МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ «СЕРВІСИ ШТУЧНОГО ІНТЕЛЕКТУ»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Є. О. Кубрак
(ініціали та прізвище)
Керівник в.о. завідувача кафедри комп'ютерних наук, к.пед.н. Пшенична О. С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А. Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кубраку Єгору Олеговичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Проєктування та розробка інформаційно-довідкової системи «Сервіси штучного інтелекту»

керівник роботи Пшенична Олена Станіславівна, к.пед.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 « грудня 2025 року № 1741-с

2. Строк подання студентом роботи 05.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи розробки інформаційно-довідкових систем та сервісів штучного інтелекту.
2. Проєктування інформаційно-довідкової системи «Сервіси штучного інтелекту».
3. Розробка та тестування інформаційно-довідкової системи.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.02.2026	Виконано
2.	Збір вихідних даних.	23.02.2026	Виконано
3.	Обробка методичних та теоретичних	05.04.2026	Виконано
	джерел.		
4.	Розробка першого та другого розділу.	10.04.2026	Виконано
5.	Розробка третього розділу.	12.05.2026	Виконано
6.	Оформлення та нормоконтроль	05.06.2026	Виконано
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2026	

.

Студент _____ Є. О. Кубрак
(підпис) (ініціали та прізвище)

Керівник роботи _____ О. С. Пшенична
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування та розробка інформаційно-довідкової системи "Сервіси штучного інтелекту"»: 77 с., 18 рис., 11 табл., 25 джерел, 2 додатки.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ІНФОРМАЦІЙНО-ДОВІДКОВА СИСТЕМА, КАТАЛОГІЗАЦІЯ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ПОШУК, РЕЛЯЦІЙНА МОДЕЛЬ, СИСТЕМА УПРАВЛІННЯ ДАНИМИ, ТЕСТУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єктом дослідження є процеси збору, зберігання та надання інформації про сервіси штучного інтелекту.

Предметом дослідження є методи та засоби проєктування і розробки інформаційно-довідкових систем для каталогізації сервісів штучного інтелекту.

Метою роботи є проєктування та розробка інформаційно-довідкової системи «Сервіси штучного інтелекту».

Методи дослідження – аналіз наукової літератури та існуючих інформаційно-довідкових систем, системний аналіз предметної області, методи проєктування програмних систем і баз даних, принципи об’єктно-орієнтованого програмування, а також методи модульного та функціонального тестування програмного забезпечення.

У роботі проаналізовано підходи до побудови інформаційно-довідкових систем і каталогізації сервісів штучного інтелекту. Спроєктовано архітектуру системи, базу даних та інтерфейс користувача, обґрунтовано вибір технологій. Реалізовано основні функціональні модулі, включно з пошуком, фільтрацією та каталогізацією сервісів. Проведене тестування підтвердило коректність роботи системи та відповідність вимогам.

SUMMARY

Bachelor’s Qualifying Thesis «Design and Development of the Information and Reference System "Artificial Intelligence Services"»: 77 pages, 18 figures, 11 tables, 25 sources, 2 supplements.

DATABASE, WEB APPLICATION, INFORMATION AND REFERENCE SYSTEM, CATALOGING, USER INTERFACE, SEARCH, RELATIONAL MODEL, DATA MANAGEMENT SYSTEM, TESTING, ARTIFICIAL INTELLIGENCE.

The object of research is the processes of collecting, storing, and providing information about artificial intelligence services.

The subject of research is methods and tools for designing and developing information and reference systems for cataloging artificial intelligence services.

The aim of the thesis is to design and develop an information and reference system “Artificial Intelligence Services”.

Research methods include analysis of scientific literature and existing information and reference systems, system analysis of the subject area, methods of software and database design, principles of object-oriented programming, as well as methods of unit and functional software testing.

The thesis analyzes approaches to building information and reference systems and cataloging artificial intelligence services. The system architecture, database, and user interface were designed, and the choice of technologies was justified. The main functional modules were implemented, including search, filtering, and cataloging of artificial intelligence services. Testing confirmed the correctness of the system’s operation and compliance with the requirements.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Теоретичні основи розробки інформаційно-довідкових систем та сервісів штучного інтелекту 9
1.1 Поняття та особливості інформаційно-довідкових систем 9
1.2 Аналіз сучасних інформаційно-довідкових систем 11
1.3 Визначення вимог до інформаційно-довідкової системи «Сервіси штучного інтелекту»..... 15
2 Проєктування інформаційно-довідкової системи «Сервіси штучного інтелекту»..... 19
2.1 Аналіз предметної області та проєктування структури системи 19
2.2 Проєктування бази даних і функціональних модулів 23
2.3 Проєктування користувацького інтерфейсу та вибір технологій розробки.. 31
3 Розробка та тестування інформаційно-довідкової системи..... 38
3.1 Реалізація програмного продукту 38
3.2 Розробка функціоналу пошуку та каталогізації сервісів штучного інтелекту..... 42
3.3 Тестування системи та аналіз результатів її роботи 46
3.4 Інструкція користувача..... 49
Висновки 52
Перелік посилань..... 54
Додаток А Лістинг реалізації програмного продукту 57
Додаток Б Інтерфейс системи 73

ВСТУП

Стрімкий розвиток технологій штучного інтелекту (ШІ) суттєво впливає на всі сфери сучасного суспільства, зокрема освіту, бізнес, науку та інформаційні технології. Сьогодні сервіси на основі штучного інтелекту активно використовуються для автоматизації рутинних процесів, генерації текстового та візуального контенту, аналізу великих обсягів даних, підтримки прийняття рішень, програмування та навчання. Щороку кількість таких сервісів зростає, а їх функціональність стає дедалі складнішою та різноманітнішою.

Водночас зростання кількості сервісів штучного інтелекту створює проблему їх систематизації та швидкого пошуку. Користувачам складно орієнтуватися у великій кількості інструментів, оскільки інформація про них є розпорошеною по різних джерелах, часто неструктурованою або неповною. Це ускладнює вибір оптимального сервісу для конкретного завдання та знижує ефективність їх використання. У зв’язку з цим виникає потреба у створенні спеціалізованих інформаційно-довідкових систем, які дозволяють централізовано зберігати, структурувати та надавати інформацію про сервіси штучного інтелекту.

Інформаційно-довідкові системи є важливим класом інформаційних систем, які призначені для накопичення, обробки та надання довідкових даних користувачам у зручному форматі. Використання сучасних вебтехнологій дозволяє реалізувати такі системи у вигляді інтерактивних платформ із можливістю пошуку, фільтрації, категоризації та оновлення даних у реальному часі. Це робить їх ефективним інструментом для роботи з великими інформаційними масивами.

Актуальність теми полягає у необхідності розробки сучасної інформаційно-довідкової системи «Сервіси штучного інтелекту», яка дозволить користувачам швидко отримувати структуровану інформацію про існуючі AI-

інструменти, їх можливості, призначення та сфери застосування. Така система є корисною як для студентів і викладачів, так і для розробників, дослідників та фахівців, які використовують технології штучного інтелекту у своїй діяльності.

Метою роботи є проєктування та розробка інформаційно-довідкової системи «Сервіси штучного інтелекту».

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область та існуючі інформаційно-довідкові системи;
- визначити вимоги до інформаційно-довідкової системи;
- виконати проєктування архітектури системи, бази даних та інтерфейсу користувача;
- обґрунтувати вибір технологій розробки;
- реалізувати програмний продукт;
- розробити функціонал пошуку та каталогізації сервісів;
- провести тестування системи та проаналізувати результати її роботи.

Об’єктом дослідження є процеси збору, зберігання та надання інформації про сервіси штучного інтелекту.

Предметом дослідження є методи та засоби проєктування і розробки інформаційно-довідкових систем для каталогізації сервісів штучного інтелекту.

Методи дослідження – аналіз наукової літератури та існуючих інформаційно-довідкових систем, системний аналіз предметної області, методи проєктування програмних систем і баз даних, принципи об’єктно-орієнтованого програмування, а також методи модульного та функціонального тестування програмного забезпечення.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНФОРМАЦІЙНО-ДОВІДКОВИХ СИСТЕМ ТА СЕРВІСІВ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Поняття та особливості інформаційно-довідкових систем

Інформаційно-довідкові системи належать до класу інформаційних систем, призначених для забезпечення користувача структурованою довідковою інформацією у відповідь на запити. У наукових джерелах вони визначаються як поєднання бази даних, пошукових механізмів та засобів подання результатів, що дозволяє швидко отримувати релевантну інформацію без необхідності аналізу всього масиву даних [1]. У практичному контексті це означає, що подібні системи формують середовище швидкого доступу до знань, де ключовим є не зберігання інформації, а її оперативне отримання за потреби.

У межах загальної теорії інформаційних систем вони трактуються як організовані сукупності взаємопов’язаних компонентів, що забезпечують збирання, зберігання, оброблення та подання інформації. Водночас інформаційно-довідкові системи відрізняються тим, що їх основна функція полягає в інформаційному обслуговуванні користувача через механізм запит-відповідь, а не в управлінні процесами або виконанні складних обчислень [1]. Це підкреслює їх прикладний характер як інструментів підтримки інформаційних потреб людини, а не автономних систем прийняття рішень.

Концептуально такі системи базуються на структурованих інформаційних ресурсах, що об’єднані спільною предметною областю. До них відносять довідники, енциклопедії, каталоги та бібліографічні записи, які в цифровому вигляді інтегруються в єдину систему з механізмами пошуку та навігації [2; 3]. Така організація дозволяє перетворити розрізнені джерела знань на цілісне інформаційне середовище, що значно підвищує ефективність роботи з великими обсягами даних.

Функціонування інформаційно-довідкових систем охоплює процеси накопичення інформації, її структуризації, індексування, обробки запитів, пошуку та відображення результатів. У сучасних реалізаціях додаються механізми фільтрації, сортування та уточнення результатів, що дозволяє підвищити точність і релевантність отриманої інформації [4]. У результаті користувач отримує не просто набір даних, а впорядкований інформаційний результат, який відповідає конкретному запиту і може бути одразу використаний у практичній діяльності.

Однією з ключових характеристик таких систем є структурована організація даних, яка реалізується через класифікатори, метадані, теги та тематичні категорії. Це забезпечує ефективну навігацію в інформаційному просторі та скорочує час пошуку необхідних відомостей [5]. У реальних умовах це дозволяє користувачеві працювати з великими масивами інформації без необхідності розуміти їх внутрішню структуру, що суттєво підвищує зручність використання системи.

Динамічність інформаційного наповнення проявляється у регулярному оновленні даних відповідно до змін предметної області. Це особливо важливо для сфер із високою швидкістю оновлення інформації, таких як інформаційні технології, наука чи технічні дисципліни. У практичному аспекті це дозволяє підтримувати актуальність даних, що безпосередньо впливає на якість прийняття рішень користувачем.

Орієнтація на користувача реалізується через спрощені механізми доступу до інформації, зокрема пошук за ключовими словами, темами або природними запитами. Інтерфейс і логіка системи формуються таким чином, щоб мінімізувати вимоги до попереднього знання структури даних [1]. Це робить інформаційно-довідкові системи доступними для широкого кола користувачів незалежно від рівня їх технічної підготовки.

Важливим аспектом є забезпечення достовірності інформації, що досягається шляхом використання перевірених або структурованих джерел, які проходять попередню обробку та верифікацію. У практичному застосуванні це

формує довіру до системи як до джерела інформації, яке може використовуватися в освітній, науковій та професійній діяльності без додаткової перевірки базових даних.

Інформаційно-довідкові системи також характеризуються можливістю інтеграції різнорідних типів даних, включаючи текстові, графічні та структуровані елементи. Взаємозв'язки між записами реалізуються через внутрішні посилання та тематичні зв'язки, що формує єдине інформаційне середовище [6]. У практичному вимірі це дозволяє користувачу не лише отримувати окремі дані, але й бачити їх у взаємозв'язку з іншими інформаційними об'єктами.

У структурі інформаційних систем цей клас займає проміжне положення між базами даних та аналітичними системами. Його основне призначення полягає у наданні структурованої інформації для подальшого самостійного аналізу користувачем без автоматизованого прийняття рішень [1]. Це визначає їх роль як інструменту інформаційної підтримки, а не як системи впливу на процес ухвалення рішень.

Отже, інформаційно-довідкові системи застосовуються як цифрові засоби доступу до знань у різних предметних областях, забезпечуючи централізоване зберігання інформації, її систематизацію та швидке відтворення за запитом користувача. У сучасних умовах це дозволяє значно підвищити ефективність роботи з інформацією та скоротити час пошуку необхідних даних у професійній діяльності.

1.2 Аналіз сучасних інформаційно-довідкових систем

Стрімкий розвиток технологій штучного інтелекту суттєво трансформував підходи до побудови ІДС. У 2025 році штучний інтелект переживає кардинальну трансформацію – від простого інструменту автоматизації до автономних систем, здатних самостійно приймати рішення та діяти без безпосереднього втручання

людини [7]. Це безпосередньо вплинуло на ІДС: сучасні системи вже не лише відповідають на запити з попередньо визначеного масиву даних, а й генерують відповіді на основі великих мовних моделей (LLM), здійснюють пошук у реальному часі та узагальнюють інформацію з різнорідних джерел.

За функціональним призначенням та архітектурними підходами сучасні ІДС на основі штучного інтелекту можна поділити на такі основні категорії [8; 9]:

– конверсаційні (діалогові) системи – орієнтовані на природномовний діалог із користувачем, генерацію зв’язних відповідей та підтримку контексту розмови (ChatGPT, Claude, Google Gemini);
– пошуково-довідкові системи з ШІ – системи, що поєднують традиційний веб-пошук із генеративними моделями для синтезу структурованих відповідей із посиланнями на джерела (Perplexity AI, Google AI Overviews);
– вбудовані корпоративні асистенти – системи, інтегровані з існуючими офісними або хмарними платформами (Microsoft 365 Copilot, Google Workspace Gemini);
– спеціалізовані довідкові системи – системи для конкретних предметних областей (медицина, право, освіта), побудовані на основі дообчислених (fine-tuned) моделей або методу RAG (Retrieval-Augmented Generation).

ChatGPT є однією з найбільш поширених ІДС на основі великих мовних моделей, розробленою компанією OpenAI. Починаючи з квітня 2025 року, OpenAI замінила модель GPT-4 на GPT-4o – більш швидкий мультимодальний наступник, що перевершує попередника в написанні текстів, програмуванні, STEM-завданнях та загальних знаннях [10]. Ключові характеристики системи: орієнтація на діалог, широкий спектр застосувань (генерація тексту, підтримка клієнтів, пошук інформації, навчальна підтримка, програмування та творчі задачі), а також інтеграція функцій реального часу пошуку з травня 2025 року [10]. ChatGPT, на відміну від пошуково-орієнтованих систем, насамперед генерує зв’язні, контекстно-насичені відповіді на основі навчальних даних [11].

Perplexity AI є системою, спеціально спроектованою для відповіді на запитання шляхом пошуку у веб-просторі та надання відповідей із посиланнями на першоджерела [11]. Архітектура платформи базується на реальному часовому отриманні даних: кожна відповідь містить інтегровані покликання на джерела, що дозволяє верифікувати інформацію. Perplexity AI позиціонується як потенційна альтернатива традиційним пошуковим системам завдяки можливості надання стислих, верифікованих відповідей [12]. У червні 2025 року платформа завершила раунд фінансування на 500 мільйонів доларів із оцінкою в 14 мільярдів доларів та запустила функцію Deep Research, що виконує десятки пошукових запитів і синтезує вичерпні звіти з комплексних тематик [13]. Система використовує спеціалізовані методи контекстного вилучення інформації, що забезпечує актуальні та верифіковані відповіді, особливо цінні для академічних і аналітичних задач [14].

Google Gemini є сімейством великих мовних моделей, розроблених Google/DeepMind, що прийшли на заміну попереднім системам PaLM2/Bard у 2024 році. У листопаді 2025 року компанія представила серію Gemini 3, яка вирізняється потужними можливостями логічного виведення, тривалими ланцюжками міркувань та розширеною підтримкою програмування [15]. Підтримка контекстного вікна до 1 мільйона токенів у версії для розробників дозволяє системі аналізувати до 1 500 сторінок тексту або понад 30 000 рядків коду в рамках одного запиту [16]. Gemini тісно інтегрований з екосистемою Google Workspace, що робить його особливо затребуваним у корпоративному середовищі. Станом на грудень 2025 року платформа налічує близько 650 мільйонів активних користувачів щомісяця [15].

Microsoft Copilot є ІДС, вбудованою в екосистему Microsoft 365 і побудованою на базі моделей OpenAI GPT-4o, що функціонують в інфраструктурі Azure OpenAI. Система інтегрована з Word, Excel, PowerPoint, Outlook, Teams, OneNote та Loop, що робить її ключовим корпоративним інструментом підвищення продуктивності [17]. На відміну від ChatGPT, Copilot доповнений підключенням до Microsoft Graph API, що дозволяє враховувати

контекст із файлів користувача, календаря та електронної пошти [15]. До 2025 року Microsoft звітувала про впровадження генеративного ШІ в 85% компаній зі списку Fortune 500 через свої платформи [17]. Система відрізняється розвинутою функціональністю корпоративної безпеки та управління даними, що особливо важливо для регульованих галузей [16].

Claude є конверсаційною ІДС від компанії Anthropic, що вирізняється особливою спрямованістю на стратегічне планування, аналіз та розширені аналітичні розмови. Система демонструє найвищу продуктивність у задачах, що потребують глибокого розуміння контексту та структурованого викладу думок. Серед усіх систем-аналогів Claude відрізняється кращими показниками емоційного інтелекту в ході розширених діалогів [13]. Архітектурно система побудована з акцентом на безпечність і корисність, що обумовлює її популярність для задач із чутливою інформацією.

Порівняльний аналіз провідних ІДС виявляє суттєві відмінності в архітектурних підходах та цільовій аудиторії. ChatGPT і Claude орієнтовані на глибокі конверсаційні взаємодії та генерацію різноманітного контенту, тоді як Perplexity AI спеціалізується на точному, верифікованому пошуку інформації та цитуванні джерел [12]. Google Gemini займає проміжну позицію, поєднуючи розширені можливості логічного виведення з глибокою інтеграцією у хмарну екосистему [15]. Microsoft Copilot, у свою чергу, орієнтований переважно на корпоративних користувачів Microsoft 365 [17].

Ключовою архітектурною відмінністю між системами є підхід до отримання актуальних даних. Perplexity AI від початку побудована навколо веб-перегляду та цитування джерел, тоді як ChatGPT спочатку мав фіксовану базу знань без атрибуції джерел. Проте станом на 2024–2025 роки ChatGPT інтегрував функції пошуку і почав надавати посилання на джерела, що суттєво скоротило цей розрив [12]. Спільною тенденцією для всіх систем є еволюція від простих запитально-відповідних інтерфейсів до мультимодальних асистентів, здатних обробляти текст, зображення та документи в рамках єдиного запиту.

Аналіз сучасного стану ринку ІДС дозволяє виокремити наступні ключові тенденції. По-перше, спостерігається стрімке зростання ринку генеративного ШІ: обсяг інвестицій склав 33,9 мільярда доларів у 2024 році, що на 18,7% більше, ніж у 2023 році [7]. По-друге, відбувається широке корпоративне впровадження: 83% компаній називають ШІ-рішення топ-пріоритетом у своїх бізнес-планах [7]. По-третє, посилюється регуляторний тиск: у серпні 2024 року набув чинності Акт про штучний інтелект ЄС (AI Act), що встановлює комплексні правові норми для розробки та використання ШІ-систем [7].

Серед технічних тенденцій варто виокремити: збільшення контекстних вікон моделей (до 1–2 мільйонів токенів), інтеграцію реального часового пошуку безпосередньо в ІДС, розвиток мультиагентних архітектур та широке впровадження методу RAG (Retrieval-Augmented Generation) для забезпечення актуальності та достовірності відповідей. Ці тенденції безпосередньо впливають на вимоги до проектування нових ІДС, зокрема системи «Сервіси штучного інтелекту», яка розглядається у даній роботі.

Таким чином, аналіз сучасних ІДС виявив, що на ринку відсутня спеціалізована україномовна довідкова система, яка б систематизувала інформацію саме про сервіси штучного інтелекту, їх характеристики, можливості та сфери застосування. Існуючі рішення (ChatGPT, Perplexity, Gemini) є загальними інструментами і не вирішують задачу структурованого каталогізування та порівняльного аналізу ШІ-сервісів. Це підтверджує актуальність проектування спеціалізованої ІДС «Сервіси штучного інтелекту» та обумовлює вимоги до її функціональних можливостей.

1.3 Визначення вимог до інформаційно-довідкової системи «Сервіси штучного інтелекту»

Визначення вимог до інформаційно-довідкової системи є ключовим етапом проєктування, оскільки саме на цьому етапі формуються основні функції

майбутнього програмного продукту, його обмеження та принципи взаємодії з користувачем. Для системи «Сервіси штучного інтелекту» вимоги визначаються специфікою предметної області, яка передбачає роботу з великою кількістю різнорідних AI-інструментів, їх класифікацію, опис характеристик і забезпечення зручного пошуку.

Функціональні вимоги (табл. 1.1) описують, які саме завдання повинна виконувати система, тоді як нефункціональні вимоги визначають якість її роботи, продуктивність, безпеку та зручність використання.

Таблиця 1.1 – Функціональні вимоги

№	Функціональна вимога	Опис
1	Реєстрація та авторизація користувачів	Система повинна підтримувати створення облікових записів та вхід у систему
2	Перегляд каталогу AI-сервісів	Користувач має можливість переглядати список сервісів штучного інтелекту
3	Пошук сервісів	Реалізація пошуку за назвою, категорією або ключовими словами
4	Фільтрація та сортування	Можливість фільтрувати сервіси за типом, призначенням або популярністю
5	Перегляд детальної інформації	Відображення опису сервісу, функцій, посилань та характеристик
6	Категоризація сервісів	Розподіл AI-сервісів за групами (текст, зображення, код, аналітика тощо)
7	Додавання нових сервісів (адмін)	Адміністратор може додавати нові записи до бази даних
8	Редагування та видалення даних (адмін)	Можливість оновлення або видалення інформації про сервіси
9	Відображення актуальних даних	Система повинна забезпечувати актуальність інформації

Пункти 1–2 описують базову взаємодію з системою: реєстрація та авторизація забезпечують доступ користувачів і розмежування ролей, а перегляд каталогу дозволяє отримувати структурований список AI-сервісів.

Пункти 3–5 стосуються роботи з інформацією: пошук забезпечує швидке знаходження сервісів, фільтрація та сортування – уточнення вибірки за параметрами, а перегляд детальної інформації дає повний опис кожного сервісу.

Пункт 6 визначає логічну організацію даних через категоризацію сервісів за типами (текст, зображення, код тощо), що спрощує навігацію в системі.

Пункти 7–8 описують адміністративні функції: додавання нових сервісів, а також редагування і видалення записів для підтримки актуальності та коректності бази даних.

Пункт 9 підкреслює необхідність постійного оновлення інформації, що є важливим через швидкий розвиток сервісів штучного інтелекту.

Нефункціональні вимоги (табл. 1.2) визначають якість роботи системи та умови її експлуатації. Вони є критично важливими для інформаційно-довідкових систем, оскільки такі системи орієнтовані на швидкий доступ до даних і повинні забезпечувати стабільну роботу навіть при зростанні обсягу інформації.

Таблиця 1.2 – Нефукціональні вимоги

№	Нефункціональна вимога	Опис
1	Швидкодія системи	Час відгуку на запит користувача не повинен перевищувати 2–3 секунди
2	Зручність інтерфейсу	Інтерфейс повинен бути інтуїтивно зрозумілим і адаптованим для користувача
3	Масштабованість	Система повинна підтримувати розширення бази даних без втрати продуктивності
4	Надійність	Забезпечення стабільної роботи системи без критичних збоїв
5	Безпека даних	Захист користувацьких даних та обмеження доступу до адміністративних функцій
6	Адаптивність	Коректне відображення на різних пристроях (ПК, планшет, смартфон)
7	Підтримуваність	Можливість подальшого оновлення та розширення функціоналу
8	Актуальність даних	Інформація в системі повинна регулярно оновлюватися

Швидкодія системи передбачає мінімальний час відгуку на запити користувача (до 2–3 секунд), що забезпечує комфортну взаємодію. Зручність інтерфейсу означає інтуїтивну навігацію та адаптацію під потреби користувача.

Масштабованість визначає здатність системи розширювати базу даних без втрати продуктивності, а надійність гарантує стабільну роботу без критичних збоїв.

Безпека даних забезпечує захист інформації користувачів і розмежування доступу до адміністративних функцій. Адаптивність передбачає коректне відображення системи на різних пристроях.

Підтримуваність означає можливість подальшого розвитку та оновлення функціоналу, а актуальність даних – регулярне оновлення інформації в системі відповідно до змін у сфері ШІ.

У результаті визначено основні функціональні та нефункціональні вимоги до інформаційно-довідкової системи «Сервіси штучного інтелекту». Функціональні вимоги охоплюють базові сценарії роботи користувача, включаючи пошук, перегляд, фільтрацію та управління інформацією про AI-сервіси, а також адміністративні можливості наповнення та редагування бази даних. Нефункціональні вимоги визначають якісні характеристики системи, зокрема швидкодію, зручність інтерфейсу, безпеку, масштабованість та актуальність даних. Сформовані вимоги створюють основу для подальшого проєктування архітектури та реалізації програмного продукту.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ «СЕРВІСИ ШТУЧНОГО ІНТЕЛЕКТУ»

2.1 Аналіз предметної області та проєктування структури системи

Предметною областю дослідження є сервіси штучного інтелекту, що являють собою програмні продукти та вебплатформи, призначені для автоматизації інтелектуальної діяльності людини. До таких сервісів належать системи генерації тексту, створення та редагування зображень, обробки аудіо й відео, написання програмного коду, аналізу даних, а також інструменти для підвищення особистої та корпоративної продуктивності.

Протягом останніх років спостерігається стрімке зростання кількості AI-сервісів, що супроводжується розширенням їх функціональних можливостей. Водночас інформація про такі сервіси є розпорошеною між офіційними сайтами розробників, тематичними ресурсами, блогами та форумами. Це ускладнює процес пошуку необхідного інструменту, його оцінювання та порівняння з альтернативними рішеннями.

У зв’язку з цим виникає потреба у створенні спеціалізованої інформаційно-довідкової системи, яка забезпечуватиме централізоване зберігання відомостей про сервіси штучного інтелекту, їх систематизацію, пошук та надання користувачеві структурованої інформації у зручному форматі.

Для кращого розуміння предметної області було визначено її основні об’єкти та їх характеристики, які наведено в таблиці 2.1.

У таблиці 2.1 наведено основні інформаційні об’єкти, які формують структуру предметної області. Центральним елементом є сервіс штучного інтелекту, навколо якого організовано інші сутності. Категорії забезпечують класифікацію сервісів, користувачі взаємодіють із системою та залишають відгуки, а сесії використовуються для підтримки механізмів автентифікації.

Виділення зазначених об’єктів є необхідним для подальшого проєктування бази даних та визначення взаємозв’язків між її компонентами.

Таблиця 2.1 – Об’єкти предметної області

Об’єкт	**Опис**	**Основні атрибути**
Сервіс ШІ	Конкретний програмний продукт або платформа	Назва, опис, посилання, модель цін, розробник, категорія
Категорія	Тематична група сервісів	Назва, ідентифікатор (slug), опис
Користувач	Зареєстрована особа, що користується системою	Ім’я, e-mail, роль, дата реєстрації
Відгук	Оцінка та коментар користувача про сервіс	Оцінка (1–5), коментар, дата
Сесія	Активний сеанс автентифікованого користувача	Ідентифікатор, користувач, дата створення

У системі передбачено три основні ролі: гість, користувач та адміністратор. Кожна з них має власний набір функціональних можливостей, що визначають рівень взаємодії із системою.

Діаграму прецедентів наведено на рисунку 2.1.

На рисунку 2.1 представлено основні сценарії використання системи залежно від ролі користувача. Гість може переглядати каталог сервісів, здійснювати пошук та фільтрацію інформації, а також переглядати детальні відомості про окремі сервіси. Після реєстрації користувач отримує можливість залишати відгуки та оцінки. Адміністратор має розширені повноваження, що включають управління сервісами, категоріями, користувачами та перегляд статистичної інформації. Представлення функціоналу у вигляді діаграми прецедентів дозволяє наочно відобразити взаємодію між акторами та системою.

Проєктована система реалізується на основі триланкової клієнт-серверної архітектури, що передбачає поділ програмного забезпечення на рівень представлення, серверний рівень та рівень даних.

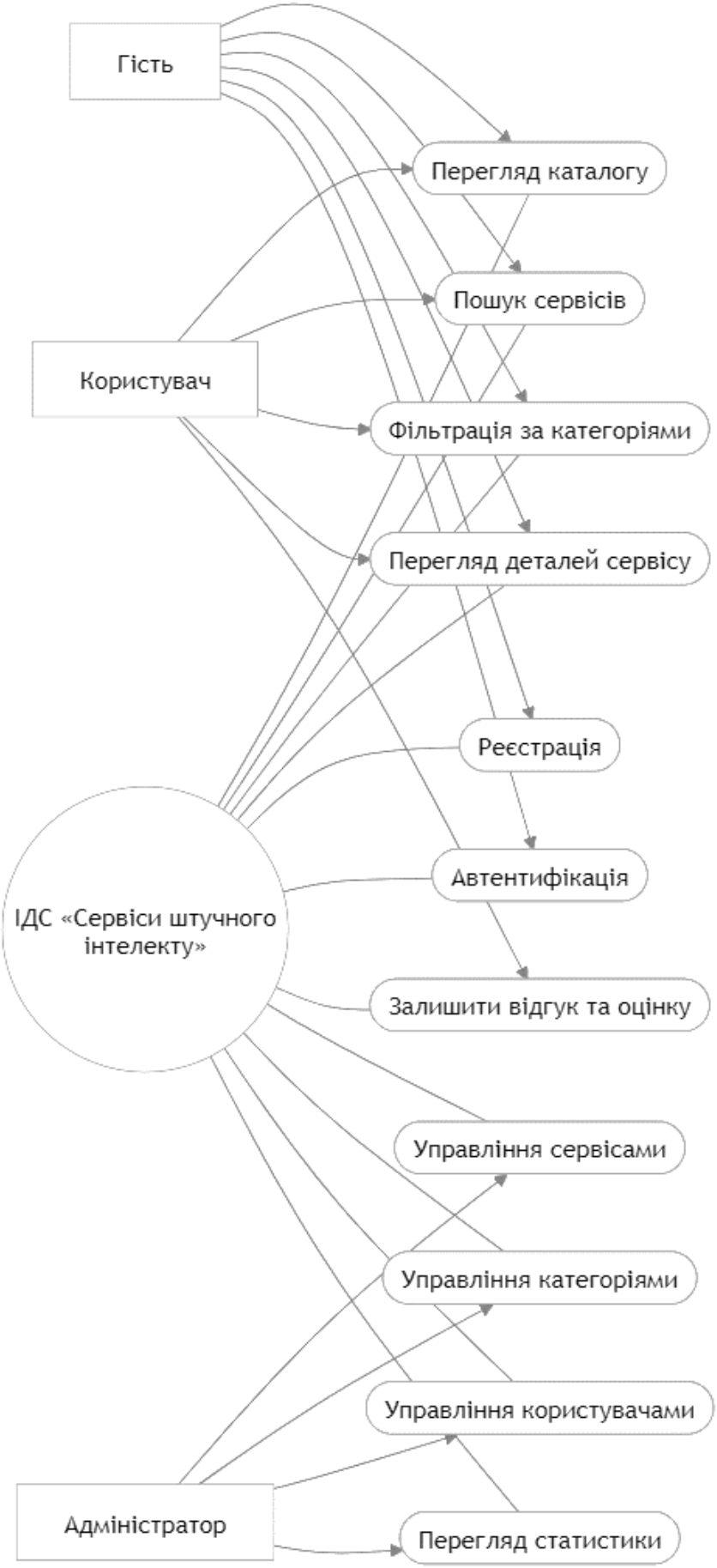


Рисунок 2.1 – Діаграма прецедентів системи

Загальну структуру системи наведено на рисунку 2.2.

На рисунку 2.2 зображено взаємодію основних компонентів програмного продукту. Клієнтський рівень відповідає за відображення інтерфейсу користувача та обробку взаємодії з ним. Серверний рівень, забезпечує виконання бізнес-логіки, обробку запитів, автентифікацію користувачів і взаємодію з базою даних. Рівень даних представлений СКБД SQLite, яка використовується для зберігання інформації про сервіси, категорії, користувачів, відгуки та сесії. Використання триланкової архітектури сприяє підвищенню модульності системи, спрощує її підтримку та створює можливості для подальшого масштабування функціоналу.

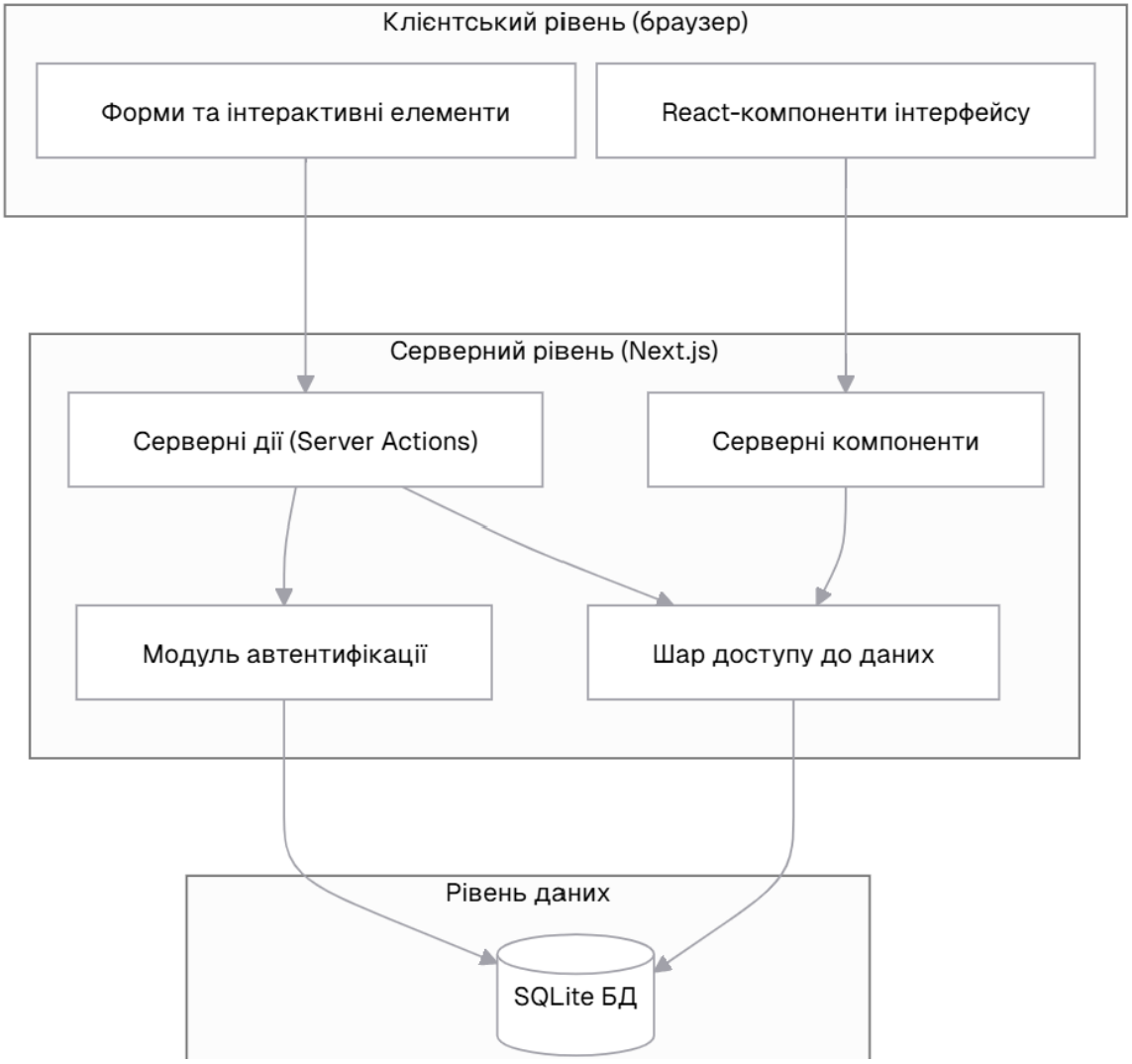


Рисунок 2.2 – Триланкова архітектура системи

У результаті аналізу предметної області було визначено основні об’єкти інформаційно-довідкової системи «Сервіси штучного інтелекту», встановлено взаємозв’язки між ними та сформовано вимоги до функціонування системи. Проєктування структури системи дозволило виділити ключові ролі користувачів, окреслити основні сценарії взаємодії та обґрунтувати вибір триланкової клієнт-серверної архітектури. Отримані результати створюють основу для подальшого проєктування бази даних, реалізації функціональних модулів і розробки програмного продукту.

2.2 Проєктування бази даних і функціональних модулів

Проєктування бази даних інформаційно-довідкової системи «Сервіси штучного інтелекту» виконано на основі реляційного підходу, що дозволяє забезпечити структуроване зберігання інформації та встановлення чітких зв’язків між сутностями предметної області. Такий підхід є доцільним для систем довідкового типу, оскільки передбачає ефективну організацію даних і підтримку їх цілісності при виконанні операцій пошуку, додавання та оновлення.

У межах проєктування виділено п’ять основних сутностей: users, categories, services, reviews, sessions, кожна з яких відповідає окремому об’єкту предметної області та виконує визначену функцію в системі. Логічну структуру взаємодії між ними подано на ER-діаграмі (рисунок 2.3).

ER-діаграма відображає ключові зв’язки між сутностями системи. Центральною сутністю є services, оскільки саме вона містить основну інформацію про сервіси штучного інтелекту. Вона пов’язана з категоріями, користувачами та відгуками, що забезпечує багаторівневу структуру даних і можливість їх логічного групування. Сутність users забезпечує зберігання даних про користувачів і підтримує механізм автентифікації, тоді як sessions використовується для контролю активних сеансів роботи в системі.

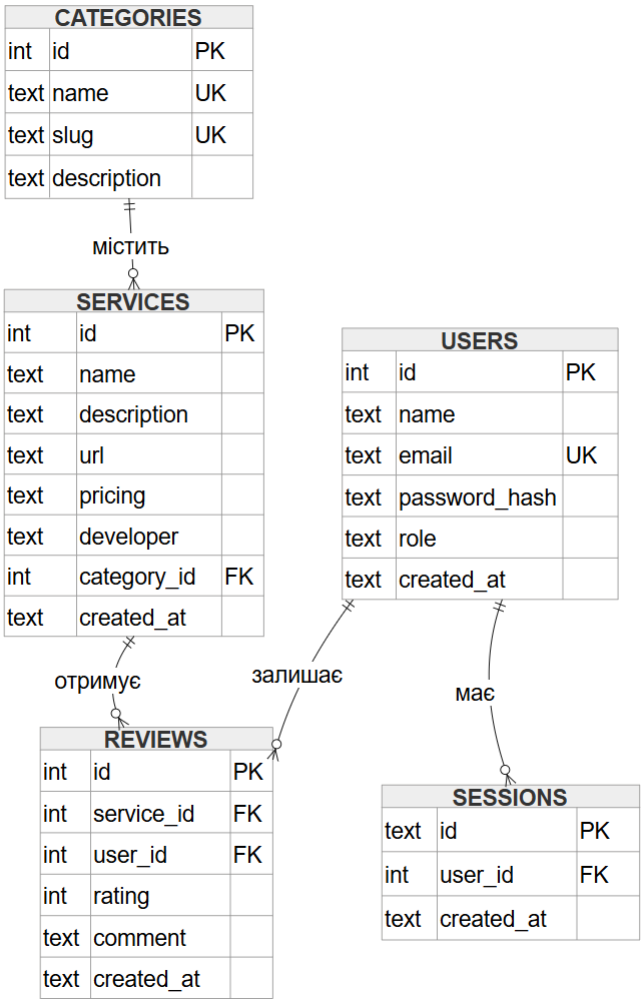


Рисунок 2.3 – ER-діаграма бази даних

Між сутностями визначено зв’язки різних типів (табл. 2.2), які регулюють цілісність даних і правила їх оброблення.

Таблиця 2.2 – Зв’язки між сутностями

Зв’язок	Тип	Правило цілісності
Категорія – Сервіси	1:N	ON DELETE SET NULL
Сервіс – Відгуки	1:N	ON DELETE CASCADE
Користувач – Відгуки	1:N	ON DELETE CASCADE
Користувач – Сесії	1:N	ON DELETE CASCADE
Сервіс + Користувач – Відгук	унікальний	UNIQUE(service_id, user_id)

У таблиці 2.2 подано логіку взаємозв’язків між сутностями бази даних. Зв’язок «один-до-багатьох» між категоріями та сервісами дозволяє групувати сервіси за тематичними напрямами, при цьому кожна категорія може містити

декілька сервісів. Каскадне видалення у зв’язках із відгуками та сесіями забезпечує автоматичне очищення залежних записів, що запобігає накопиченню застарілих або неактуальних даних. Обмеження унікальності service_id та user_id гарантує, що один користувач може залишити лише один відгук для конкретного сервісу, що підвищує коректність оцінювання.

Фізичне проєктування бази даних передбачає визначення конкретної структури таблиць, типів даних та обмежень, які забезпечують коректне зберігання та оброблення інформації в системі.

Таблиця users (табл. 2.3) призначена для зберігання інформації про зареєстрованих користувачів системи.

Таблиця 2.3 – Структура таблиці users

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор
name	TEXT	NOT NULL	Ім’я користувача
email	TEXT	NOT NULL, UNIQUE	Електронна пошта
password_hash	TEXT	NOT NULL	Хеш пароля
role	TEXT	CHECK IN (‘user’,’admin’)	Роль користувача
created_at	TEXT	DEFAULT datetime(‘now’)	Дата реєстрації

Поля таблиці users призначені для зберігання облікових даних користувачів системи. Ідентифікатор id використовується для унікальної ідентифікації записів у базі даних. Поля name та email містять базову інформацію про користувача, при цьому email виконує функцію логіна та має обмеження унікальності. Поле password_hash забезпечує безпечне зберігання пароля у зашифрованому вигляді. Атрибут role визначає рівень доступу користувача в системі, а created_at фіксує дату створення облікового запису.

Таблиця categories (табл. 2.4) забезпечує класифікацію сервісів штучного інтелекту за напрямами використання.

Поля таблиці categories використовуються для групування сервісів за тематичними напрямами. Ідентифікатор id забезпечує унікальність записів, тоді як name визначає назву категорії.

Поле slug застосовується для формування зручних URL-адрес і внутрішньої ідентифікації категорій. Атрибут description містить додаткову текстову інформацію про призначення категорії.

Таблиця 2.4 – Структура таблиці categories

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Ідентифікатор категорії
name	TEXT	NOT NULL, UNIQUE	Назва категорії
slug	TEXT	NOT NULL, UNIQUE	Текстовий ідентифікатор
description	TEXT	–	Опис категорії

Основною таблицею бази даних є таблиця services (табл. 2.5), у якій накопичуються відомості про сервіси штучного інтелекту.

Таблиця 2.5 – Структура таблиці services

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Ідентифікатор сервісу
name	TEXT	NOT NULL	Назва сервісу
description	TEXT	NOT NULL	Опис сервісу
url	TEXT	–	Посилання на сервіс
pricing	TEXT	–	Модель оплати
developer	TEXT	–	Розробник
category_id	INTEGER	FK – categories(id)	Категорія
created_at	TEXT	DEFAULT datetime(‘now’)	Дата додавання

Поля таблиці services містять основну інформацію про сервіси штучного інтелекту. Атрибути name та description відображають базові характеристики сервісу. Поле url дозволяє здійснювати перехід до офіційного ресурсу сервісу, а pricing та developer містять додаткові відомості про умови використання та розробника. Зовнішній ключ category_id забезпечує зв’язок із таблицею категорій, що дозволяє реалізувати логічну класифікацію сервісів.

Таблиця reviews (табл. 2.6) використовується для організації механізму зворотного зв’язку з користувачами.

Поля таблиці reviews призначені для реалізації функції зворотного зв’язку користувачів. Атрибут rating дозволяє оцінювати сервіси за п’ятибальною шкалою, тоді як comment використовується для текстових відгуків. Зв’язки через

service_id та user_id забезпечують прив’язку відгуку до конкретного сервісу та його автора, що дозволяє аналізувати користувацьку активність і якість сервісів.

Таблиця 2.6 – Структура таблиці reviews

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Ідентифікатор відгуку
service_id	INTEGER	FK – services(id)	Сервіс
user_id	INTEGER	FK – users(id)	Користувач
rating	INTEGER	CHECK 1–5	Оцінка
comment	TEXT	–	Коментар
created_at	TEXT	DEFAULT datetime(‘now’)	Дата створення

Також систему декомпозовано на набір функціональних модулів (рис. 2.4).

Кожен з модулів відповідає за окрему підсистему та реалізує принцип єдиної відповідальності. Такий підхід дозволяє спростити розробку, тестування та подальше масштабування системи.

У таблиці 2.7 подано детальніше призначення кожного програмного модуля та його реалізацію в структурі проєкту.

Таблиця 2.7 – Функціональні модулі та їх реалізація

Модуль	Файл / частина системи	Призначення
Модуль автентифікації	lib/auth.ts	Реєстрація користувачів, вхід у систему, управління сесіями та ролями
Шар доступу до даних	lib/queries.ts	Виконання запитів до бази даних і отримання структурованих даних
Серверні дії	lib/actions.ts	Обробка форм і реалізація CRUD-операцій
База даних	lib/db.ts	Підключення до SQLite та ініціалізація схеми
Модуль каталогу	app/services/	Відображення сервісів, пошук, фільтрація та перегляд деталей
Адмін-панель	app/admin/	Управління сервісами, категоріями та користувачами

Таблиця 2.7 містить перелік основних функціональних модулів системи та їх реалізацію у структурі проєкту. Вона відображає розподіл системи на логічно відокремлені частини, кожна з яких відповідає за окремий напрям функціональності та має власну зону відповідальності.

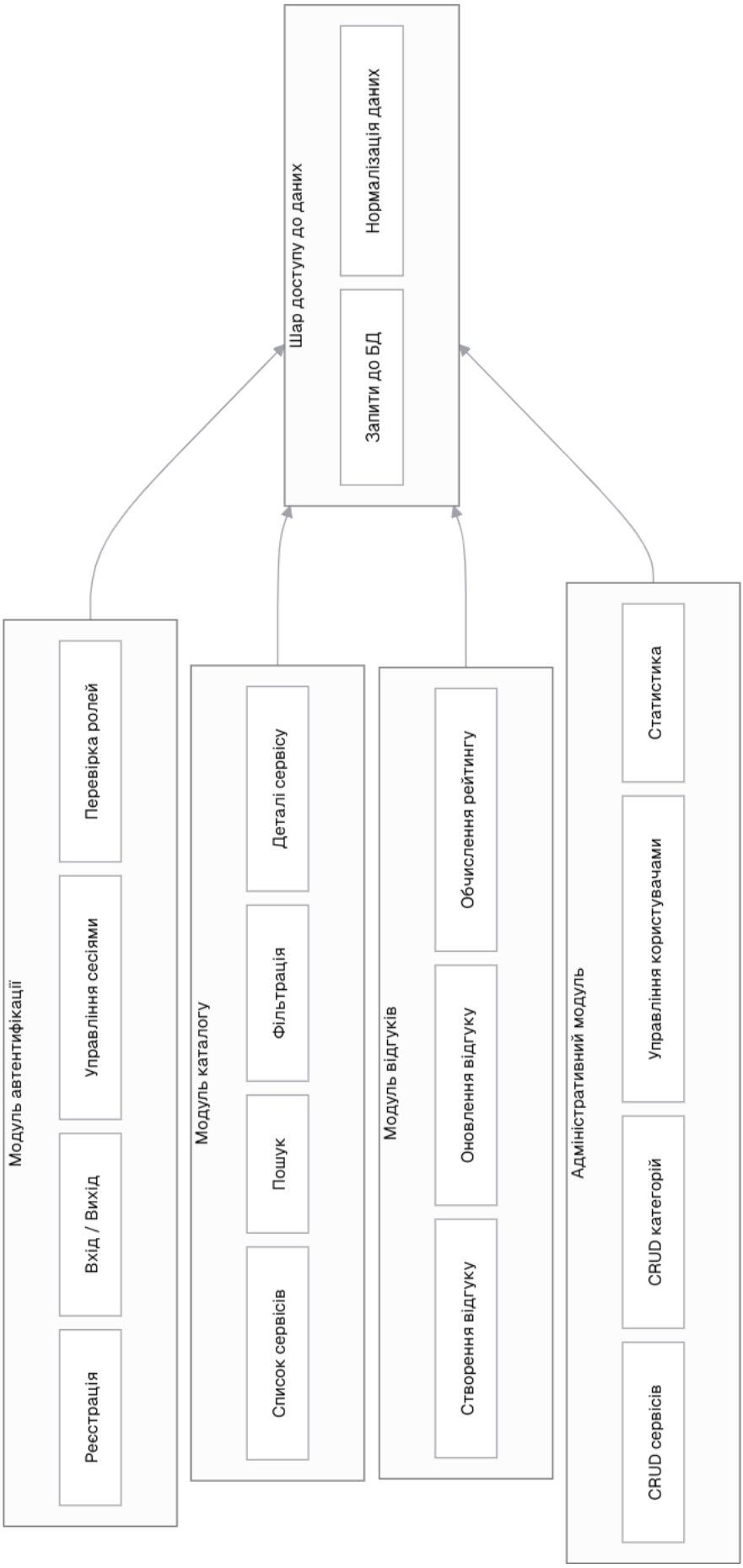


Рисунок 2.4 – Функціональна структура системи

Модуль автентифікації забезпечує всі процеси, пов’язані з реєстрацією користувачів, входом у систему, перевіркою облікових даних, створенням сесій і визначенням ролей доступу. Шар доступу до даних відповідає за взаємодію з базою даних, формування запитів та отримання структурованої інформації про сутності системи.

Серверні дії реалізують обробку даних, що надходять із форм користувача, та виконують операції створення, оновлення і видалення записів. База даних містить логіку підключення до SQLite та ініціалізацію її структури, забезпечуючи стабільне зберігання та доступ до інформації.

Модуль каталогу охоплює користувацьку частину системи, що відповідає за перегляд списку сервісів, їх пошук, фільтрацію та отримання детальної інформації. Адміністративний модуль призначений для управління контентом системи, включаючи сервіси, категорії та користувачів, а також контроль і редагування даних.

Алгоритм автентифікації забезпечує захищений доступ до системи та розмежування прав користувачів залежно від їх ролей. Процес входу та перевірки доступу включає кілька послідовних етапів, які реалізуються на стороні сервера.

Процес автентифікації побудований на перевірці облікових даних через хешування пароля (bcrypt), після чого створюється сесія користувача. Подальший доступ до сторінок визначається роллю, яка зберігається в базі даних і перевіряється при кожному запиті. Такий підхід забезпечує захищений доступ до адміністративної частини системи та унеможливлює несанкціоноване використання функцій управління.

Отже виконано проєктування бази даних інформаційно-довідкової системи «Сервіси штучного інтелекту» та визначено її основні функціональні модулі. Побудовано концептуальну та логічну модель даних, що включає ключові сутності системи та зв’язки між ними, які забезпечують цілісність і узгодженість інформації.

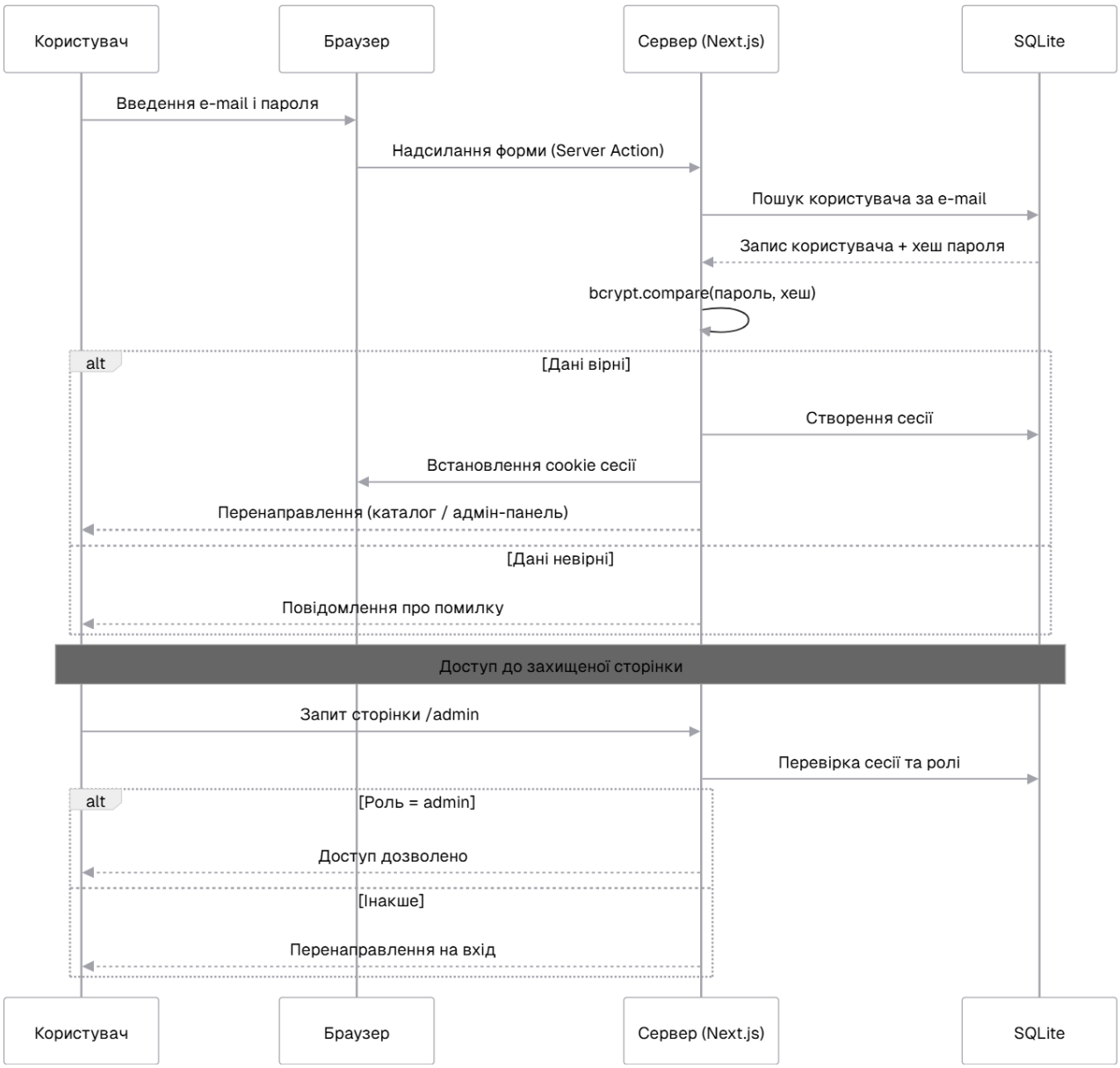


Рисунок 2.5 – Діаграма послідовності автентифікації та контролю доступу

Також визначено фізичну структуру таблиць бази даних із описом полів та обмежень, що гарантують коректне зберігання даних. Окремо виділено функціональні модулі системи, які відповідають за автентифікацію, роботу з каталогом сервісів, обробку відгуків, адміністрування та доступ до даних. Це забезпечує чіткий розподіл відповідальності між компонентами системи та створює основу для її подальшої реалізації.

2.3 Проєктування користувацького інтерфейсу та вибір технологій розробки

Проєктування користувацького інтерфейсу системи здійснено з урахуванням принципу mobile-first, що передбачає пріоритетну оптимізацію інтерфейсу для мобільних пристроїв із подальшою адаптацією під більші екрани. Такий підхід забезпечує зручність використання системи незалежно від типу пристрою користувача.

Єдина логіка навігації реалізована через спільний компонент шапки (site-header), який забезпечує доступ до основних розділів системи, зменшуючи кількість дублювання елементів інтерфейсу та підвищуючи узгодженість взаємодії.

Основні сторінки системи включають головну сторінку, каталог сервісів, сторінку деталей сервісу, а також сторінки автентифікації та адміністративну панель. Перехід між ними реалізовано через маршрутизацію Next.js, що дозволяє будувати багатосторінковий застосунок із динамічним рендерингом.

Логіку переходів між сторінками подано на рисунку 2.6.

Головною точкою входу є головна сторінка (/), з якої користувач може перейти до каталогу сервісів, сторінки входу, реєстрації або входу адміністратора. Каталог сервісів (/services) є центральним елементом системи, через який здійснюється перехід до детальної інформації про окремий сервіс (/services/:id).

Сторінки автентифікації забезпечують реєстрацію та вхід користувачів, після чого вони отримують доступ до розширених функцій системи, зокрема можливості взаємодії з сервісами. Адміністратор після входу переходить до адміністративної панелі (/admin), де має доступ до управління контентом і може повертатися до каталогу або перегляду сервісів.

З метою розмежування доступу користувачів визначено різні рівні прав. Гості мають доступ лише до перегляду каталогу та окремих сервісів,

зареєстровані користувачі можуть залишати відгуки, а адміністратори отримують доступ до адміністративної панелі та функцій керування даними.

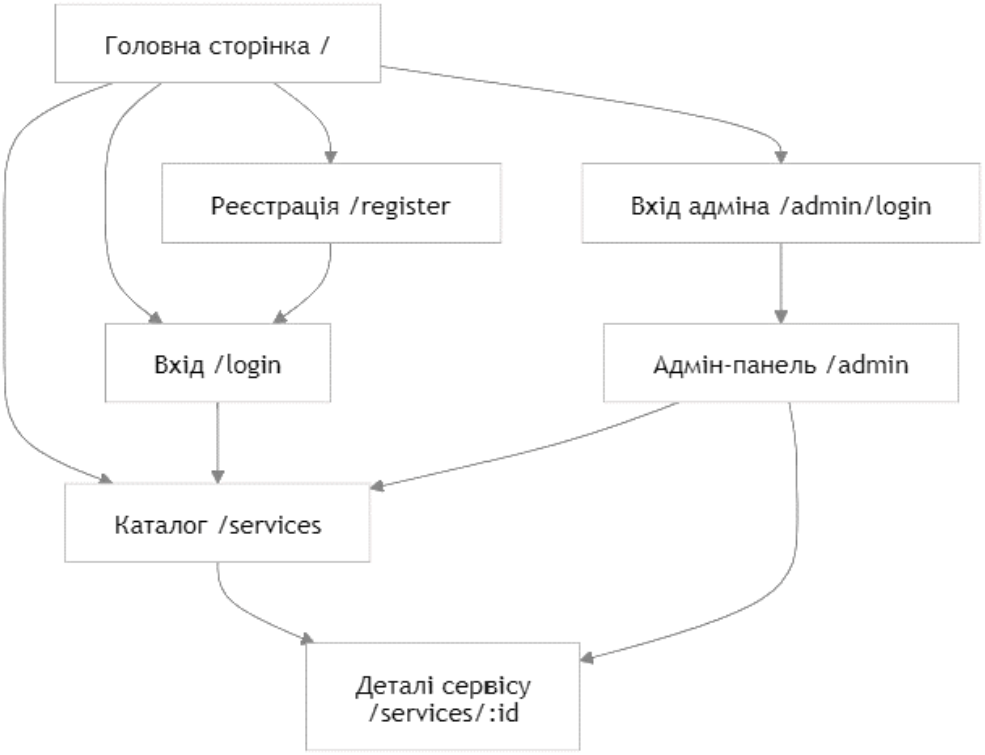


Рисунок 2.6 – Карта навігації сторінок системи

Візуальне оформлення інтерфейсу базується на принципах мінімалізму та читабельності. Для системи обрано теплу коричневу кольорову палітру, яка формує відчуття надійності, стабільності та комфортного сприйняття контенту.

Дизайн обмежено невеликою кількістю основних кольорів (3–5 відтінків), що відповідає сучасним вимогам UI/UX-дизайну та дозволяє уникнути візуального перевантаження користувача. Основний акцент зроблено на контрастності між фоновими елементами та текстом, що підвищує читабельність інтерфейсу.

Типографічна система побудована на шрифтах сімейства Geist, де Geist Sans використовується для основного тексту та інтерфейсних елементів, а Geist Mono – для службових і технічних даних. Використання двох гарнітур забезпечує візуальну структурованість без надмірної складності.

Інтерфейс системи буде реалізовано на основі компонентного підходу React, що передбачає побудову UI з незалежних, багаторазово використовуваних компонентів. Такий підхід забезпечує модульність, простоту підтримки та масштабованість системи.

Кожна сторінка складається з набору базових і функціональних компонентів, які відповідають за окремі частини інтерфейсу, зокрема відображення сервісів, фільтрацію, форми відгуків та адміністративні операції.

Структура компонентів включає як глобальні елементи (layout, навігація), так і локальні компоненти сторінок, що дозволяє розділити відповідальність між частинами системи та уникнути дублювання коду.

Ієрархію компонентів подано на рисунку 2.7.

Діаграма відображає ієрархічну компонентну структуру інтерфейсу розробленої системи та демонструє принцип побудови застосунку на основі компонентного підходу. В основі архітектури знаходиться RootLayout, який виконує роль загального контейнера інтерфейсу та забезпечує єдину структуру сторінок. Він відповідає за підключення глобальних стилів, шрифтів, темізації, а також містить базові елементи, що присутні на всіх сторінках системи, зокрема навігаційні та службові компоненти.

На наступному рівні ієрархії розташовані основні сторінки застосунку, які формують логічну структуру навігації. До них належать головна сторінка, сторінка каталогу сервісів, сторінка детального перегляду окремого сервісу, адміністративна панель, а також сторінки автентифікації користувачів (вхід і реєстрація). Кожна з цих сторінок відповідає за окремий функціональний сценарій роботи користувача та взаємодіє з серверною частиною системи для отримання або обробки даних.

Кожна сторінка, у свою чергу, складається з набору перевикористовуваних UI-компонентів, що забезпечує модульність та зручність супроводу коду. До таких компонентів належать елементи каталогу сервісів (фільтри, пошукові панелі, картки сервісів), компоненти взаємодії з користувачем (форми відгуків, елементи оцінювання, відображення рейтингів), а також адміністративні модулі, які забезпечують керування сервісами, категоріями та користувачами.

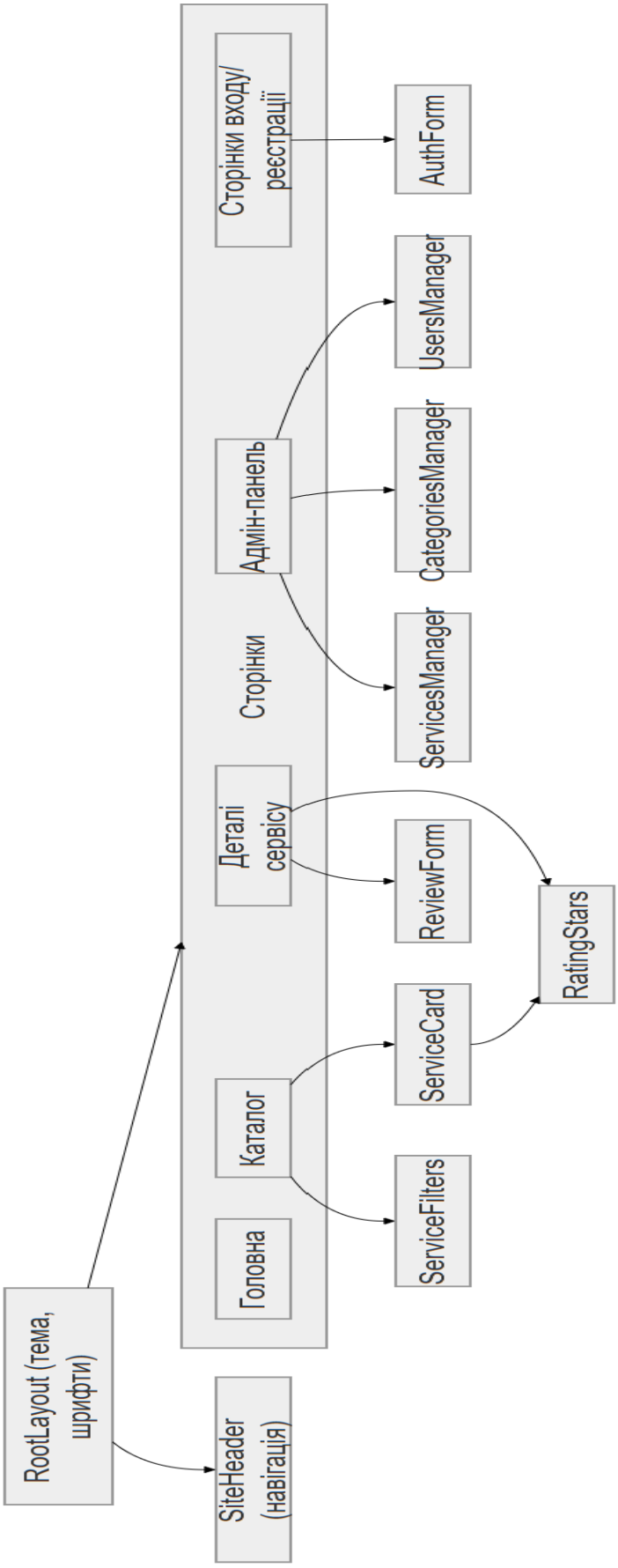


Рисунок 2.7 – Ієрархія компонентів інтерфейсу

Такий підхід до структурування інтерфейсу дозволяє забезпечити чіткий поділ відповідальності між компонентами, спрощує масштабування системи та підвищує повторне використання коду, що є важливим для підтримки та подальшого розвитку веб-застосунку.

Вибір технологічного стеку зумовлений вимогами до продуктивності, простоти розробки та відповідності сучасним стандартам веб-розробки.

Основним фреймворком обрано Next.js [18], який забезпечує серверний рендеринг, підтримку серверних дій та зручну маршрутизацію. Це дозволяє створити єдиний повноцінний веб-застосунок із мінімальними витратами на конфігурацію.

Для побудови користувацького інтерфейсу у розробленій системі використано бібліотеку React [19], яка реалізує компонентний підхід до створення веб-застосунків. Така архітектура дозволяє розділяти інтерфейс на незалежні та перевикористовувані компоненти, що суттєво спрощує супровід коду, його масштабування та подальше розширення функціональності системи. Завдяки використанню віртуального DOM забезпечується також висока продуктивність оновлення інтерфейсу, оскільки змінюються лише ті частини сторінки, які безпосередньо залежать від оновлених даних.

Додатково для підвищення надійності та передбачуваності коду використано мову TypeScript [20], яка є надбудовою над JavaScript і додає статичну типізацію. Це дозволяє виявляти значну кількість потенційних помилок ще на етапі компіляції, зокрема помилки, пов’язані з невідповідністю типів даних, неправильним використанням функцій або некоректною передачею параметрів між компонентами. У результаті підвищується загальна якість коду, його читабельність і зменшується кількість помилок під час виконання програми.

Поєднання React і TypeScript забезпечує баланс між гнучкістю розробки та структурованістю коду, що є особливо важливим для інформаційно-довідкових систем, які передбачають подальше розширення функціоналу та інтеграцію нових модулів.

Як система керування базою даних обрано SQLite [21], що забезпечує простоту розгортання без необхідності окремого серверного середовища. Для стилізації використано Tailwind CSS [22], який дозволяє швидко створювати адаптивний інтерфейс через утилітарні класи.

Додатково застосовано бібліотеки shadcn/ui [23] для готових UI-компонентів, bcryptjs для безпечного хешування паролів та Jest для модульного тестування.

Для зберігання даних обрано SQLite як легку вбудовану реляційну систему керування базами даних. Її ключовою перевагою є відсутність необхідності у встановленні окремого серверного процесу, оскільки вся база даних зберігається у вигляді одного файлу.

У порівнянні з такими системами керування базами даних, як PostgreSQL [24] та MySQL [25], SQLite [21] є значно простішою у розгортанні та налаштуванні, оскільки не потребує окремого серверного середовища, конфігурації користувачів або мережевої взаємодії. Саме це робить її особливо зручною для навчальних, курсових та невеликих прикладних проєктів, де важливо швидко отримати працююче рішення без зайвих витрат часу на адміністрування інфраструктури.

Водночас SQLite повноцінно реалізує ключові принципи реляційної моделі даних. Вона підтримує зовнішні ключі, механізми забезпечення цілісності даних, транзакційність операцій, а також базові SQL-конструкції для вибірки, фільтрації та агрегації даних. Це дозволяє використовувати її не лише як спрощену альтернативу серверним СКБД, але й як повноцінний інструмент для реалізації логіки зберігання та обробки даних у межах невеликих інформаційних систем.

Додатковою важливою перевагою є висока портативність SQLite-бази. Уся база даних зберігається у вигляді одного файлу, який може бути легко перенесений разом із програмним проєктом без необхідності додаткового налаштування або міграції даних. Це суттєво спрощує розгортання системи, її тестування на різних середовищах та демонстрацію результатів роботи.

Окрему увагу в розробленій системі приділено питанням безпеки доступу до даних. Використання параметризованих SQL-запитів через вбудований модуль node:sqlite дозволяє повністю виключити можливість SQL-ін’єкцій, оскільки користувацькі дані не підставляються безпосередньо у SQL-рядок, а передаються як параметри. Це значно підвищує рівень захищеності системи та відповідає базовим вимогам інформаційної безпеки сучасних веб-застосунків.

Тобто вибір SQLite у межах даної кваліфікаційної роботи є обґрунтованим компромісом між простотою реалізації, функціональністю та достатнім рівнем надійності для реалізації інформаційно-довідкової системи «Сервіси штучного інтелекту».

Отже, здійснено проєктування користувацького інтерфейсу та обґрунтовано вибір технологій розробки системи. Визначено структуру навігації, яка базується на принципах mobile-first та забезпечує зручний доступ до основних функцій через єдину систему маршрутизації.

Також розроблено візуальний стиль інтерфейсу з використанням обмеженої кольорової палітри та сучасної типографіки, що сприяє зручності сприйняття інформації. Окремо визначено компонентну структуру інтерфейсу, яка забезпечує модульність і повторне використання елементів.

Крім того, обґрунтовано вибір технологічного стеку, який включає Next.js, React, TypeScript, SQLite та інші інструменти, що забезпечують ефективність, безпеку та простоту розробки. Загалом, запропоновані рішення формують сучасний, масштабований та зручний у використанні вебзастосунок.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ

3.1 Реалізація програмного продукту

Реалізація програмного продукту виконувалася відповідно до попередньо розробленої архітектури системи та визначених функціональних вимог. Основу застосунку становить модульна структура, в якій окремі компоненти відповідають за роботу з базою даних, автентифікацію користувачів, серверну логіку обробки запитів, а також інтерфейсні частини системи. Такий підхід забезпечує розділення відповідальності між модулями та спрощує подальший супровід і масштабування проєкту.

На першому етапі реалізації було створено механізм ініціалізації бази даних, який забезпечує єдине підключення до сховища даних протягом усього життєвого циклу застосунку. У межах цього модуля використано патерн «одинак» (singleton), що дозволяє зберігати активне з’єднання в пам’яті та запобігає повторному відкриттю файлу бази даних, зокрема під час «гарячого» перезавантаження застосунку. Після створення з’єднання додатково активуються режими журналювання WAL та підтримка зовнішніх ключів, що підвищує надійність і цілісність даних.

Після цього виконується ініціалізація структури бази даних, яка включає створення таблиць та початкове наповнення тестовими або базовими даними. Загальний алгоритм цього процесу представлено на рисунку 3.1. Як видно з діаграми, система спочатку перевіряє наявність активного з’єднання, і у випадку його відсутності створює нове підключення. Далі виконується налаштування параметрів роботи бази даних, після чого створюється схема та, за необхідності, виконується початкове заповнення таблиць. Така послідовність забезпечує коректний старт системи та унеможливлює дублювання ініціалізації.

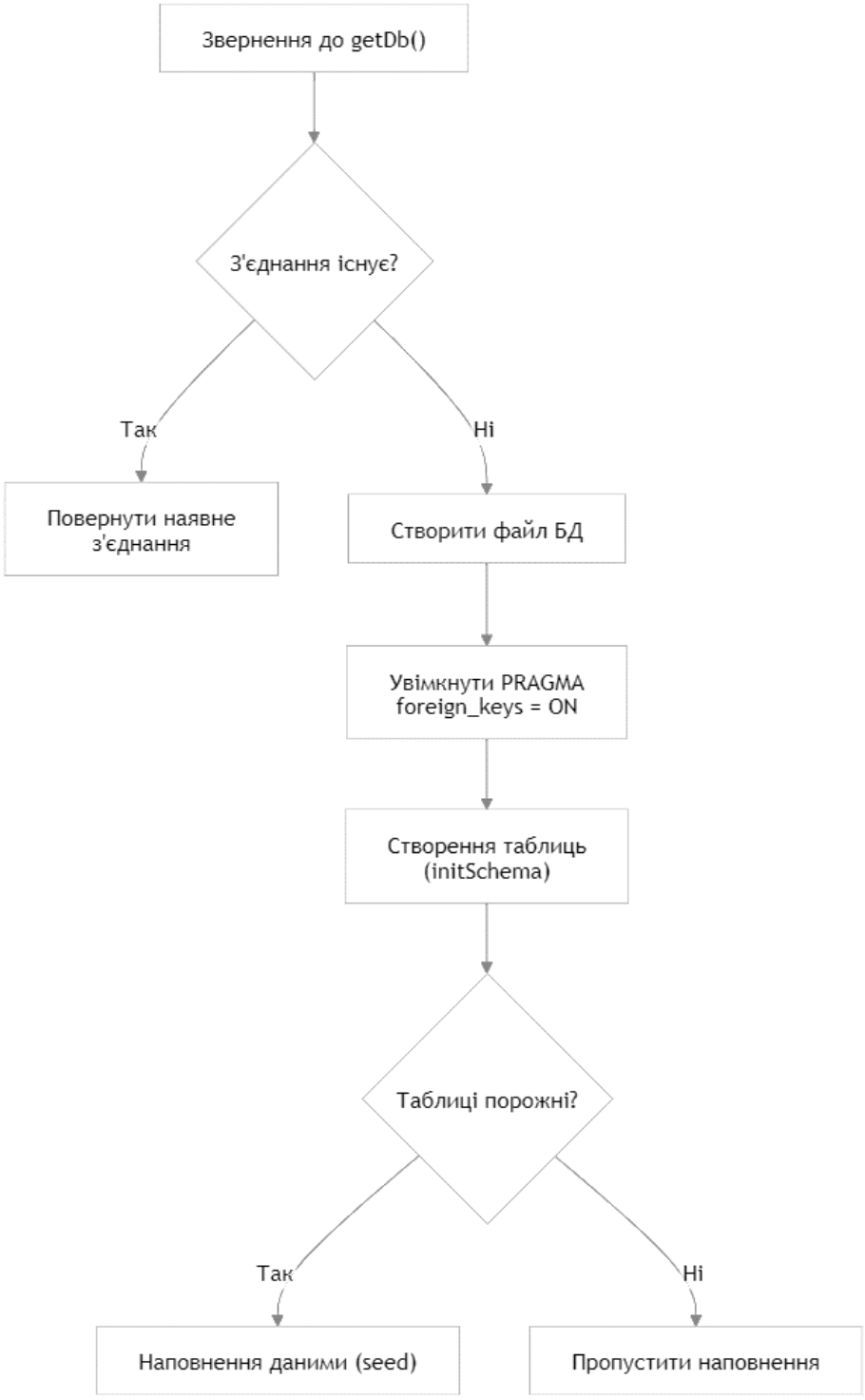


Рисунок 3.1 – Алгоритм ініціалізації бази даних

Діаграма відображає процес ініціалізації бази даних. Спочатку система перевіряє наявність активного з’єднання. Якщо воно вже існує – повертається поточне підключення, якщо ні – створюється новий файл бази даних і налаштовуються параметри роботи SQLite. Далі створюється структура таблиць,

після чого система перевіряє, чи є дані в базі. У разі порожньої бази виконується початкове наповнення (seed), інакше цей етап пропускається.

Наступним важливим компонентом реалізації є модуль доступу до бази даних, який безпосередньо використовує вбудований у Node.js модуль node:sqlite. Код цього модуля наведено в додатку А.1. Його особливістю є використання безпечного способу підключення через process.getBuiltinModule, що дозволяє уникнути проблем сумісності та забезпечує стабільну роботу в різних середовищах виконання. У межах цього модуля також виконується налаштування режиму журналювання та активація зовнішніх ключів, після чого ініціалізується схема даних і виконується початкове наповнення таблиць.

Окремим етапом реалізації стало створення модуля автентифікації користувачів. Код цього функціоналу наведено в додатку А.2. У даному модулі реалізовано реєстрацію, вхід та вихід користувачів із системи, а також механізм управління сесіями. Важливим аспектом є забезпечення безпеки зберігання паролів: вони не зберігаються у відкритому вигляді, а хешуються за допомогою алгоритму bcrypt із використанням 10 раундів. Після успішної реєстрації створюється сесія користувача, яка зберігається у базі даних та передається через захищені cookie-параметри. Під час входу система додатково перевіряє роль користувача, що дозволяє розмежовувати доступ до адміністративних функцій.

Реалізація серверної логіки здійснюється через окремий модуль серверних дій, код якого наведено в додатку А.3. У цьому модулі зосереджено функції, що відповідають за зміну даних у системі, зокрема створення та оновлення сервісів, а також виконання адміністративних операцій. Для забезпечення безпеки використовується перевірка прав доступу через функцію-обгортку requireAdmin, яка перевіряє наявність активної сесії та відповідність ролі користувача. Після виконання змін у базі даних викликається механізм revalidatePath, який оновлює кеш сторінок і забезпечує відображення актуальних даних без перезавантаження застосунку.

Завершальним елементом реалізації програмного продукту є система відгуків користувачів, код якої наведено в додатку А.4. Її особливістю є

реалізація логіки «один користувач – один відгук на сервіс», що забезпечується за допомогою унікального обмеження в базі даних та механізму upsert. Це дозволяє одночасно реалізувати створення та оновлення відгуку однією операцією, що спрощує логіку клієнтської частини та підвищує ефективність роботи з даними.

Загальний алгоритм обробки відгуків користувачів представлено на рисунку 3.2.

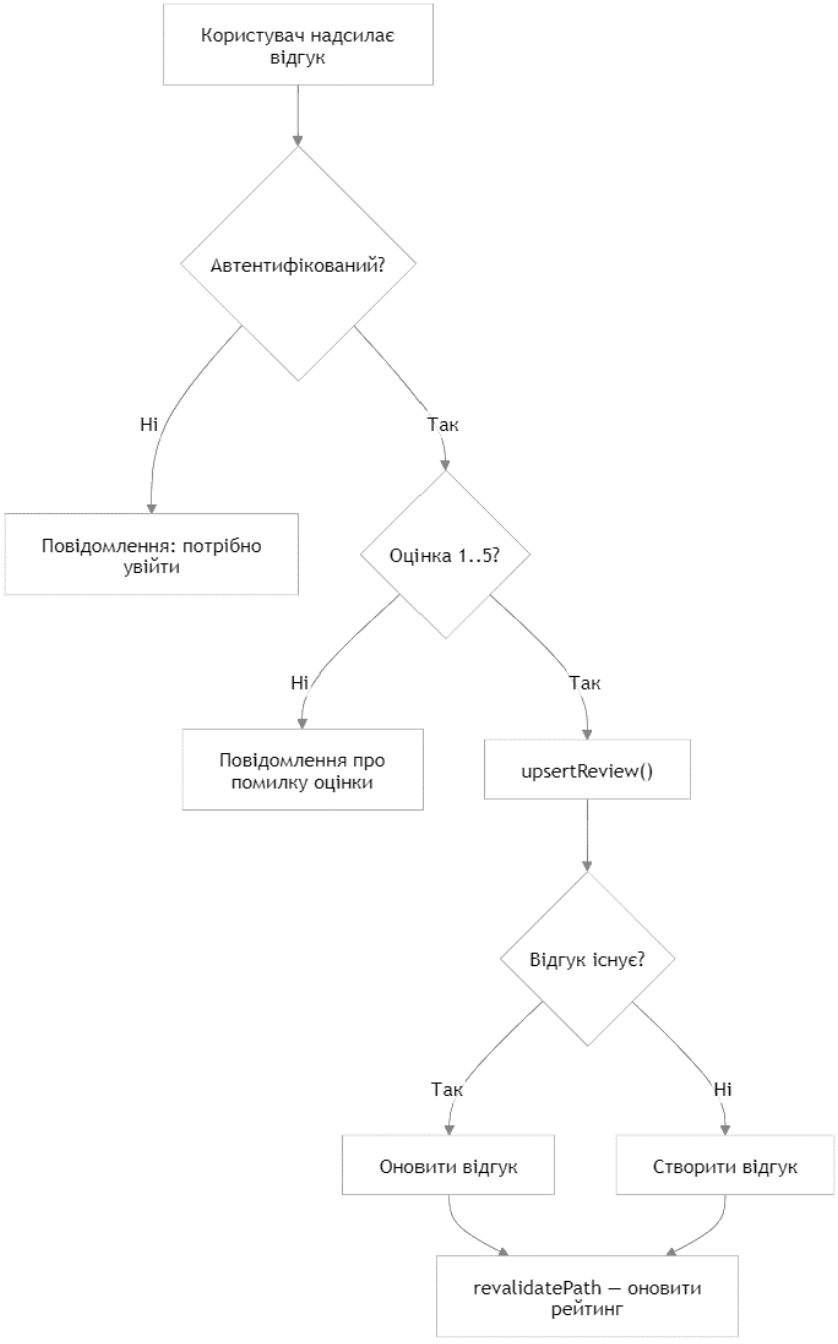


Рисунок 3.2 – Алгоритм обробки відгуку користувача

Згідно з діаграмою, перед поданням відгуку система перевіряє, чи автентифікований користувач. У разі відсутності авторизації йому пропонується виконати вхід у систему. Якщо користувач авторизований, перевіряється коректність оцінки, після чого виконується операція додавання або оновлення відгуку. На завершальному етапі система оновлює дані рейтингу за допомогою механізму revalidatePath, що забезпечує актуальність відображуваної інформації.

Розроблений інтерфейс системи наведено в додатку Б.

Таким чином, реалізований програмний продукт об’єднує взаємопов’язані модулі, які забезпечують стабільну роботу системи, захист даних, розмежування доступу та інтерактивну взаємодію користувачів із сервісом.

3.2 Розробка функціоналу пошуку та каталогізації сервісів штучного інтелекту

Розробка функціоналу пошуку та каталогізації сервісів штучного інтелекту є одним із ключових етапів реалізації програмного продукту, оскільки саме цей модуль забезпечує користувачеві доступ до структурованого переліку сервісів і дозволяє ефективно здійснювати їх відбір за різними критеріями. Основна логіка каталогу побудована таким чином, щоб підтримувати одночасне використання повнотекстового пошуку та фільтрації за категоріями, що значно підвищує зручність взаємодії із системою.

Принцип роботи каталогу полягає у передачі параметрів пошуку через URL-рядок (наприклад, ?q=...&category=...), що дозволяє зберігати стан сторінки, використовувати навігацію браузера та формувати прямі посилання на відфільтровані результати. Взаємодію між користувачем, клієнтською частиною, сервером та базою даних під час виконання пошуку наведено на рисунку 3.3. Як видно з діаграми, процес починається з дії користувача, який вводить запит або обирає категорію. Далі компонент фільтрів формує новий URL і через router.push() ініціює оновлення сторінки. Серверна частина перехоплює

параметри запиту, виконує звернення до бази даних SQLite та повертає відфільтрований список сервісів разом із додатковими агрегованими даними, такими як рейтинг і кількість відгуків.

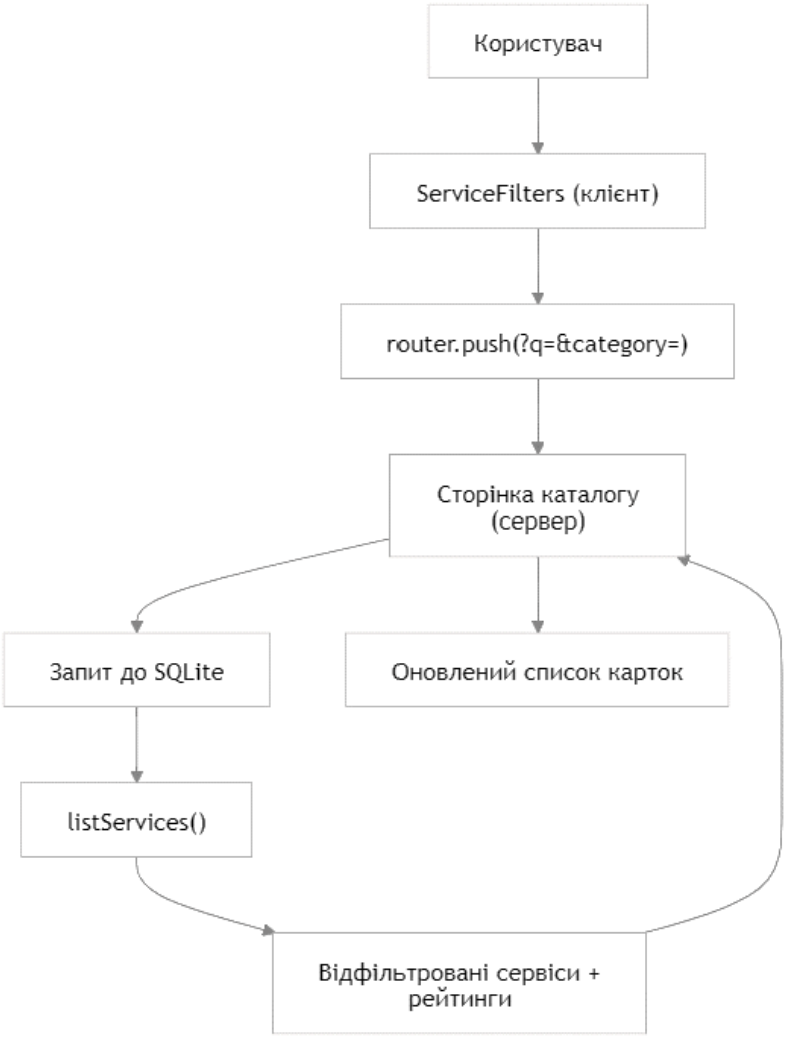


Рисунок 3.3 – Взаємодія компонентів під час пошуку та фільтрації сервісів

Логіка формування запитів до бази даних реалізована у функції listServices, яка динамічно будує SQL-запит залежно від переданих параметрів. Код цього функціоналу наведено в додатку А.6. У межах цієї функції умови пошуку додаються до масиву where лише за наявності відповідних параметрів, що дозволяє формувати гнучкі та оптимізовані запити. Наприклад, при наявності текстового запиту виконується пошук одразу за кількома полями таблиці сервісів – назвою, описом та розробником. Це реалізовано через оператор LIKE, який

забезпечує часткове співпадіння рядків. У спрощеному вигляді ця логіка виглядає наступним чином:

```
if (filter.search && filter.search.trim()) {
  where.push("(s.name LIKE ? OR s.description LIKE ? OR s.developer LIKE ?)")
  const like = `%${filter.search.trim()}%`
  params.push(like, like, like)
}
```

Якщо ж користувач обирає категорію, до запиту додається додаткова умова точного збігу за ідентифікатором категорії:

```
if (filter.categoryId) {
  where.push("s.category_id = ?")
  params.push(filter.categoryId)
}
```

Такий підхід забезпечує одночасну підтримку як повнотекстового пошуку, так і структурованої фільтрації без дублювання логіки та з мінімальними витратами на виконання запитів.

Окрему роль у реалізації каталогу відіграє механізм обчислення рейтингу сервісів. Для цього використовується SQL-запит із об’єднанням таблиць сервісів та відгуків за допомогою LEFT JOIN. Код запиту наведено в додатку А.7. Він дозволяє отримати не лише основні дані про сервіс, але й обчислити середній рейтинг та кількість відгуків у межах однієї вибірки. У спрощеному вигляді цей запит виглядає наступним чином:

```
SELECT s.*,
    ROUND(AVG(r.rating), 1) AS avg_rating,
    COUNT(r.id) AS review_count
```

```
FROM services s
LEFT JOIN reviews r ON r.service_id = s.id
WHERE ...
GROUP BY s.id
```

Завдяки використанню LEFT JOIN сервіси, які ще не мають відгуків, також відображаються у каталозі, але з порожнім значенням рейтингу. Це забезпечує повноту вибірки та коректне відображення всіх записів незалежно від активності користувачів.

Клієнтська частина фільтрації реалізована у компоненті ServiceFilters, код якого наведено в додатку А.8. Його основна задача полягає не у виконанні запитів до бази даних, а у зміні параметрів URL та ініціації повторного рендерингу сторінки. При зміні параметрів використовується метод router.push, який оновлює адресу сторінки без повного перезавантаження застосунку. Для підвищення плавності інтерфейсу використовується useTransition, що дозволяє виконувати оновлення стану асинхронно і не блокувати взаємодію користувача з інтерфейсом:

```
startTransition(() => {
  router.push(`/services?q=${query}&category=${category}`)
})
```

Окремо слід зазначити, що каталог сервісів підтримує категоризацію даних, де кожен сервіс належить до однієї категорії у відношенні типу 1:N. Це дозволяє впорядкувати сервіси за напрямами використання штучного інтелекту, такими як генерація тексту, зображень, коду, аудіо та інші прикладні області. Така структура забезпечує логічну організацію даних і спрощує навігацію користувачів у системі.

Отже, реалізовано функціонал пошуку та каталогізації сервісів штучного інтелекту, який є ключовим компонентом розробленої системи. Сформовано

механізм динамічного формування SQL-запитів, що дозволяє поєднувати повнотекстовий пошук за кількома атрибутами сервісів із фільтрацією за категоріями, забезпечуючи гнучкість і точність відбору даних.

Окрему увагу приділено оптимізації роботи з базою даних через використання параметризованих запитів, що підвищує рівень безпеки та захищає систему від SQL-ін’єкцій. Також реалізовано механізм обчислення середнього рейтингу та кількості відгуків безпосередньо під час формування вибірки, що зменшує кількість додаткових звернень до бази даних і підвищує продуктивність системи.

Клієнтська частина функціоналу побудована на основі роботи з URL-параметрами, що забезпечує зручну навігацію, можливість збереження стану фільтрації та коректну інтеграцію з серверним рендерингом. У результаті створено ефективний, масштабований і зручний у використанні модуль каталогу, який забезпечує швидкий доступ до сервісів та їх структуровану організацію.

3.3 Тестування системи та аналіз результатів її роботи

У процесі розробки програмного продукту було проведено комплексне тестування, спрямоване на перевірку коректності реалізації бізнес-логіки, стабільності роботи системи та відповідності функціональних модулів вимогам технічного завдання. Тестування виконувалося з урахуванням поділу системи на окремі логічні компоненти, що дозволило перевірити як ізольовану роботу функцій, так і взаємодію між модулями в реальних сценаріях використання.

Загальний підхід до перевірки передбачав використання двох основних рівнів тестування: модульного та функціонального. Модульне тестування було спрямоване на перевірку окремих функцій бізнес-логіки без залежності від бази даних та інтерфейсу користувача, тоді як функціональне тестування забезпечувало оцінку роботи системи з точки зору кінцевого користувача в умовах браузерної взаємодії.

Модульне тестування виконувалося із використанням фреймворку Jest і охоплювало ключові допоміжні функції, винесені до окремого модуля lib/helpers.ts. Такий підхід дозволив забезпечити ізольовану перевірку логіки без впливу зовнішніх залежностей, що підвищило надійність результатів.

Перелік реалізованих тестових сценаріїв подано в таблиці 3.1.

Таблиця 3.1 – Перелік модульних тестів системи

Функція	**Опис перевірки**	**Кількість тестів**
slugify	Формування URL-friendly рядка та видалення спеціальних символів	2
pluralizeService	Коректність відмінювання слова «сервіс» у різних формах	5
validateRegistration	Перевірка валідності даних при реєстрації користувача	3
isValidRating	Перевірка допустимого діапазону оцінки (1–5)	2
averageRating	Обчислення середнього значення рейтингу	2

Загалом модульні тести підтвердили коректність роботи всіх перевірених функцій. Наприклад, для функції обчислення середнього рейтингу було перевірено як базовий випадок (відсутність оцінок), так і сценарій із декількома значеннями, що дозволило підтвердити правильність математичної логіки:

```
describe("averageRating", () => {
  it("повертає 0 без відгуків", () => {
    expect(averageRating([])).toBe(0)
  })

  it("коректно обчислює середнє значення", () => {
    expect(averageRating([{ rating: 5 }, { rating: 4 }, { rating: 4 }])).toBe(4.3)
  })
})
```

Функціональне тестування проводилося вручну в середовищі браузера та охоплювало основні сценарії використання системи різними типами користувачів. Основною метою було перевірити коректність роботи інтерфейсу, взаємодію з серверною частиною та обробку користувацьких дій у реальному часі.

Результати функціонального тестування наведено в таблиці 3.2.

Таблиця 3.2 – Результати функціонального тестування

№	Тестовий сценарій	Очікуваний результат	Результат
1	Реєстрація нового користувача	Створення облікового запису та авторизація	Успішно
2	Вхід у систему	Перенаправлення до каталогу сервісів	Успішно
3	Доступ адміністратора	Вхід до адміністративної панелі	Успішно
4	Спроба доступу без прав	Блокування доступу	Успішно
5	Пошук сервісів	Відображення релевантних результатів	Успішно
6	Фільтрація за категоріями	Коректне формування вибірки	Успішно
7	Перегляд деталей сервісу	Відображення повної інформації	Успішно
8	Додавання відгуку	Збереження та оновлення рейтингу	Успішно
9	Створення нового сервісу (адмін)	Додавання запису в каталог	Успішно
10	Редагування сервісу	Оновлення даних у базі	Успішно
11	Видалення сервісу	Видалення з каталогу	Успішно
12	Захист адміністративних дій	Заборона несанкціонованих операцій	Успішно

Отримані результати тестування свідчать про стабільну роботу програмного продукту та коректну реалізацію всіх основних функціональних модулів. Усі модульні тести були успішно пройдені, що підтверджує правильність реалізації базової логіки системи, включаючи обробку даних, валідацію вхідної інформації та виконання допоміжних функцій. Це дозволяє зробити висновок, що внутрішні алгоритми системи працюють передбачувано та відповідають закладеним вимогам.

Функціональні сценарії також виконані без помилок, що вказує на коректну взаємодію клієнтської та серверної частин, а також стабільну роботу механізмів маршрутизації, авторизації та обробки користувацьких запитів. Особливо важливо, що всі ключові сценарії – від реєстрації та входу користувача

до пошуку сервісів, фільтрації та взаємодії з відгуками – були реалізовані без збоїв і забезпечили очікувану поведінку системи.

Під час тестування не було виявлено критичних помилок, які могли б впливати на працездатність або безпеку системи. Виявлені незначні неточності та поведінкові особливості були оперативно усунені ще на етапі розробки, що дозволило підвищити загальну стабільність та передбачуваність роботи застосунку. Окремо варто відзначити, що тестування також підтвердило коректність реалізованих механізмів захисту даних, зокрема авторизації, перевірки ролей користувачів та обробки сесій.

Загалом система продемонструвала повну відповідність визначеним функціональним і нефункціональним вимогам. Отримані результати дозволяють зробити висновок про готовність програмного продукту до практичного використання, а також його потенційну масштабованість для подальшого розширення функціоналу без необхідності суттєвих змін у базовій архітектурі.

3.4 Інструкція користувача

Інформаційно-довідкова система «Сервіси штучного інтелекту» призначена для зручного перегляду, пошуку та каталогізації сучасних сервісів штучного інтелекту. Система доступна через веббраузер і не потребує встановлення додаткового програмного забезпечення, що забезпечує простоту її використання для кінцевого користувача.

Після відкриття вебзастосунку користувач потрапляє на головну сторінку, де представлено загальну інформацію про систему та доступ до основних розділів. Через навігаційне меню користувач може перейти до каталогу сервісів, сторінки авторизації або реєстрації, а також перегляду детальної інформації про окремі сервіси.

Для повноцінного використання функціоналу системи необхідно пройти процедуру реєстрації, яка передбачає введення імені, електронної пошти та

пароля. Після успішної реєстрації користувач отримує доступ до особистого облікового запису та може виконувати дії, доступні для авторизованих користувачів. Вхід у систему здійснюється через форму авторизації шляхом введення зареєстрованих даних. У разі некоректного введення система повідомляє користувача про помилку.

Основним функціональним елементом системи є каталог сервісів. На цій сторінці користувач може переглядати перелік доступних сервісів штучного інтелекту у вигляді карток, що містять базову інформацію: назву, категорію та короткий опис. Для зручності реалізовано механізм пошуку за ключовими словами, який дозволяє швидко знаходити потрібні сервіси, а також фільтрацію за категоріями. Зміна параметрів пошуку автоматично оновлює список результатів.

При переході на сторінку окремого сервісу користувач отримує розширену інформацію, включаючи детальний опис, функціональні можливості та середній рейтинг, сформований на основі відгуків інших користувачів. Також на цій сторінці доступна система відгуків, яка дозволяє оцінити сервіс та залишити коментар.

Авторизовані користувачі мають можливість додавати власні відгуки до кожного сервісу, а також редагувати вже створені. При цьому реалізовано правило унікальності відгуку, відповідно до якого один користувач може залишити лише один відгук на кожен окремий сервіс. У разі повторного надсилання відгуку система не створює новий запис, а автоматично оновлює вже наявний, зберігаючи актуальність оцінки та коментаря.

Додатково користувачі можуть переглядати середній рейтинг сервісу, який формується на основі всіх залишених відгуків, що дозволяє отримати узагальнену оцінку якості сервісу перед його використанням. Інтерфейс взаємодії з відгуками є інтуїтивно зрозумілим і включає форму введення рейтингу та текстового коментаря, а також відображення вже опублікованих відгуків інших користувачів.

Для адміністратора системи передбачено окрему панель керування, яка надає розширені можливості адміністрування контенту. З її допомогою можна додавати нові сервіси до каталогу, редагувати вже існуючі записи, оновлювати їх опис, категорію або інші характеристики, а також видаляти застарілі чи неактуальні дані. Окрім цього, реалізовано керування категоріями сервісів, що дозволяє підтримувати структуровану організацію каталогу та забезпечує зручну навігацію для користувачів.

Також адміністративна частина включає контроль за користувацькими даними та забезпечує обмеження доступу до критичних функцій системи відповідно до ролей користувачів. Це дозволяє запобігти несанкціонованому внесенню змін та забезпечує цілісність даних.

У разі виникнення помилок або некоректних дій з боку користувача система відображає інформативні повідомлення, які пояснюють причину проблеми та підказують можливі шляхи її вирішення. Такий підхід підвищує зручність використання системи та зменшує ймовірність помилок під час взаємодії з інтерфейсом.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було здійснено комплексне проєктування, розробку та тестування інформаційно-довідкової системи «Сервіси штучного інтелекту». Основною метою роботи було створення сучасного веб-застосунку, який забезпечує структуроване зберігання інформації про сервіси штучного інтелекту, а також надає зручні механізми їх пошуку, фільтрації, оцінювання та адміністрування. Поставлена мета була досягнута в повному обсязі, а всі визначені завдання виконані відповідно до вимог технічного завдання.

У теоретичній частині роботи було досліджено поняття інформаційно-довідкових систем, їх основні характеристики та роль у сучасному інформаційному середовищі. Розглянуто особливості систем, що працюють із великими обсягами структурованих даних, а також проаналізовано сучасні аналогічні рішення. Проведений аналіз дозволив виявити низку типових проблем існуючих систем, зокрема недостатньо гнучкі механізми пошуку, обмежені можливості фільтрації та відсутність спеціалізації на динамічно зростаючій галузі сервісів штучного інтелекту. На основі цього було сформульовано вимоги до власної системи, включаючи функціональні можливості, вимоги до безпеки, продуктивності та зручності використання.

У проєктному розділі було виконано детальне моделювання структури системи, визначено її основні компоненти та логіку взаємодії між ними. Спроєктовано базу даних, яка забезпечує цілісне зберігання інформації про сервіси, категорії, користувачів та відгуки, а також підтримує зв’язки між сутностями. Окрему увагу приділено побудові функціональних модулів системи та проєктуванню користувацького інтерфейсу, який орієнтований на інтуїтивну навігацію та мінімізацію кількості дій користувача для отримання необхідної інформації. Також було обґрунтовано вибір сучасного технологічного стеку, що забезпечив ефективну реалізацію клієнтської та серверної частин застосунку.

У практичній частині кваліфікаційної роботи було реалізовано повноцінний програмний продукт. Розроблено серверну логіку, що включає роботу з базою даних, автентифікацію користувачів, розмежування прав доступу та обробку бізнес-логіки системи. Реалізовано ключовий функціонал, зокрема каталог сервісів, систему пошуку та фільтрації, модуль відгуків і рейтингів, а також адміністративну панель для керування даними. Особливу увагу приділено безпеці системи: використано хешування паролів, параметризовані SQL-запити та механізми контролю доступу, що мінімізує ризики несанкціонованого втручання.

Окремим важливим етапом стало тестування розробленої системи. Було проведено модульне тестування окремих функцій бізнес-логіки та функціональне тестування основних сценаріїв використання системи. Результати тестування підтвердили коректність роботи всіх основних модулів, відсутність критичних помилок та стабільність системи в різних сценаріях використання. Виявлені незначні недоліки були усунені на етапі розробки, що позитивно вплинуло на загальну якість програмного продукту.

Загалом розроблена інформаційно-довідкова система «Сервіси штучного інтелекту» є функціонально завершеним, стабільним і розширюваним програмним рішенням. Вона забезпечує зручний доступ до структурованої інформації, підтримує ефективний пошук і фільтрацію даних, а також дозволяє користувачам взаємодіяти із системою через відгуки та рейтинги. Отримані результати свідчать про те, що система відповідає сучасним вимогам до веб-застосунків подібного типу та може бути використана як основа для подальшого розвитку, зокрема інтеграції з зовнішніми API, розширення аналітичних можливостей та впровадження елементів штучного інтелекту для персоналізації контенту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Freundschuh S. M. Wayfinding and Navigation Behavior. *International Encyclopedia of the Social & Behavioral Sciences.* Elsevier eBooks, 2001. P. 16391–16394. DOI : https://doi.org/10.1016/b0-08-043076-7/02495-5.

2. Шаров С. В., Нікітенко Д. С. Визначення та завдання довідково-інформаційних систем. *Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення :* збірник тез доповідей Всеукраїнської наукової Internet-конференції (м. Тернопіль, 30–31 січня 2014). № 1. Тернопіль : Тайп, 2014. С. 15–19.

3. Information and Reference Subsystem. CLIR. URL : https://www.clir.org/pubs/reports/pub76/information/ (date of access : 05.04.2026).

4. Science Reference Source. EBSCO Information Services. URL : https://about.ebsco.com/products/research-databases/science-reference-source (date of access : 05.04.2026).

5. Research guides & tutorials: Science resources: Reference sources. Ambrose University Library Guides. URL : https://ambrose.libguides.com/science/reference-sources (date of access : 05.04.2026).

6. Хахановський В. Г., Корнейко О. В. Актуальні питання інформаційного права : навч. посіб. Київ : Нац. акад. внутр. справ, 2024. 258 с.

7. Тренди штучного інтелекту у 2025 році: революційні зміни та нові можливості. *TechExpert.* 2025. URL : https://techexpert.ua/artificial-intelligence-trends-in-2025/ (дата звернення : 05.04.2026).

8. Zhao W. X. et al. A survey of large language models. *arXiv*. 2023. URL : https://arxiv.org/abs/2303.18223 (date of access : 05.04.2026).

9. OpenAI Research. OpenAI. URL : https://openai.com/research/ (date of access : 05.04.2026).

10. ChatGPT vs Microsoft Copilot vs Google Gemini : Full report and comparison. *DataStudios*. 2025. URL : https://www.datastudios.org/post/chatgpt-vs-

microsoft-copilot-vs-google-gemini-full-report-and-comparison-july-2025-update (date of access : 05.04.2026).

11. ChatGPT vs Perplexity AI: full report and comparison. DataStudios. 2025. URL : https://www.datastudios.org/post/chatgpt-vs-perplexity-ai-full-report-and-comparison-comparison-of-features-capabilities-pricing (date of access : 05.04.2026).

12. ChatGPT vs Perplexity AI: which one is better? *Brytesoft*. 2026. URL : https://brytesoft.com/blog/compare-chatgpt-and-perplexity-ai-which-ones-better.html (date of access : 05.04.2026).

13. Perplexity vs ChatGPT, Gemini & Claude: AI comparison 2025. SentiSight. 2025. URL : https://www.sentisight.ai/perplexity-vs-other-genai-models/ (date of access : 05.04.2026).

14. Perplexity AI vs ChatGPT in 2025: key differences and features. 2025. URL : https://4idiotz.com/tech/artificial-intelligence/perplexity-ai-vs-chatgpt-in-2025-key-differences-features-which-ai-wins/ (date of access : 05.04.2026).

15. Google Gemini vs Microsoft Copilot: AI tools comparison. *Neontri*. 2025. URL : https://neontri.com/blog/gemini-vs-copilot/ (date of access : 05.04.2026).

16. Copilot vs Gemini : AI assistant comparison 2026. *Cloudwards*. 2026. URL : https://www.cloudwards.net/copilot-vs-gemini/ (date of access : 05.04.2026).

17. Claude vs ChatGPT vs Copilot vs Gemini: enterprise guide 2026. *IntuitionLabs*. 2026. URL : https://intuitionlabs.ai/articles/claude-vs-chatgpt-vs-copilot-vs-gemini-enterprise-comparison (date of access : 05.04.2026).

18. Next.js – the React framework. *Vercel*. URL : https://nextjs.org/ (date of access : 05.04.2026).

19. React – A JavaScript library for building user interfaces. *React*. URL : https://react.dev/ (date of access : 05.04.2026).

20. TypeScript: JavaScript with syntax for types. *TypeScript*. URL : https://www.typescriptlang.org/ (date of access : 05.04.2026).

21. SQLite Documentation. *SQLite*. URL : https://sqlite.org/docs.html (date of access : 05.04.2026).

22. Tailwind CSS Documentation. *Tailwind CSS*. URL : https://tailwindcss.com/ (date of access : 05.04.2026).

23. shadcn/ui Documentation. *shadcn/ui.* URL : https://ui.shadcn.com/ (date of access : 05.04.2026).

24. PostgreSQL Documentation. *PostgreSQL*. URL : https://www.postgresql.org/ (date of access : 05.04.2026).

25. MySQL Documentation. *MySQL*. URL : https://dev.mysql.com/doc/ (date of access : 05.04.2026).

ДОДАТОК А

Лістинг реалізації програмного продукту

Лістинг А.1 lib/db.ts

```
import bcrypt from "bcryptjs"
import path from "path"
import fs from "fs"

// node:sqlite is a built-in Node.js module (Node 22+). Access it via
// process.getBuiltinModule so the bundler does not try to resolve/bundle it.
const { DatabaseSync } = process.getBuiltinModule("node:sqlite")
type DatabaseSync = InstanceType<typeof DatabaseSync>

// Singleton database connection (survives hot reloads in dev)
const globalForDb = globalThis as unknown as { __aidb?: DatabaseSync }

const DB_DIR = path.join(process.cwd(), "data")
const DB_PATH = path.join(DB_DIR, "ai-services.db")

function createConnection(): DatabaseSync {
  if (!fs.existsSync(DB_DIR)) {
    fs.mkdirSync(DB_DIR, { recursive: true })
  }
  const db = new DatabaseSync(DB_PATH)
  db.exec("PRAGMA journal_mode = WAL")
  db.exec("PRAGMA foreign_keys = ON")
  initSchema(db)
  seed(db)
  return db
}

function initSchema(db: DatabaseSync) {
```

```
db.exec(`
  CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    role TEXT NOT NULL DEFAULT 'user' CHECK (role IN ('user', 'admin')),
    created_at TEXT NOT NULL DEFAULT (datetime('now'))
  );

  CREATE TABLE IF NOT EXISTS categories (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL UNIQUE,
    slug TEXT NOT NULL UNIQUE,
    description TEXT
  );

  CREATE TABLE IF NOT EXISTS services (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    description TEXT NOT NULL,
    url TEXT,
    pricing TEXT,
    developer TEXT,
    category_id INTEGER,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET NULL
  );

  CREATE TABLE IF NOT EXISTS reviews (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    service_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    rating INTEGER NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    FOREIGN KEY (service_id) REFERENCES services(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
```

```
    UNIQUE (service_id, user_id)
   );

   CREATE TABLE IF NOT EXISTS sessions (
    id TEXT PRIMARY KEY,
    user_id INTEGER NOT NULL,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
   );
  `)
 }

 function seed(db: DatabaseSync) {
  const userCount = db.prepare("SELECT COUNT(*) AS c FROM users").get() as { c: number }
  if (userCount.c === 0) {
   const insertUser = db.prepare(
    "INSERT INTO users (name, email, password_hash, role) VALUES (?, ?, ?, ?)",
   )
   insertUser.run("Адміністратор", "admin@ai.ua", bcrypt.hashSync("admin123", 10), "admin")
   insertUser.run("Тестовий користувач", "user@ai.ua", bcrypt.hashSync("user123", 10), "user")
  }

  const catCount = db.prepare("SELECT COUNT(*) AS c FROM categories").get() as { c: number }
  if (catCount.c === 0) {
   const insertCat = db.prepare(
    "INSERT INTO categories (name, slug, description) VALUES (?, ?, ?)",
   )
   const categories = [
    ["Текстові чат-боти", "chatbots", "Універсальні ШІ-асистенти для діалогу та генерації тексту"],
    ["Генерація зображень", "images", "Сервіси для створення зображень за текстовим описом"],
    ["Генерація відео", "video", "Інструменти для створення та редагування відео за допомогою ШІ"],
    ["Програмування", "coding", "ШІ-помічники для написання та аналізу коду"],
    ["Аудіо та музика", "audio", "Синтез голосу, музики та обробка звуку"],
    ["Продуктивність", "productivity", "ШІ-інструменти для роботи з документами та задачами"],
   ]
```

```
  db.exec("BEGIN")
  for (const c of categories) insertCat.run(c[0], c[1], c[2])
  db.exec("COMMIT")
 }

 const svcCount = db.prepare("SELECT COUNT(*) AS c FROM services").get() as { c: number }
 if (svcCount.c === 0) {
  const cats = db.prepare("SELECT id, slug FROM categories").all() as { id: number; slug: string }[]
  const catId = (slug: string) => cats.find((c) => c.slug === slug)?.id ?? null

  const insertSvc = db.prepare(
   "INSERT INTO services (name, description, url, pricing, developer, category_id) VALUES (?, ?, ?, ?, ?, ?)",
  )
  const services: [string, string, string, string, string, number | null][] = [
   ["ChatGPT", "Розмовний ШІ від OpenAI на базі моделей GPT для генерації тексту, відповідей на запитання, написання коду та аналізу даних.", "https://chat.openai.com", "Freemium", "OpenAI", catId("chatbots")],
   ["Claude", "Сімейство ШІ-моделей від Anthropic з фокусом на безпеку, довгий контекст та якісний аналіз тексту й документів.", "https://claude.ai", "Freemium", "Anthropic", catId("chatbots")],
   ["Gemini", "Мультимодальний ШІ-асистент від Google з інтеграцією у сервіси Google та підтримкою тексту, зображень і коду.", "https://gemini.google.com", "Freemium", "Google", catId("chatbots")],
   ["Midjourney", "Один з найпопулярніших генераторів зображень за текстовим описом з високою художньою якістю результатів.", "https://midjourney.com", "Платний", "Midjourney", catId("images")],
   ["DALL·E 3", "Генератор зображень від OpenAI з точним розумінням текстових запитів та інтеграцією у ChatGPT.", "https://openai.com/dall-e-3", "Freemium", "OpenAI", catId("images")],
   ["Stable Diffusion", "Відкрита модель генерації зображень, яку можна запускати локально та налаштовувати під власні задачі.", "https://stability.ai", "Безкоштовний", "Stability AI", catId("images")],
   ["Sora", "Модель від OpenAI для генерації реалістичного відео за текстовим описом.", "https://openai.com/sora", "Платний", "OpenAI", catId("video")],
   ["Runway", "Платформа для генерації та редагування відео за допомогою ШІ, популярна серед креативних професіоналів.", "https://runwayml.com", "Freemium", "Runway", catId("video")],
   ["GitHub Copilot", "ШІ-помічник для програмістів, що пропонує автодоповнення коду безпосередньо в редакторі.", "https://github.com/features/copilot", "Платний", "GitHub", catId("coding")],
   ["Cursor", "Редактор коду зі вбудованим ШІ для генерації, рефакторингу та пояснення коду.", "https://cursor.com", "Freemium", "Anysphere", catId("coding")],
```

```
["ElevenLabs", "Сервіс синтезу реалістичного голосу та клонування голосу на багатьох мовах.", "https://elevenlabs.io", "Freemium", "ElevenLabs", catId("audio")],
["Suno", "Генератор музики та пісень за текстовим описом, включно з вокалом та інструментами.", "https://suno.com", "Freemium", "Suno", catId("audio")],
["Notion AI", "Вбудований у Notion ШІ-помічник для написання, підсумовування та організації нотаток.", "https://notion.so/product/ai", "Платний", "Notion Labs", catId("productivity")],
["Perplexity", "ШІ-пошуковик, що дає відповіді з посиланнями на джерела в реальному часі.", "https://perplexity.ai", "Freemium", "Perplexity AI", catId("productivity")],
    ]
    db.exec("BEGIN")
    for (const s of services) insertSvc.run(...s)
    db.exec("COMMIT")
  }
}

export function getDb(): DatabaseSync {
  if (!globalForDb.__aidb) {
    globalForDb.__aidb = createConnection()
  }
  return globalForDb.__aidb
}
```

Лістинг А.2 lib/auth.ts

```
"use server"

import { cookies } from "next/headers"
import { randomUUID } from "crypto"
import bcrypt from "bcryptjs"
import { getDb } from "./db"
import type { User, Role } from "./types"

const SESSION_COOKIE = "ai_session"

export interface AuthResult {
  ok: boolean
  error?: string
```

```
  user?: User
}

function rowToUser(row: any): User {
  return {
    id: row.id,
    name: row.name,
    email: row.email,
    role: row.role as Role,
    created_at: row.created_at,
  }
}

export async function registerUser(name: string, email: string, password: string): Promise<AuthResult> {
  const db = getDb()
  const normalizedEmail = email.trim().toLowerCase()

  if (!name.trim() || !normalizedEmail || password.length < 6) {
    return { ok: false, error: "Заповніть усі поля. Пароль має містити щонайменше 6 символів." }
  }

  const existing = db.prepare("SELECT id FROM users WHERE email = ?").get(normalizedEmail)
  if (existing) {
    return { ok: false, error: "Користувач з такою поштою вже існує." }
  }

  const hash = bcrypt.hashSync(password, 10)
  const info = db
    .prepare("INSERT INTO users (name, email, password_hash, role) VALUES (?, ?, ?, 'user')")
    .run(name.trim(), normalizedEmail, hash)

  await createSession(Number(info.lastInsertRowid))
  const user = db.prepare("SELECT * FROM users WHERE id = ?").get(info.lastInsertRowid)
  return { ok: true, user: rowToUser(user) }
}
```

```
export async function loginUser(email: string, password: string, expectedRole?: Role): Promise<AuthResult> {
  const db = getDb()
  const normalizedEmail = email.trim().toLowerCase()
  const row: any = db.prepare("SELECT * FROM users WHERE email = ?").get(normalizedEmail)

  if (!row || !bcrypt.compareSync(password, row.password_hash)) {
    return { ok: false, error: "Невірна пошта або пароль." }
  }

  if (expectedRole && row.role !== expectedRole) {
    return {
      ok: false,
      error:
        expectedRole === "admin"
          ? "Цей обліковий запис не має прав адміністратора."
          : "Використовуйте вхід для адміністратора.",
    }
  }

  await createSession(row.id)
  return { ok: true, user: rowToUser(row) }
}

async function createSession(userId: number) {
  const db = getDb()
  const sessionId = randomUUID()
  db.prepare("INSERT INTO sessions (id, user_id) VALUES (?, ?)").run(sessionId, userId)
  const store = await cookies()
  store.set(SESSION_COOKIE, sessionId, {
    httpOnly: true,
    sameSite: "lax",
    path: "/",
    maxAge: 60 * 60 * 24 * 7,
  })
}

export async function logout() {
```

```
  const store = await cookies()
  const sessionId = store.get(SESSION_COOKIE)?.value
  if (sessionId) {
    getDb().prepare("DELETE FROM sessions WHERE id = ?").run(sessionId)
    store.delete(SESSION_COOKIE)
  }
}

export async function getCurrentUser(): Promise<User | null> {
  const store = await cookies()
  const sessionId = store.get(SESSION_COOKIE)?.value
  if (!sessionId) return null

  const row: any = getDb()
    .prepare(
      `SELECT u.* FROM sessions s JOIN users u ON u.id = s.user_id WHERE s.id = ?`,
    )
    .get(sessionId)

  return row ? rowToUser(row) : null
}
```

Лістинг А.3 lib/actions.ts

```
"use server"

import { revalidatePath } from "next/cache"
import { redirect } from "next/navigation"
import { getCurrentUser, loginUser, registerUser, logout } from "./auth"
import * as q from "./queries"

export interface FormState {
  error?: string
}

export async function loginAction(_prev: FormState, formData: FormData): Promise<FormState> {
```

```
  const email = String(formData.get("email") ?? "")
  const password = String(formData.get("password") ?? "")
  const res = await loginUser(email, password, "user")
  if (!res.ok) return { error: res.error }
  redirect("/services")
}

export async function adminLoginAction(_prev: FormState, formData: FormData): Promise<FormState> {
  const email = String(formData.get("email") ?? "")
  const password = String(formData.get("password") ?? "")
  const res = await loginUser(email, password, "admin")
  if (!res.ok) return { error: res.error }
  redirect("/admin")
}

export async function registerAction(_prev: FormState, formData: FormData): Promise<FormState> {
  const name = String(formData.get("name") ?? "")
  const email = String(formData.get("email") ?? "")
  const password = String(formData.get("password") ?? "")
  const res = await registerUser(name, email, password)
  if (!res.ok) return { error: res.error }
  redirect("/services")
}

export async function logoutAction() {
  await logout()
  redirect("/")
}

export async function submitReviewAction(serviceId: number, rating: number, comment: string) {
  const user = await getCurrentUser()
  if (!user) return { error: "Потрібно увійти." }
  if (rating < 1 || rating > 5) return { error: "Оцінка має бути від 1 до 5." }
  q.upsertReview(serviceId, user.id, rating, comment.trim())
  revalidatePath(`/services/${serviceId}`)
  return { ok: true }
```

```
}

async function requireAdmin() {
  const user = await getCurrentUser()
  if (!user || user.role !== "admin") throw new Error("Доступ заборонено")
  return user
}

export async function saveServiceAction(formData: FormData) {
  await requireAdmin()
  const id = formData.get("id") ? Number(formData.get("id")) : null
  const payload = {
    name: String(formData.get("name") ?? "").trim(),
    description: String(formData.get("description") ?? "").trim(),
    url: String(formData.get("url") ?? "").trim() || null,
    pricing: String(formData.get("pricing") ?? "").trim() || null,
    developer: String(formData.get("developer") ?? "").trim() || null,
    category_id: formData.get("category_id") ? Number(formData.get("category_id")) : null,
  }
  if (!payload.name || !payload.description) return { error: "Назва та опис обов’язкові." }

  if (id) q.updateService(id, payload)
  else q.createService(payload)
  revalidatePath("/admin")
  revalidatePath("/services")
  return { ok: true }
}

export async function deleteServiceAction(id: number) {
  await requireAdmin()
  q.deleteService(id)
  revalidatePath("/admin")
  revalidatePath("/services")
  return { ok: true }
}

export async function createCategoryAction(formData: FormData) {
  await requireAdmin()
```

```
  const name = String(formData.get("name") ?? "").trim()
  const slug =
    String(formData.get("slug") ?? "")
      .trim()
      .toLowerCase()
      .replace(/\s+/g, "-") || name.toLowerCase().replace(/\s+/g, "-")
  const description = String(formData.get("description") ?? "").trim()
  if (!name) return { error: "Введіть назву категорії." }
  try {
    q.createCategory(name, slug, description)
  } catch {
    return { error: "Категорія з такою назвою або slug уже існує." }
  }
  revalidatePath("/admin")
  return { ok: true }
}

export async function deleteCategoryAction(id: number) {
  await requireAdmin()
  q.deleteCategory(id)
  revalidatePath("/admin")
  return { ok: true }
}

export async function deleteUserAction(id: number) {
  await requireAdmin()
  q.deleteUser(id)
  revalidatePath("/admin")
  return { ok: true }
}
```

Лістинг А.4 – lib/queries.ts

```
import { getDb } from "./db"
import type { Category, Service, ServiceWithMeta, Review, User } from "./types"

// node:sqlite returns rows as null-prototype objects, which cannot be passed
```

```
// to Client Components. Convert them to plain serializable objects.
function plain<T>(row: unknown): T {
  return { ...(row as object) } as T
}
function plainAll<T>(rows: unknown[]): T[] {
  return rows.map((r) => plain<T>(r))
}

export interface ServiceFilter {
  search?: string
  categoryId?: number | null
}

export function listServices(filter: ServiceFilter = {}): ServiceWithMeta[] {
  const db = getDb()
  const where: string[] = []
  const params: any[] = []

  if (filter.search && filter.search.trim()) {
    where.push("(s.name LIKE ? OR s.description LIKE ? OR s.developer LIKE ?)")
    const like = `%${filter.search.trim()}%`
    params.push(like, like, like)
  }
  if (filter.categoryId) {
    where.push("s.category_id = ?")
    params.push(filter.categoryId)
  }

  const whereClause = where.length ? `WHERE ${where.join(" AND ")}` : ""

  const rows = db
    .prepare(
      `SELECT s.*, c.name AS category_name,
              ROUND(AVG(r.rating), 1) AS avg_rating,
              COUNT(r.id) AS review_count
       FROM services s
       LEFT JOIN categories c ON c.id = s.category_id
       LEFT JOIN reviews r ON r.service_id = s.id
```

```
      ${whereClause}
      GROUP BY s.id
      ORDER BY s.name COLLATE NOCASE ASC`,
    )
    .all(...params)

  return plainAll<ServiceWithMeta>(rows)
}

export function getService(id: number): ServiceWithMeta | null {
  const db = getDb()
  const row = db
    .prepare(
      `SELECT s.*, c.name AS category_name,
            ROUND(AVG(r.rating), 1) AS avg_rating,
            COUNT(r.id) AS review_count
      FROM services s
      LEFT JOIN categories c ON c.id = s.category_id
      LEFT JOIN reviews r ON r.service_id = s.id
      WHERE s.id = ?
      GROUP BY s.id`,
    )
    .get(id)
  return row ? plain<ServiceWithMeta>(row) : null
}

export function listCategories(): Category[] {
  return plainAll<Category>(
      getDb().prepare("SELECT * FROM categories ORDER BY name COLLATE NOCASE
ASC").all(),
  )
}

export function listReviews(serviceId: number): Review[] {
  return plainAll<Review>(
    getDb()
      .prepare(
        `SELECT r.*, u.name AS user_name
```

```
      FROM reviews r JOIN users u ON u.id = r.user_id
      WHERE r.service_id = ?
      ORDER BY r.created_at DESC`,
     )
     .all(serviceId),
  )
}

export function upsertReview(serviceId: number, userId: number, rating: number, comment: string) {
 getDb()
  .prepare(
   `INSERT INTO reviews (service_id, user_id, rating, comment)
    VALUES (?, ?, ?, ?)
    ON CONFLICT (service_id, user_id)
      DO UPDATE SET rating = excluded.rating, comment = excluded.comment, created_at = datetime('now')`,
  )
  .run(serviceId, userId, rating, comment)
}

export function createService(s: Omit<Service, "id" | "created_at">) {
 return getDb()
  .prepare(
   `INSERT INTO services (name, description, url, pricing, developer, category_id)
    VALUES (?, ?, ?, ?, ?, ?)`,
  )
  .run(s.name, s.description, s.url, s.pricing, s.developer, s.category_id)
}

export function updateService(id: number, s: Omit<Service, "id" | "created_at">) {
 getDb()
  .prepare(
   `UPDATE services SET name = ?, description = ?, url = ?, pricing = ?, developer = ?, category_id = ?
    WHERE id = ?`,
  )
  .run(s.name, s.description, s.url, s.pricing, s.developer, s.category_id, id)
}
```

```
export function deleteService(id: number) {
  getDb().prepare("DELETE FROM services WHERE id = ?").run(id)
}

export function createCategory(name: string, slug: string, description: string) {
  getDb()
    .prepare("INSERT INTO categories (name, slug, description) VALUES (?, ?, ?)")
    .run(name, slug, description)
}

export function deleteCategory(id: number) {
  getDb().prepare("DELETE FROM categories WHERE id = ?").run(id)
}

export function listUsers(): User[] {
  return plainAll<User>(
    getDb()
      .prepare("SELECT id, name, email, role, created_at FROM users ORDER BY created_at DESC")
      .all(),
  )
}

export function deleteUser(id: number) {
  getDb().prepare("DELETE FROM users WHERE id = ? AND role != 'admin'").run(id)
}

export interface AdminStats {
  services: number
  categories: number
  users: number
  reviews: number
}

export function getStats(): AdminStats {
  const db = getDb()
  const one = (q: string) => (db.prepare(q).get() as { c: number }).c
  return {
```

```
    services: one("SELECT COUNT(*) AS c FROM services"),
    categories: one("SELECT COUNT(*) AS c FROM categories"),
    users: one("SELECT COUNT(*) AS c FROM users"),
    reviews: one("SELECT COUNT(*) AS c FROM reviews"),
  }
}

export interface CategoryStat {
  name: string
  count: number
}

export function getServicesPerCategory(): CategoryStat[] {
  return plainAll<CategoryStat>(
    getDb()
      .prepare(
        `SELECT c.name AS name, COUNT(s.id) AS count
       FROM categories c LEFT JOIN services s ON s.category_id = c.id
       GROUP BY c.id ORDER BY count DESC`,
      )
      .all(),
  )
}
```

ДОДАТОК Б

Інтерфейс системи

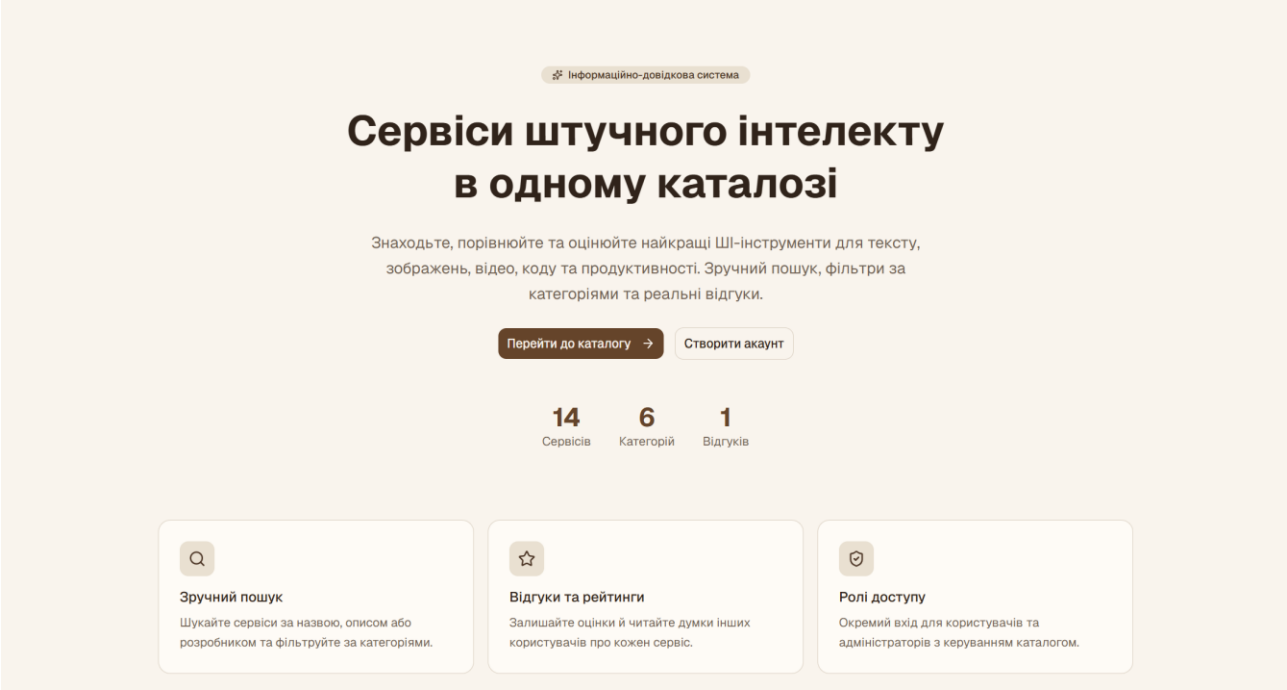


Рисунок Б.1 – Головна сторінка

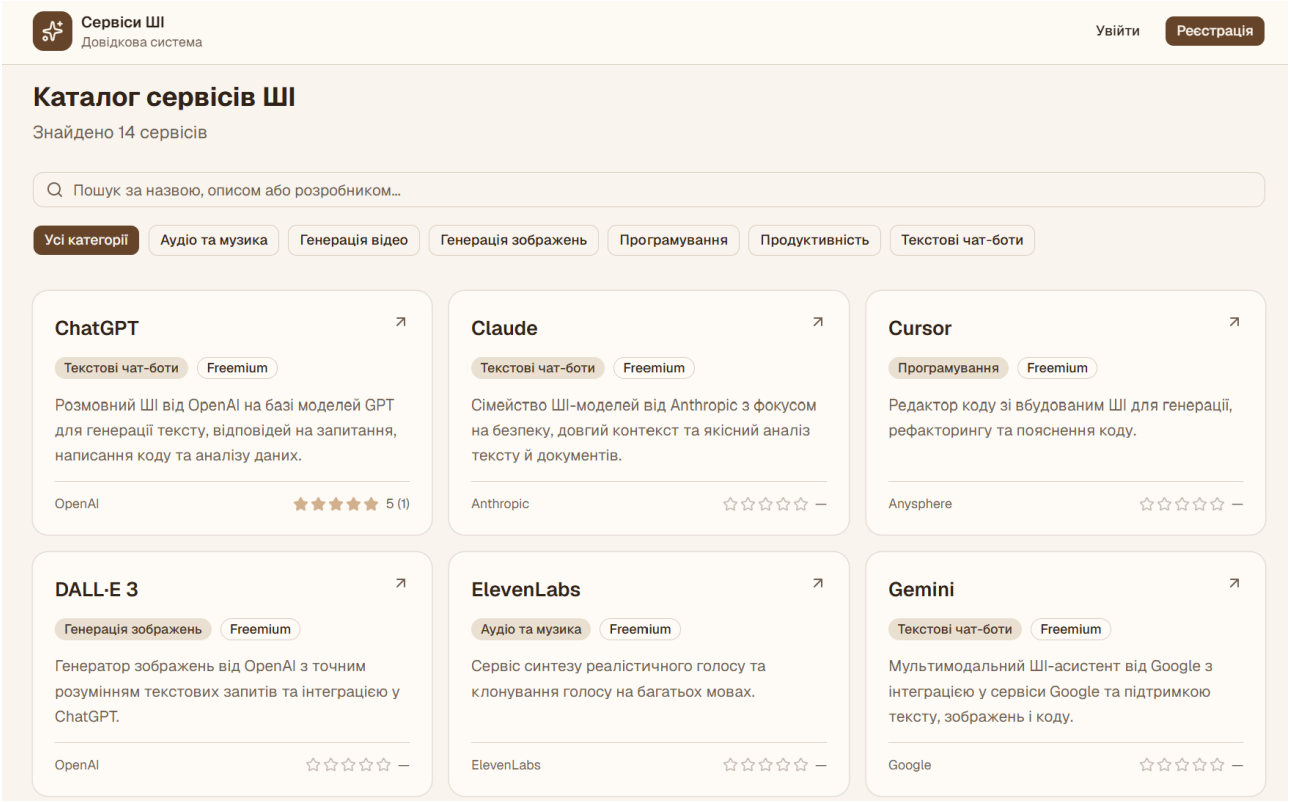


Рисунок Б.2 – Каталог

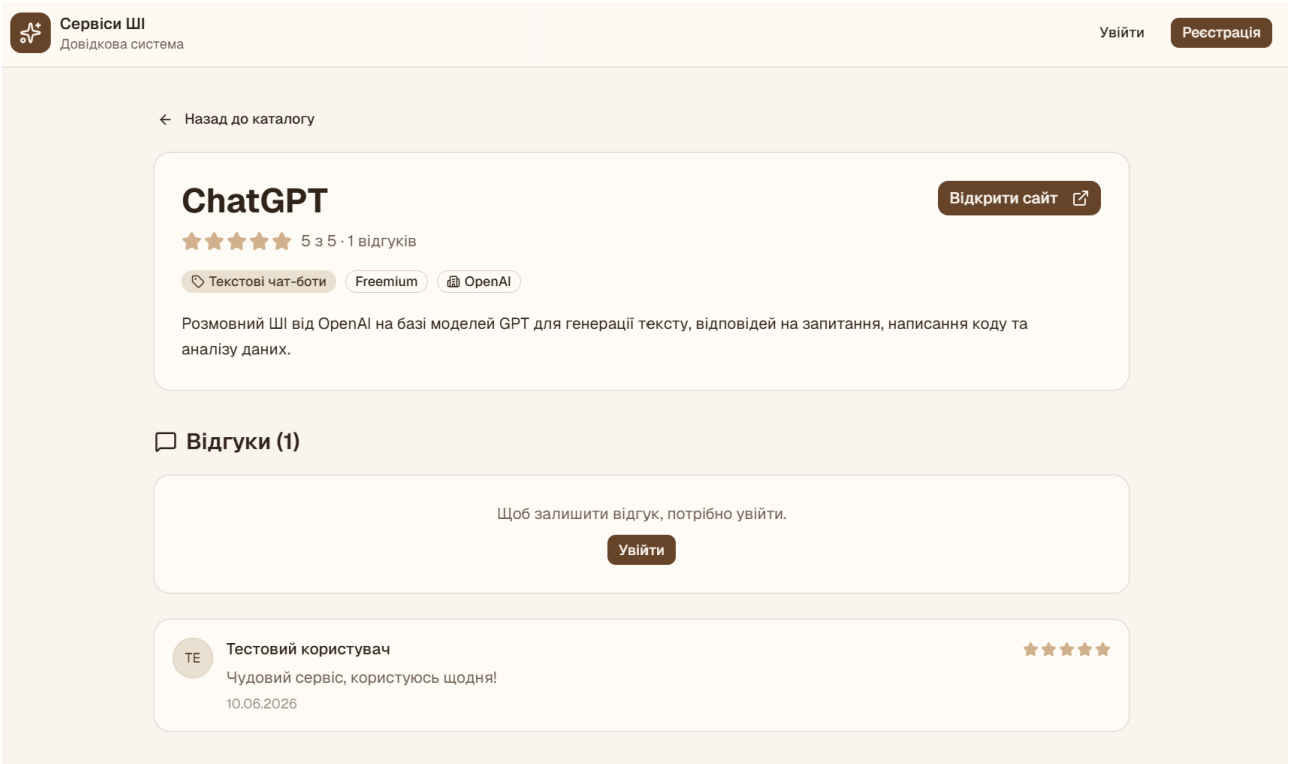


Рисунок Б.3 – Сторінка даних про ШІ

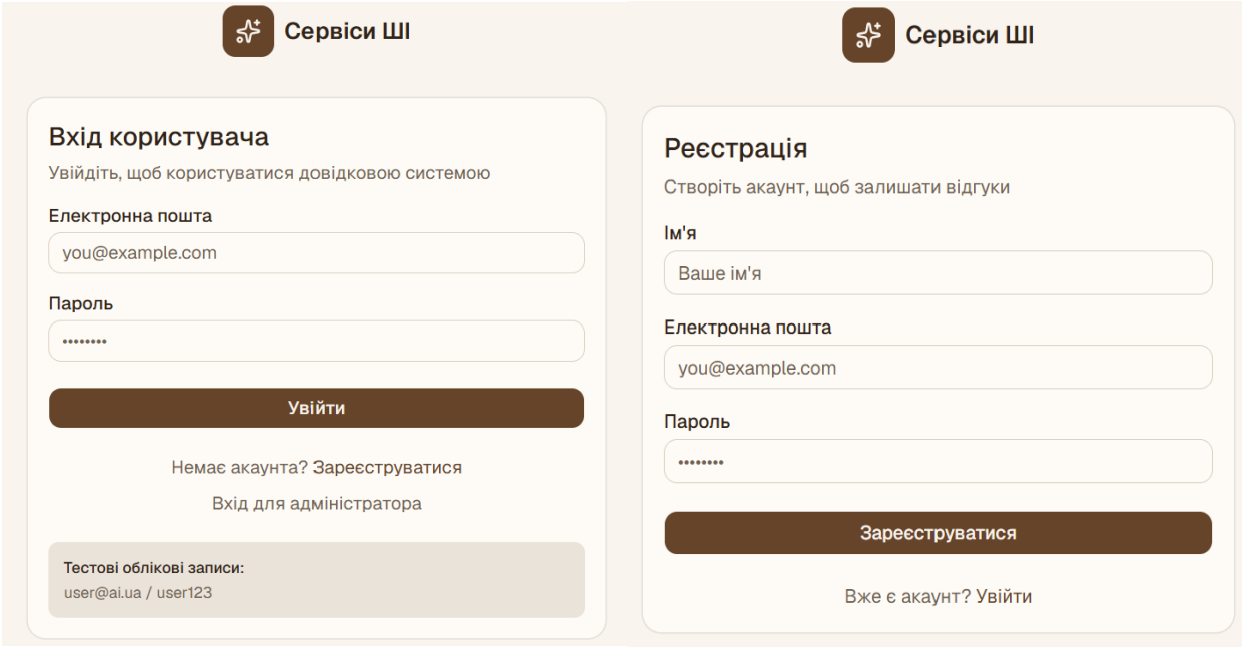


Рисунок Б.4 – Сторінка входу / реєстрації

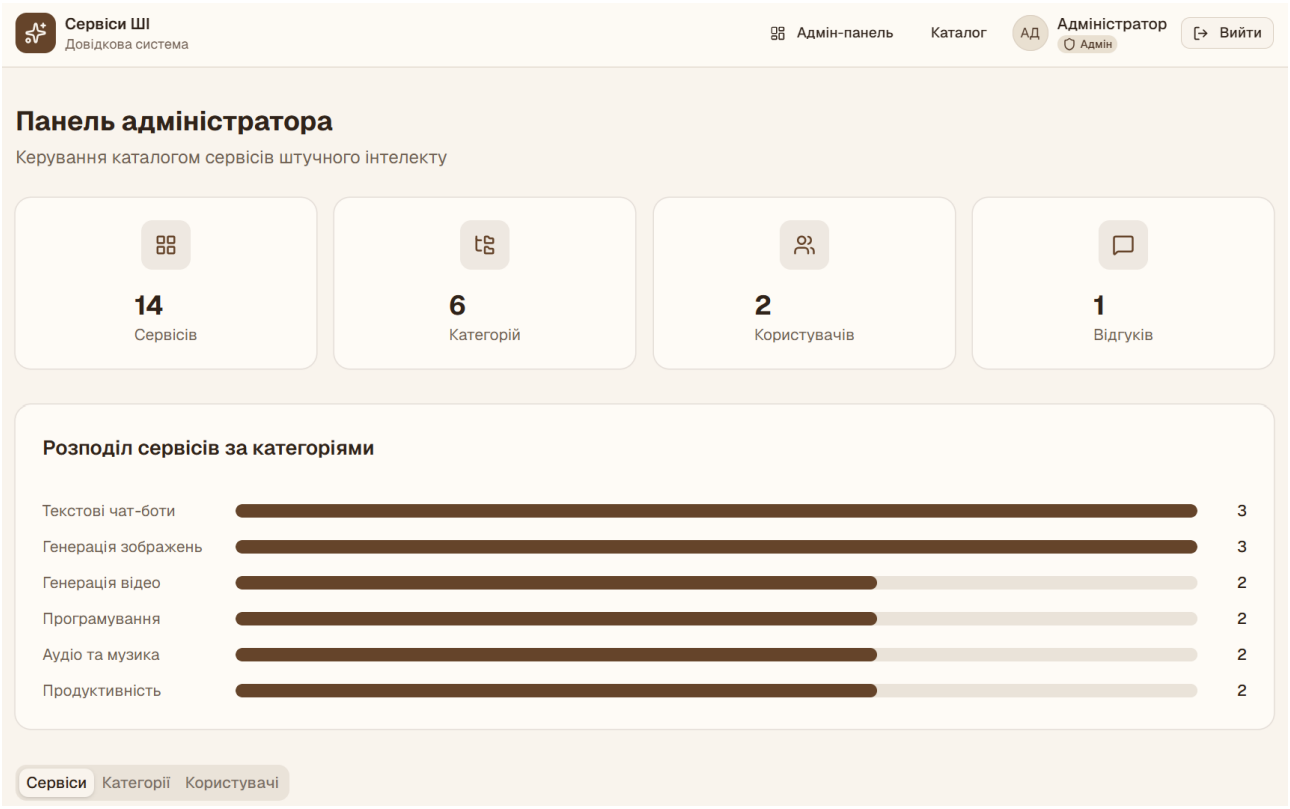


Рисунок Б.5 – Панель адміністратора

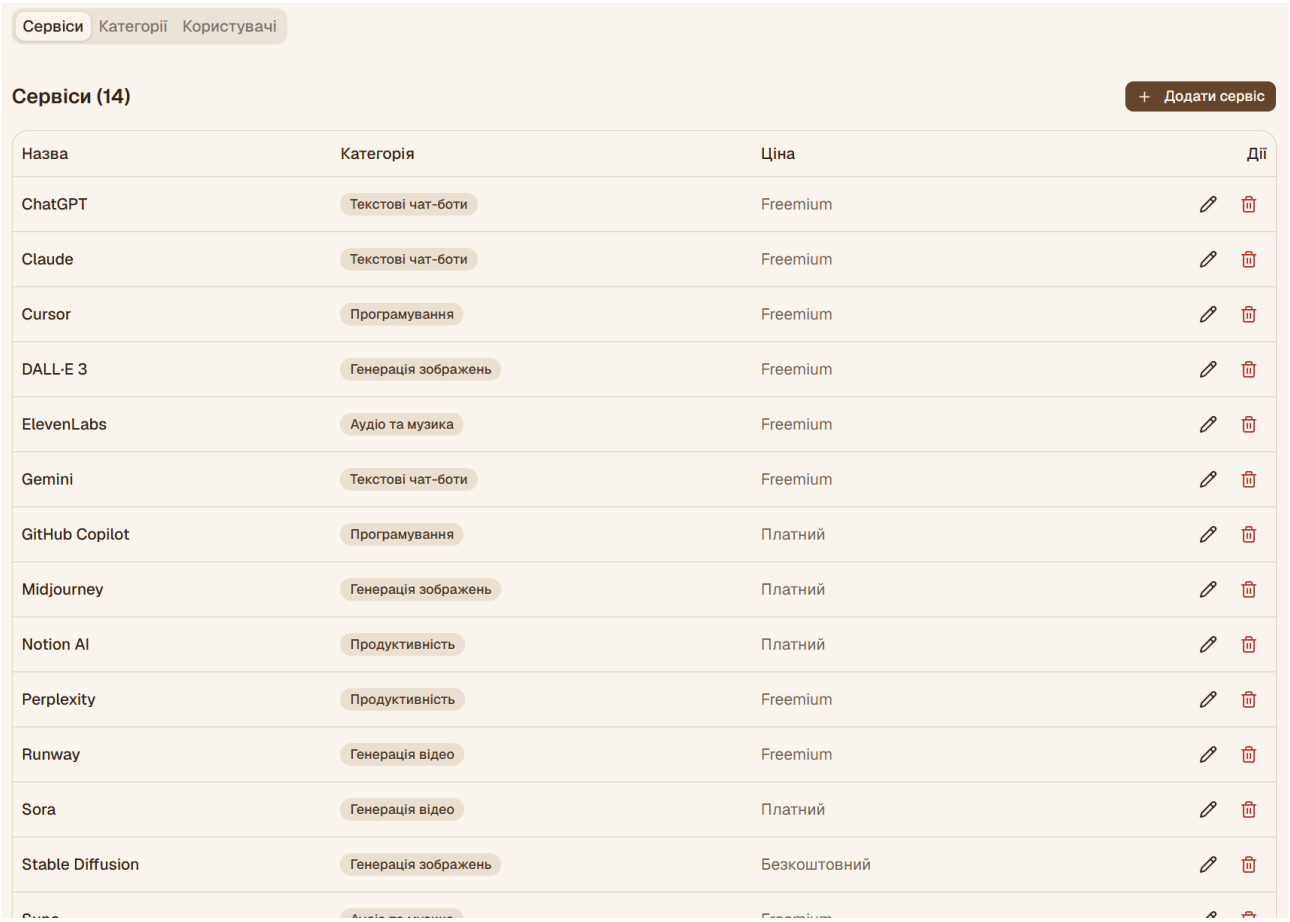


Рисунок Б.6 – Панель адміністратора (сервіси)

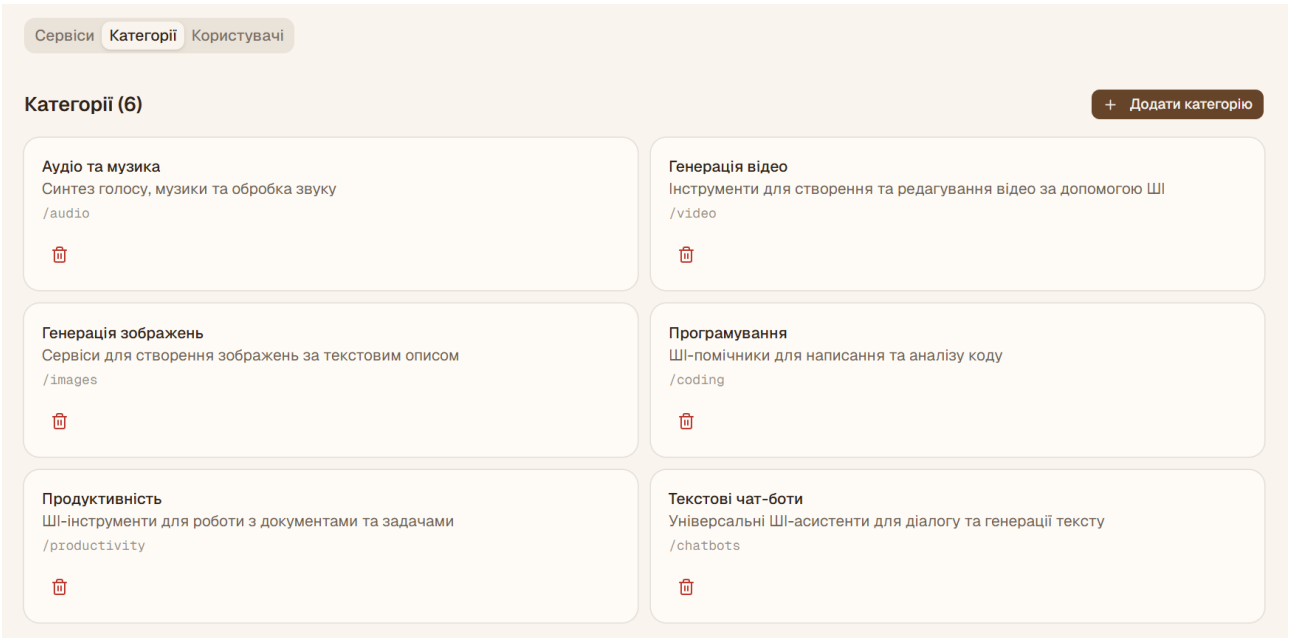


Рисунок Б.7 – Панель адміністратора (атегорії)

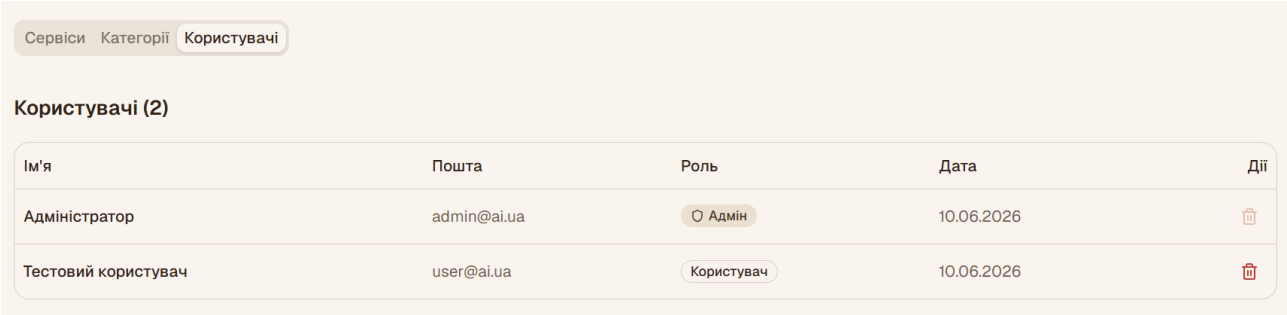


Рисунок Б.8 – Панель адміністратора (користувачі)