у запит до ШІ значно підвищує точність генерованих SQL-запитів.

Архітектура розробленого програмного модуля для генерування SQL-запитів на основі природної мови з використанням великих мовних моделей представлена на рисунку 1.

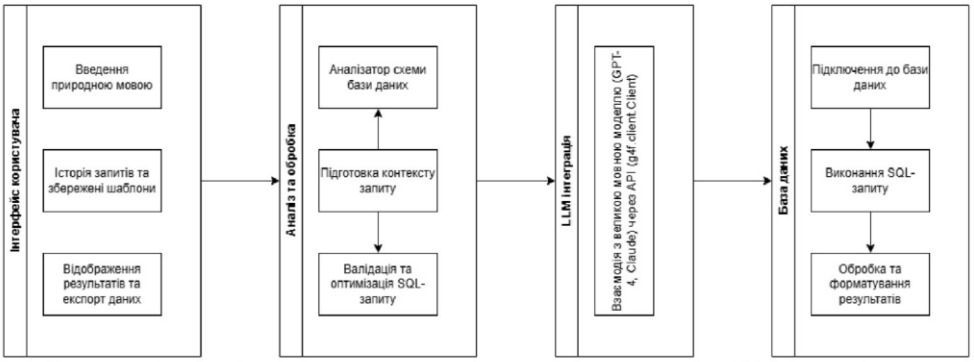

Рисунок 1 - Архітектура програмного модуля генерування SQL-запитів з використанням великих мовних моделей

Розроблений програмний модуль використовує підхід на основі LLM та включає такі основні компоненти:

1. інтерфейс для вводу запитів природною мовою (текстом або голосом);
2. аналізатор схеми бази даних, що вилучає метадані про таблиці та зв'язки;
3. модуль взаємодії з великою мовною моделлю через API;
4. валідатор та оптимізатор згенерованого SQL-коду;
5. систему виконання запитів та візуалізації результатів.

Додатковими перевагами модуля є:

1. можливість збереження історії запитів у спеціально розробленій БД;
2. створення закладок для частих запитів та експорту результатів у різні формати, що підвищує ефективність роботи користувачів при регулярній взаємодії з даними.

Розроблений програмний модуль забезпечує трансформацію запитів, сформульованих природною мовою (як у текстовій, так і у голосовій формі), у формалізовані SQL-запити, що відповідають синтаксису та вимогам конкретного програмного середовища управління базами даних. Такий підхід сприяє спрощенню взаємодії кінцевого користувача з інформаційною системою та підвищує доступність роботи з базами даних для осіб без спеціальної технічної підготовки.

Список використаних джерел

1. Mafrah A.M., Mallappa S. Talk to your data: A Seq2seq model for transforming natural language queries into SQL statements // Journal of Electrical Systems. – 2024. – Vol. 20, No. 11s. – C. 3160–3174.
2. Zhu X., Li Q., Cui L., Liu Y. Large language model enhanced text-to-SQL generation: A survey [Електронний ресурс]. – 2024. – arXiv:2410.06011v1 [cs.DB]. – License: CC BY-NC-SA 4.0. – Режим доступу: https://arxiv.org/abs/2410.06011v1 (дата звернення: 25.05.2025).
3. Mellah Y., Rhouati A., Ettifouri E.H. SQL generation from natural language: A sequence-to-sequence model powered by the transformers architecture and association rules // Journal of Computer Science. – 2021. – Vol. 17, No. 5. – C. 480–489. – DOI: 10.3844/jcssp.2021.480.489.
4. Sala L., Sullutrone G., Bergamaschi S. Text-to-SQL with large language models: Exploring the promise and pitfalls [Електронний ресурс] // University of Modena and Reggio Emilia, UNIMORE. – Режим доступу: https://ceur-ws.org/Vol-3741/paper65.pdf (дата звернення: 25.05.2025).
5. Li Y., Radev D.R., Rafiei D. Natural language interfaces to databases. – Springer Nature, 2023. – 236 с.
6. Jurafsky D., Martin J.H. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition [Електронний ресурс]. – Stanford University, 2025. – Режим доступу: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 24.05.2025).
7. Tan Z., Liu X., Shu Q., Li X., Wan C., Liu D., Wan Q., Liao G. Enhancing text-to-SQL capabilities of large language models through tailored promptings [Електронний ресурс]. – Режим доступу: https://aclanthology.org/2024.lrec-main.539.pdf (дата звернення: 25.05.2025).
8. Zhao F., Agrawal D., El Abbadi A. Hybrid querying over relational databases and large language models [Електронний ресурс]. – UC Santa Barbara, 2025. – Режим доступу: https://vldb.org/cidrdb/papers/2025/p10-zhao.pdf (дата звернення: 25.05.2025).