Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *126 «Інформаційні системи та технології»*
Шифр, назва

Освітня програма *«Інформаційні системи та технології»*
Назва

Шифр КвРІСТс 230154.23.01.03 ПЗ

Виконав здобувач III курсу, група ІСТс-23-1 _____ Дмитро КЛІМІШЕН
Підпис Ініціали, прізвище

Керівник _____ Ольга ПАВЛОВА
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. Підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 126 Інформаційні системи та технології

Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Клімішену Дмитру Віталійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків
Керівник проекту (роботи) Павлова Ольга Олександрівна, д.ф., доцент.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2026 р. № 5
2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.
3. Вихідні дані до роботи Завдання на кваліфікаційну роботу
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметної області та технологій візуалізації відкритих медичних даних системи openFDA, постановка задачі на розробку
Проєктування архітектури, алгоритмічного забезпечення та інтерфейсу веб-орієнтованої інформаційної системи порівняльного аналізу даних
Програмна реалізація та апробація веб-орієнтованої інформаційної системи відображення статистичних даних у вигляді графіків
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Діаграма послідовності для модуля drug comparison
Концептуальна er-діаграма предметної області відкритих медичних даних системи FAERS
Візуалізація результатів програмної реалізації інформаційної системи

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання «_10_» 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір компонентів для проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	01.04.2026	виконано
5	Робота над розділом 3 – проєктування системи адаптивного застосування моніторингових елементів розвідувального БПЛА	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Дмитро КЛІМІШЕН
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТс 230154.23.01.03 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРІСТс 230154.23.01.03 Е8	Функціональна модель та структура	1		
3		КвРІСТс 230154.23.01.03 Е8	Компонентна архітектура веб-орієнтованої інформаційної системи	1		
4		КвРІСТс 230154.23.01.03 Е8	Поведінкові моделі асинхронної взаємодії з відкритим арі	1		

Зм	Арк	№ докум	Підпис	Дата	КвРІСТс 230154.23.01.03 ВП		
Розробив		Клімішен		01.06	Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків	Літера	Аркуш	Аркушів
Перевір.		Павлова		01.06		У	1	1
Н. контр.		Лисенко				ХНУ, ІСТс-23-1		
Затв.		Павлова		01.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків».

Автор роботи: Дмитро КЛІМІШЕН.

Керівник роботи: Ольга ПАВЛОВА.

Пояснювальна записка: 65 с., 7 рис., 2 табл., 4 дод., 40 джерел.

Графічна частина: 3 креслення.

АНАЛІЗ ДАНИХ, АРХІТЕКТУРА, ВЕБ-ОРІЄНТОВАНА СИСТЕМА, ВІДКРИТІ ДАНІ, ВІЗУАЛІЗАЦІЯ, МОНІТОРИНГ, ФАРМАКОНАГЛЯД.

Кваліфікаційна робота бакалавра присвячена розробці веб-орієнтованої інформаційної системи для інтерактивного відображення та аналізу статистичних даних безпеки лікарських засобів на основі відкритих реєстрів openFDA. Актуальність теми зумовлена необхідністю швидкої інтерпретації великих масивів неструктурованих медичних даних для оперативного прийняття клінічних рішень. Метою роботи є проєктування та програмна реалізація високоефективного вебзастосунку, який забезпечує візуалізацію та паралельне порівняння профілів безпеки препаратів. У процесі виконання проаналізовано предметну область, спроєктовано архітектуру безсерверного агрегатора та розроблено алгоритмічне забезпечення з використанням стека (Next.js, React 19, TypeScript, Tailwind CSS). Система успішно пройшла апробацію та готова до практичного використання.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Сучасний етап розвитку глобальної системи охорони здоров'я характеризується безпрецедентними темпами цифрової трансформації та інституціоналізацією парадигми відкритих медичних даних. Глобальні регуляторні органи поступово відмовляються від монополізації статистичної інформації, надаючи незалежним інженерам та клініцистам доступ до гігантських урядових реєстрів. Ініціатива openFDA, реалізована Управлінням з продовольства і медикаментів США, є фундаментальним зрушенням у сфері фармаконагляду [1] Відкриття доступу до Системи повідомлень про побічні ефекти через машиночитані програмні інтерфейси створило потужний технологічний базис для створення інноваційних інструментів моніторингу. Зміна формату розповсюдження даних усуває необхідність ручного завантаження статичних звітів, дозволяючи отримувати актуальні зрізи клінічної статистики в режимі реального часу.

Стрімке зростання обсягів доступної медичної інформації неминуче провокує виникнення феномену перевантаження даними. Фахівці з фармаконагляду щоденно стикаються з колосальними масивами неструктурованих або глибоко вкладених JSON-об'єктів [2], які вкрай важко піддаються інтерпретації без спеціалізованого програмного забезпечення. Робота з «сирими» результатами запитів до відкритого API вимагає невиправдано високих когнітивних зусиль, оскільки текстовий формат приховує неочевидні статистичні патерни, демографічні вразливості та критичні маркери небезпеки лікарських засобів. Наявні на ринку державні реєстри часто пропонують технічно перевантажені та архаїчні інтерфейси, які не здатні забезпечити високу швидкість агрегації багатовимірних вибірок. Комерційні аналоги надмірно спрощують клінічну картину, приховуючи первинну статистику за узагальнювальними текстами. Виявлена проблематика чітко артикулює гостру потребу у розробці легких, вузькоспеціалізованих

вебзастосунків із потужним фокусом на інтерактивну візуалізацію та паралельний порівняльний аналіз.

Метою кваліфікаційної роботи є проєктування та програмна реалізація високоефективного вебзастосунку для аналізу та інтерактивного порівняння профілів безпеки лікарських засобів на основі масивів відкритих медичних даних. Досягнення визначеної мети спирається на використання сучасного технологічного стека, що включає фреймворк Next.js із застосуванням архітектури App Router, бібліотеку React 19, строгу типізацію TypeScript та утилітарний рушій стилізації Tailwind CSS 4. Синергія цих технологій має забезпечити високу швидкість обробки клінічної інформації та створення бездоганного користувацького досвіду для фахівців медичної галузі.

Об’єктом дослідження виступає процес аналізу, агрегації та візуалізації великих масивів відкритих медичних даних в умовах сучасного фармаконагляду.

Предметом дослідження є методи, алгоритми та програмні засоби інтерактивної візуалізації, а також механізми паралельного порівняння частотних характеристик побічних ефектів лікарських засобів на основі відкритих реєстрів Управління з продовольства і медикаментів США.

Реалізація заявленої мети вимагає послідовного виконання комплексу логічно пов'язаних науково-практичних завдань. Початковим етапом роботи є проведення глибокого аналізу предметної області, що включає дослідження світових тенденцій розвитку ініціатив відкритого доступу в медицині та вивчення технологічної екосистеми openFDA. Цей етап також охоплює порівняльний аналіз існуючих рішень для моніторингу безпеки препаратів з метою виявлення їхніх архітектурних та функціональних недоліків. Наступний крок передбачає технічне обґрунтування вибору програмного стека для побудови високопродуктивних клінічних інтерфейсів, здатних витримувати навантаження при обробці гетерогенних JSON-відповідей.

Подальша робота фокусується на системному проєктуванні інформаційної архітектури застосунку. Ця стадія включає формування суворих функціональних

вимог, аналіз потреб цільової аудиторії та розробку сценаріїв використання для гіпотетичного дослідника даних. Обов'язковою складовою цього етапу є візуалізація поведінкових патернів через побудову UML-моделей, зокрема діаграм прецедентів та діаграм послідовності для складних асинхронних транзакцій. Наступне завдання стосується безпосередньої розробки алгоритмічного забезпечення системи. Критичним викликом на цій стадії є оптимізація процесу пошуку статистичних перетинів між двома медикаментами, що вимагає переходу від наївних методів обчислення до використання механізмів на основі хеш-таблиць із лінійною часовою складністю обчислень.

Фінальна фаза дослідження присвячена практичній програмній реалізації вебзастосунку. Цей комплекс завдань охоплює ініціалізацію середовища розробки, налаштування модулів безпечної мережевої взаємодії з відкритим API, розробку декларативних інтерфейсів для валідації масивів даних та побудову інтерактивних графічних панелей. Завершується етап розробки проведенням технічної апробації створеного продукту, валідацією стійкості алгоритмів до аномальних структур даних та оцінкою загальної продуктивності платформи в умовах, наближених до реальної експлуатації.

Методологічна основа дослідження базується на комплексному застосуванні методів системного аналізу для формування вимог до програмного продукту. Проєктування архітектури системи спирається на принципи об'єктно-орієнтованого аналізу та компонентного підходу до розробки користувацьких інтерфейсів. Для вирішення проблем продуктивності клієнтської частини активно застосовуються методи алгоритмічної оптимізації та теорії структур даних.

Наукова новизна одержаних результатів полягає у вдосконаленні методів клієнтської обробки та візуалізації неструктурованих медичних масивів. Вперше запропоновано та реалізовано інтегровану архітектурну модель, яка поєднує можливості серверних компонентів Next.js для попередньої маршрутизації зі спеціалізованим алгоритмом обчислення перетинів симптомів безпосередньо у

браузері користувача. Впровадження оптимізованого методу зіставлення профілів безпеки медикаментів усуває необхідність використання важких серверних баз даних для проміжних обчислень, гарантуючи миттєву побудову паралельних порівняльних гістограм.

Практичне значення отриманих результатів визначається створенням повністю функціональної вебплатформи, готової до використання спеціалістами з фармаконагляду, медичними статистиками та практикуючими клініцистами. Розроблений інструмент радикально знижує когнітивне навантаження під час аналізу урядових звітів, трансформуючи сирі числові ідентифікатори у зрозумілі інтерактивні панелі демографічних ризиків. Впровадження динамічного модуля порівняння лікарських засобів дозволяє фахівцям за лічені секунди здійснювати об'єктивну оцінку альтернативних курсів лікування. Програмний продукт мінімізує вплив людського фактора при обробці статистичних даних, підвищуючи загальну швидкість та точність прийняття критичних клінічних рішень. Спроєктована архітектура застосунку відрізняється високим рівнем відмовостійкості та може слугувати надійним технологічним шаблоном для створення інших медичних аналітичних систем на базі відкритих програмних інтерфейсів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ ВІЗУАЛІЗАЦІЇ МЕДИЧНИХ ДАНИХ

1.1 Концепція Open Data та її роль у цифровій трансформації медицини

Глобальна інфраструктура охорони здоров'я наразі проходить етап інтенсивного переходу від ізольованих локальних сховищ до інтероперабельних екосистем. Концепція відкритого доступу до медичних даних формує нову фундаментальну базу для досліджень у сфері фармаконагляду. Історично інформація про побічні реакції на лікарські засоби зберігалася у закритих державних реєстрах або поширювалася виключно у вигляді періодичних статичних звітів. Сучасний підхід докорінно змінює цю модель, пропонуючи надання машиночитаних даних через відкриті програмні інтерфейси в режимі реального часу.

Аналіз світових ініціатив демонструє, що провідні регулятори поступово відмовляються від монополізації аналітики. Проєкт openFDA від Управління з продовольства і медикаментів США є одним із найбільш показових прикладів такої технологічної трансформації. До появи подібних ініціатив дослідники були змушені вручну завантажувати об'ємні дампи баз даних, які вимагали складного парсингу та розгортання власних серверних рішень для виконання базових пошукових запитів. Сьогодні архітектура взаємодії будується на принципах RESTful API [4]. Це дозволяє розробникам безпосередньо інтегрувати актуальні масиви інформації у вебплатформи, минаючи етап проміжних баз даних.

Розширення доступу до медичної статистики не обмежується лише фактом її публікації. Критичним фактором успіху концепції Open Data є стандартизація форматів обміну. Використання структурованих JSON-відповідей замість застарілих форматів експорту суттєво спрощує програмну інтеграцію. Клієнтські застосунки отримують можливість безпосередньо мапити ієрархічні об'єкти на внутрішні моделі даних. У контексті використання сучасних фреймворків строга

типізація таких відповідей дозволяє перенести виявлення помилок структури даних на етап компіляції коду.

Динаміка відкриття медичних даних жорстко контролюється вимогами до їх деідентифікації. Публічні датасети, такі як масиви повідомлень про побічні ефекти, проходять складний етап агрегації та повного видалення персональної інформації пацієнтів. Завдяки цьому механізму інженери та дослідники отримують доступ до репрезентативних вибірок, які включають точні демографічні показники, дозування препаратів та класифікацію реакцій за словником MedDRA [6]. При цьому повністю зберігається відповідність жорстким стандартам конфіденційності.

Інституціоналізація відкритих даних стимулює стрімкий розвиток незалежних аналітичних інструментів. Дослідницькі центри, незалежні фахівці та IT-компанії отримали можливість створювати спеціалізовані клієнтські інтерфейси поверх гігантських державних сховищ. Це усуває обмеження в аналітиці в аналітиці, коли швидкість обробки медичної інформації залежала виключно від обчислювальних ресурсів та пріоритетів самого регулятора.

Масиви сирих медичних даних у форматі JSON самі по собі не несуть прямої практичної цінності для кінцевого користувача, будь то клініцист або спеціаліст з безпеки ліків. Їх читання та інтерпретація вимагають невиправдано високих когнітивних зусиль. Вплив візуалізації даних на процеси аналізу у дослідницькій практиці є ключовим фактором ефективності сучасних медичних інформаційних систем. Візуальний рівень абстракції перетворює тисячі рядків тексту та складні набори пар «ключ-значення» на зрозумілі патерни, неочевидні тренди та статистичні аномалії.

Оцінка швидкості прийняття рішень підтверджує, що графічне представлення статистичних розподілів кардинально скорочує час на ідентифікацію потенційних ризиків. Розгляд типового сценарію аналізу безпеки препарату демонструє цю різницю найбільш яскраво. Читання багатосторінкової таблиці з тисячами розрізнених записів про побічні ефекти займає години роботи

і супроводжується ризиком пропуску критичної інформації. Натомість побудова продуманого аналітичного дашборду дозволяє миттєво оцінити, наприклад, залежність ускладнень від віку пацієнта. Використання стовпчастих чи радіальних діаграм дає змогу фахівцю охопити загальну картину по препарату за лічені секунди.

Сучасні підходи до візуалізації медичної статистики категорично вимагають високого рівня інтерактивності. Статичні графіки, згенеровані на стороні сервера, поступово відходять у минуле. Інтерфейси нового покоління дозволяють користувачеві взаємодіяти з даними безперервно: фільтрувати метрики на льоту, ізолювати окремі вікові групи або отримувати детальну інформацію про конкретний симптом при наведенні курсора. Це створює оптимальне середовище для дослідника, де процес формування гіпотези та її перевірки відбувається майже одночасно.

Окремим і надзвичайно складним аспектом візуалізації є порівняльний аналіз препаратів. У фармаконагляді постійно виникає потреба зіставити профілі безпеки кількох діючих речовин для вибору оптимального курсу лікування. Текстове порівняння двох багатовимірних ієрархічних структур даних є неефективним і схильним до помилок. Візуальне накладання метрик, використання компонентів прогресу для відображення частоти спільних симптомів (Common Symptoms) та паралельні графіки підвищують об'єктивність оцінки альтернативних засобів.

Проєктування таких інтерфейсів вимагає жорсткої дисципліни щодо колірного кодування та мінімізації візуального шуму. Архітектура компонентів повинна зосереджувати увагу експерта виключно на даних, уникаючи зайвих декоративних елементів. Ергономіка аналітичної панелі безпосередньо корелює з імовірністю виявлення критичних сигналів небезпеки (safety signals) на ранніх етапах моніторингу. Що прозоріше репрезентовані дані з відкритого API, то нижча ймовірність хибної інтерпретації результатів людиною.

1.2 Технологічний огляд екосистеми openFDA

Базовим джерелом інформації для розроблюваної аналітичної платформи виступає кінцева точка openFDA, яка надає доступ до масивів Системи повідомлень про побічні ефекти (FAERS). Архітектура відповіді API на рівні JSON-структури концептуально розділена на два кореневі вузли. Перший вузол «meta» містить службову інформацію, включаючи параметри пагінації, загальну кількість знайдених записів та юридичні застереження регулятора щодо використання медичної статистики. Другий і головний вузол «results» представляє собою масив індивідуальних звітів про безпеку (ICSR). Залежно від типу відправленого запиту цей масив може повертати або повні багатовимірні об'єкти клінічних випадків, або агреговані метрики у форматі частотних словників, якщо застосовується параметр серверного групування.

Внутрішня ієрархія кожного повного звіту будується навколо центрального об'єкта «patient». Цей архітектурний підхід повністю відображає клінічну реальність фармаконагляду, де відправною точкою моніторингу є конкретний пацієнт, який зазнав небажаної медичної події під час курсу лікування. Структура передбачає відношення «один до багатьох». У межах одного зареєстрованого випадку людина може приймати одразу кілька лікарських засобів та мати цілий спектр побічних реакцій. Відповідно, вузол пацієнта містить два критично важливі масиви – «drug» та «reaction». З інженерної точки зору така багаторівнева вкладеність вимагає розробки спеціальних алгоритмів денормалізації на стороні клієнтського застосунку. Для побудови двовимірних графіків чи порівняльних матриць у модулі Drug Comparison необхідно програмно трансформувати складне дерево JSON у лінійні структури.

Аналіз демографічних параметрів, які використовуються для візуалізації вікових та гендерних груп ризику, суттєво ускладнюється специфікою типізації відкритих даних FDA. Поле «patientonsetage», яке фіксує вік людини на момент

виникнення ускладнення, передається сервером у форматі простого рядка, хоча його семантичне значення є виключно числовим. Безпосередня конвертація цього рядка у тип числа в JavaScript є недостатньою для точної аналітики. Значення віку набуває валідного статусу виключно у прив'язці до сусіднього поля «patientonsetageunit». Історично система FAERS використовує специфічні числові коди для позначення одиниць виміру часу. Код 801 відповідає рокам, 802 – місяцям, а 804 – дням.

Аналогічна ситуація спостерігається з ідентифікацією статі в полі «patientsex». Замість інтуїтивно зрозумілих текстових значень система застосовує цифрові ідентифікатори за міжнародною конвенцією E2B, де одиниця позначає чоловічу стать, а двійка – жіночу. Практична реалізація аналітичного інтерфейсу вимагає створення надійних локальних словників-декодерів для коректного мапінгу цих сирих кодів у текстові мітки для компонентів Recharts.

Блок інформації про лікарські засоби демонструє глибоку неоднорідність реальних медичних даних. Масив препаратів містить базове поле «medicinalproduct», куди записується найменування ліків безпосередньо зі слів репортера – лікаря, фармацевта або самого пацієнта. Це поле вкрай нестабільне. Воно часто містить орфографічні помилки, змішані назви діючих речовин із торговими марками, абревіатури та сторонній текст.

Для вирішення проблеми ідентифікації архітектура openFDA передбачає наявність додаткового вкладеного об'єкта «openfda», який виступає в ролі нормалізованого медичного довідника. У цьому вузлі система намагається співставити сирий текст із валідованими реєстрами та надає очищені масиви даних. Найбільшу цінність для розробки становлять масиви «brand_name» для торгової назви та «substance_name» для активної речовини. Використання саме цих нормалізованих ідентифікаторів є абсолютно критичною умовою для алгоритмів модуля Drug Comparison. Спиратися на сирі текстові введення репортерів неможливо через колосальний ризик дублювання та хибного розпізнавання медикаментів.

Клінічні наслідки прийому препаратів інкапсульовані у масиві реакцій. Основною аналітичною одиницею цього вузла виступає поле «reactionmeddrapt», яке містить суворо стандартизований термін за міжнародним медичним словником MedDRA. Саме цей параметр формує базис для побудови частотних діаграм та виявлення перетинів (Common Symptoms) при паралельному зіставленні двох медикаментів.

Проєктування інформаційної системи вимагає постійного врахування типових аномалій масивів даних FAERS. Далеко не кожен репорт містить повний набір атрибутів. Нормалізований вузол довідника може бути повністю відсутнім навіть за умови коректно заповненого поля назви препарату, а ключові демографічні показники часто залишаються порожніми (null). Крім того, наявність кількох препаратів у одному кейсі позбавляє дослідника можливості встановити прямий причинно-наслідковий зв'язок між конкретною пігулкою та конкретним симптомом на рівні одиничного запису. Аналітика безпеки спирається виключно на статистичну агрегацію та виявлення патернів на великих вибірках.

На рівні програмної реалізації всі перелічені виклики вирішуються шляхом впровадження суворої типізації TypeScript. Архітектура застосунку передбачає створення двох ізольованих шарів інтерфейсів. Перший шар точно відображає сиру структуру відповідей API з масовим використанням опціональних типів. Другий шар репрезентує фінальні прикладні моделі даних, куди об'єкти потрапляють лише після проходження етапів парсингу, застосування словників-декодерів та відсіювання невалідних репортів.

Взаємодія аналітичної платформи з масивами даних Системи повідомлень про побічні ефекти реалізується через RESTful архітектуру відкритого програмного інтерфейсу Управління з продовольства і медикаментів США. Базовою точкою входу для отримання інформації про небажані явища виступає спеціалізований ендпоінт «drug/event.json». Замість традиційних механізмів фільтрації через тіло запиту або складні GraphQL-схеми, екосистема openFDA

використовує підхід на основі рядків запиту (query parameters). Архітектура інтерфейсу базується на використанні кількох ключових параметрів, серед яких центральне місце посідають «search» для вибірки цільових репортів та «count» для виконання серверної агрегації. Управління обсягом даних та навігацією здійснюється за допомогою параметрів «limit» та «skip». Візуальну структуру формування такого мережевого запиту з його основними компонентами наведено на рисунку 1.1.

Рисунок 1.1 – Синтаксична структура HTTP-запиту до кінцевої точки програмного інтерфейсу openFDA

Синтаксис пошукових запитів відрізняється високим рівнем гнучкості та дозволяє виконувати глибоку фільтрацію за будь-яким вузлом ієрархічної JSON-моделі. Базовий формат передбачає використання конструкції «поле:значення», де цільовий параметр вказується у вигляді точкової нотації, наприклад «patient.reaction.reactionmeddrapt». Формування складних критеріїв вибірки реалізується шляхом об'єднання базових умов логічними операторами. Система підтримує використання операторів кон'юнкції та диз'юнкції, які в URL-кодуванні передаються як «+AND+» та «+OR+». Для точного пошуку багатослівних термінів застосовується екранування подвійними лапками, тоді як діапазонні запити, такі як фільтрація за віком пацієнта або датою отримання репорту, конструюються за допомогою оператора «TO» у квадратних дужках.

Отримання сирих записів клінічних випадків є ресурсомісткою операцією, яка в контексті розробки візуальних панелей використовується рідко. Аналітичні компоненти вебзастосунку здебільшого спираються на механізм агрегації,

котрий активується параметром «count». Замість масиву індивідуальних репортів сервер повертає набір статистичних сегментів (buckets), кожен з яких містить значення терміна та кількість його згадувань у відфільтрованій вибірці. Критичним архітектурним нюансом при роботі з текстовими полями є використання суфікса «.exact». За замовчуванням пошуковий рушій openFDA токенізує рядки, розбиваючи складні медичні діагнози на окремі слова. Додавання вказаного суфікса гарантує, що серверна логіка рахуватиме частоту повного терміна як єдиної неподільної сутності. Такий підхід є абсолютно необхідним для формування достовірних гістограм частоти побічних реакцій.

Проєктування інформаційної системи вимагає постійного врахування жорстких технічних обмежень API щодо обсягу вибірки. Максимальний розмір однієї сторінки відповідей обмежений тисячею записів, тоді як параметр зміщення дозволяє заглибитися не більше ніж на двадцять п'ять тисяч позицій. Подолання цього бар'єра вимагає переходу на механізм глибокої пагінації «search_after». Стабільність роботи клієнтського застосунку в умовах продакшену також залежить від дотримання квот на частоту звернень. Використання авторизаційного ключа суттєво розширює добові ліміти, проте фронтенд-архітектура все одно повинна передбачати патерни дедуплікації викликів, кешування результатів та обмеження частоти виконання функцій (debounce) під час введення тексту в поля пошуку.

Обробка відповідей сервера становить окремий рівень складності через специфічне трактування кодів стану HTTP в екосистемі відкритого API. Успішна транзакція повертає код 200 та масив з результатами вибірки. Головна специфіка openFDA полягає у тому, що відсутність збігів за заданими критеріями не призводить до повернення порожнього масиву, як це заведено в класичних REST-архітектурах. Натомість система генерує помилку з кодом 404 та статусом «NOT_FOUND». На рівні клієнтської логіки цей статус категорично не можна трактувати як збій мережі або падіння сервера. Це очікувана легітимна відповідь,

яка просто сигналізує про відсутність зареєстрованих побічних ефектів для введених параметрів.

Програмна архітектура модуля Drug Comparison вимагає одночасного виконання кількох незалежних мережевих запитів для формування порівняльної матриці двох препаратів. Використання стандартного методу Promise.all у цьому контексті створює серйозний ризик каскадної відмови. Якщо статистика за одним із препаратів відсутня і сервер повертає 404, загальний проміс миттєво відхиляється, блокуючи рендер усієї аналітичної панелі. Для нівелювання цієї проблеми застосунок реалізує обробку запитів за допомогою конструкції Promise.allSettled [10]. Цей підхід дозволяє повністю ізолювати результати паралельних звернень до API. Навіть якщо пошук даних щодо другого медикаменту завершується статусом відхилення, інтерфейс успішно рендериться, відображаючи побудовані графіки для першого препарату та відповідне інформаційне повідомлення (Empty State) для другого.

Комплексна інтеграція з відкритими медичними даними реалізується через створення надійної проміжної абстракції у вигляді спеціалізованого конструктора запитів. Цей програмний шар інкапсулює логіку типізованого складання URL, автоматичне кодування спеціальних символів та обробку мережевих помилок. Завдяки такому розділенню відповідальності React-компоненти отримують очищені та валідовані TypeScript-інтерфейси, що кардинально знижує когнітивну складність кодової бази та забезпечує передбачувану поведінку візуальних елементів.

1.3 Порівняльний аналіз наявних рішень для моніторингу безпеки лікарських засобів

Ринок інформаційних систем у сфері охорони здоров'я пропонує низку інструментів для моніторингу побічних реакцій. Найбільш репрезентативним еталоном у цій ніші виступає офіційний сервіс FAERS Public Dashboard,

розроблений Управлінням з продовольства і медикаментів США. Головна перевага цієї платформи полягає у прямому доступі до первинного масиву неагрегованих даних регулятора. Дослідники отримують можливість працювати з мільйонами клінічних звітів, згенерованих безпосередньо медичними працівниками та пацієнтами. Водночас архітектурний підхід та інтерфейсні рішення цієї системи мають суттєві обмеження для динамічної роботи. Інтерфейс користувача перевантажений надлишковими елементами управління, що створює високий когнітивний бар'єр для нових спеціалістів. Швидкість рендерингу складних графіків відчутно падає при спробі обробити великі вибірки або застосувати багатовимірні фільтри до масивів JSON. Найбільш критичним недоліком для повсякденної роботи дослідника є відсутність інтуїтивного інструменту для паралельного порівняння двох препаратів. Фахівці змушені відкривати кілька вкладок браузера, вивантажувати статистику в окремі файли та вручну зіставляти частоту симптомів, що категорично суперечить принципам сучасного проєктування аналітичних систем.

Альтернативним інструментом глобального масштабу є база даних VigiAccess, яка підтримується Всесвітньою організацією охорони здоров'я. Ця платформа акумулює інформацію з національних центрів фармаконагляду десятків країн, забезпечуючи безпрецедентне географічне охоплення клінічних випадків. З інженерної точки зору платформа функціонує як повністю закрита екосистема. Відсутність публічного програмного інтерфейсу унеможливлює інтеграцію цих даних у сторонні аналітичні застосунки чи автоматизацію ETL-процесів. Візуалізація статистичної інформації реалізована у вигляді жорстко заданих таблиць та базових ієрархічних списків, що спираються на класифікатор MedDRA. Кінцевий користувач не має можливості динамічно фільтрувати масиви за складною комбінацією факторів – наприклад, швидко ізолювати групу літніх пацієнтів у межах обраної серйозної побічної реакції. Така технологічна ригідність перетворює VigiAccess на інструмент для загального ознайомлення, а не для проведення глибокого інтерактивного аналізу.

Комерційний сегмент медичної аналітики представлений популярними порталами формату Drugs.com або WebMD. Зазначені ресурси демонструють кардинально інший підхід до побудови інформаційної архітектури. Вони орієнтовані на масового споживача, тому відрізняються продуманою ергономікою, високою швидкістю завантаження сторінок та чуйним дизайном. Інформація про побічні ефекти тут подається у максимально спрощеному текстовому форматі. Глибока кількісна аналітика, побудована на реальних клінічних звітах, зазвичай відсутня або навмисно прихована за узагальнювальними категоріями на кшталт «дуже часто», «рідко» або «в окремих випадках». Такий підхід повністю нівелює можливість точного статистичного зіставлення. Комерційні платформи рідко розкривають повну методологію агрегації своїх даних, що робить їхні висновки непридатними для використання у верифікованих клінічних дослідженнях. Зведену характеристику розглянутих інформаційних систем та розроблюваної платформи наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика систем моніторингу безпеки лікарських засобів

Критерії порівняння	FAERS Public Dashboard	VigiAccess (ВООЗ)	Drugs.com / WebMD	Розроблювана система (Drug Safety Analytics)
Джерело та доступність даних	Відкриті урядові дані FDA	Глобальні дані ВООЗ (закрита екосистема, без API)	Комерційні бази (прихована методологія)	Відкриті урядові дані через RESTful API openFDA
Ергономіка інтерфейсу	Перевантажений, високий когнітивний бар'єр	Базовий (жорсткі списки та таблиці)	Висока (орієнтація на масового споживача)	Мінімалістичний компонентний дизайн (shadcn/ui)

Кінець таблиці 1.1

Швидкодія та рендеринг	Низька при роботі з великими вибірками	Стабільна, але без динамічних масивів	Висока швидкість завантаження сторінок	Висока (Server Components Next.js + Recharts)
Глибока інтерактивна фільтрація	Присутня, але ускладнена	Відсутня (немає крос-фільтрації)	Відсутня (узагальнені текстові категорії)	Присутня (динамічне сортування за віком, статтю)
Паралельне порівняння препаратів	Відсутнє (вимагає ручного вивантаження)	Відсутнє	Відсутнє у кількісному вигляді	Присутнє (автоматизований пошук спільних симптомів)
Цільове призначення	Складний клінічний аналіз	Загальне ознайомлення зі статистикою	Спрощене інформування пацієнтів	Швидка перевірка клінічних гіпотез дослідниками

Аналіз наявних програмних рішень чітко окреслює технологічний розрив між надскладними, але повільними державними реєстрами та зручними, але аналітично поверхневими комерційними сайтами. Вирішення цієї проблеми вимагає розробки легкого, швидкого вебзастосунку з акцентом на інтерактивну візуалізацію. Використання фреймворку Next.js у зв'язці з архітектурою App Router дозволяє перенести частину обчислювального навантаження на сторону сервера, оптимізуючи доставку контенту. Застосування компонентного підходу React 19 створює ідеальне середовище для розробки ізольованих аналітичних модулів, які незалежно оновлюють свій стан без перезавантаження всієї сторінки. Інтеграція бібліотеки Recharts розв'язує проблему повільного

рендерингу статистичних масивів, трансформуючи сирі агрегації openFDA у чуйні графіки розподілу вікових та гендерних ризиків.

Ключовою функціональною перевагою запропонованої архітектури є впровадження спеціалізованого модуля Drug Comparison. На відміну від монолітних інтерфейсів наявних баз даних, розроблюваний застосунок використовує асинхронні патерни для генерації паралельних мережевих запитів. Це дозволяє миттєво отримувати зрізи даних по двох різних діючих речовинах та візуалізувати перетини їхніх симптомів (Common Symptoms) в єдиному аналітичному просторі. Комбінація суворої типізації TypeScript та сучасної системи стилізації Tailwind CSS 4 гарантує високу стабільність кодової бази і дозволяє реалізувати мінімалістичний інтерфейс на базі компонентів shadcn/ui. Розроблена платформа фокусується виключно на інструментарії дослідника, надаючи механізми для швидкої перевірки клінічних гіпотез при повному збереженні прозорості та достовірності відкритих урядових даних.

1.4 Обґрунтування вибору технологічного стека для розробки аналітичної платформи

Еволюція фронтенд-розробки призвела до суттєвого перегляду архітектурних патернів при створенні медичних інформаційних систем. Використання традиційних односторінкових застосунків (SPA) для візуалізації масивів відкритого API часто стикається з фундаментальними обмеженнями продуктивності. Завантаження повного обсягу логіки маршрутизації, парсингу JSON та рендерингу безпосередньо у браузер клієнта створює невиправдане навантаження на обчислювальні ресурси пристрою. Вибір екосистеми Next.js із застосуванням сучасної моделі App Router докорінно змінює цю парадигму, переносячи центр обчислювальної ваги на серверну інфраструктуру. React 19 виступає ключовим рушієм цієї трансформації, забезпечуючи нативну підтримку нових механізмів генерації сторінок. Це дозволяє створювати гібридні

інтерфейси, де важкі математичні агрегації ізольовані від клієнтського середовища.

Ключовою інновацією обраного технологічного стека є чітка архітектурна дихотомія між серверними (React Server Components, RSC) та клієнтськими компонентами. Серверні компоненти виконуються виключно на бекенді під час формування запиту, генеруючи готовий статичний HTML-код. У контексті розроблюваної аналітичної платформи базова розмітка сторінок, глобальна конфігурація Layout та безпосередні мережеві запити до ендпоінтів openFDA інкапсульовані саме в RSC. Завдяки цьому об'ємні бібліотеки для форматування дат чи глибокої обробки відповідей сервера не потрапляють до фінального збірочного файлу (бандла), що передається мережею. Натомість інтерактивні елементи системи вимагають доступу до браузерних програмних інтерфейсів для маніпуляцій з об'єктною моделлю документа (DOM) та SVG-графікою. Бібліотека Recharts фізично не здатна функціонувати у суто серверному середовищі. Застосування директиви «use client» на рівні компонентів діаграм дозволяє розмежувати зону відповідальності: сервер миттєво віддає загальний каркас панелі Dashboard, а браузер завантажує мінімально необхідний код лише для забезпечення інтерактивності статистичних зрізів.

Реалізація аналітичної панелі вимагала інтеграції надійної бібліотеки для побудови графіків, здатної ефективно обробляти динамічні масиви медичних даних. На етапі проєктування було розглянуто три найпопулярніші рішення в екосистемі JavaScript: низькорівневу бібліотеку D3.js, рішення на базі Canvas – Chart.js, та декларативну React-орієнтовану бібліотеку Recharts. Кожна з цих технологій володіє унікальною специфікою рендерингу та відрізняється за складністю інтеграції в сучасні компонентні фреймворки. Глибокий інженерний аналіз їхніх можливостей є обов'язковим кроком для запобігання проблемам із продуктивністю на етапі масштабування застосунку. Зведену порівняльну характеристику цих інструментів наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика бібліотек для візуалізації даних

Критерії порівняння	Recharts[12]	Chart.js[13]	D3.js[14]
Архітектурний підхід	Декларативний (React-компоненти)	Імперативний (конфігураційні об'єкти)	Низькорівневий (маніпуляція DOM)
Інтеграція з React / Next.js	Нативна (створена спеціально для React)	Потребує додаткових обгорток (react-chartjs-2)	Складна (конфлікт із віртуальним DOM React)
Крива навчання	Низька (інтуїтивний синтаксис JSX)	Середня	Дуже висока (вимагає глибокого розуміння SVG та математики)
Гнучкість та кастомізація	Висока (достатня для 95% бізнес-задач)	Середня (обмежена рамками Canvas)	Абсолютна (можливість створення будь-яких візуалізацій)
Механізм рендерингу	SVG (масштабується без втрати якості)	HTML5 Canvas (растрова графіка)	SVG, Canvas, HTML
Адаптивність (Responsive)	Вбудований компонент <ResponsiveContainer>	Підтримується з коробки	Вимагає ручного написання логіки перерахунку розмірів

Процес перетворення статичної розмітки на повноцінний інтерактивний застосунок спирається на механізм гідратації (hydration). Для аналітичного медичного інтерфейсу, де швидкість первинного доступу до інформації є критичним показником якості системи, цей механізм гарантує бездоганний користувацький досвід. Дослідник майже миттєво бачить початковий екран із загальною статистикою adverse events, оскільки рушій браузера просто відображає отриманий текстовий документ. Паралельно, у фоновому режимі, React 19 непомітно підвантажує клієнтські скрипти та «оживляє» інтерфейс, прив'язуючи обробники подій до кнопок перемикання вкладок чи інтерактивних сегментів кругових діаграм. Користувач не стикається з порожніми екранами завантаження, оскільки візуальне відображення випереджає ініціалізацію складної логіки. Такий підхід кардинально зменшує час до повної готовності сторінки, що особливо важливо при рендерингу глибоко вкладених структур даних на менш потужних офісних комп'ютерах.

Специфіка взаємодії з урядовими медичними реєстрами вимагає жорсткого контролю над інтенсивністю мережевого обміну. Відкрита інфраструктура FDA має суворі квоти на частоту звернень (rate limits), систематичне перевищення яких неминуче призводить до блокування доступу. Архітектура Next.js App Router пропонує багаторівневу систему агресивного кешування, яка елегантно розв'язує цю інженерну проблему. Механізм мемоізації запитів (Request Memoization) гарантує, що виклик функції fetch із однаковими параметрами пошуку в межах одного циклу генерації сторінки не створить дублюючих звернень до зовнішнього API. Крім того, глобальний Data Cache дозволяє зберігати результати важких агрегацій безпосередньо у файловій системі сервера між різними сесіями. Практична користь цієї технології полягає в тому, що навігація дослідника між режимами Dashboard та Drug Comparison не змушує систему щоразу «смикати» сервери FDA для отримання одних і тих самих метрик. Застосунок миттєво віддає закешовані фрагменти, суттєво

оптимізуючи мережевий трафік та забезпечуючи стабільну роботу платформи навіть при пікових навантаженнях.

Окремої уваги вимагає підхід до проєктування візуального шару аналітичної системи. Інтерфейси для клінічних досліджень диктують суворі вимоги: відсутність зайвого декоративного шуму, високий контраст тексту та максимальна концентрація на статистичних аномаліях. Інтеграція фреймворку Tailwind CSS 4 у поєднанні з колекцією компонентів shadcn/ui створює ідеальний інструментарій для розробки такого продукту. Перехід на четверту версію компілятора фундаментально змінив швидкість обробки стилів. Новий рушій аналізує проєкт на рівні абстрактного синтаксичного дерева та генерує виключно ті правила, які фактично залучені в компонентах. Традиційні підходи на кшталт методології BEM або CSS-in-JS часто призводять до експоненціального розростання таблиць стилів, що гальмує відмальовування. Натомість застосування атомарних утилітарних класів безпосередньо в JSX-коді дозволяє форматувати складні табличні масиви та панелі сповіщень (Safety Alert Banner) без написання жодного рядка кастомного CSS. Це радикально зменшує розмір стильового бандла. Вебзастосунок завантажується за лічені мілісекунди, а стандартизована дизайн-система гарантує бездоганну читабельність ієрархії побічних ефектів.

Інтеграція зовнішніх програмних інтерфейсів у вебзастосунки традиційно супроводжується ризиками, пов'язаними з динамічною природою класичного JavaScript. Отримання масштабних JSON-відповідей від серверів openFDA вимагає детермінованого підходу до обробки структур, оскільки найменша невідповідність очікуваного формату реальному здатна викликати каскадні збої в роботі інтерфейсу. У цьому контексті використання базових можливостей мови TypeScript із застосуванням універсального типу «any» є категорично неприпустимим архітектурним рішенням. Такий підхід створює лише ілюзію безпеки, повністю відключаючи механізми статичного аналізу та дозволяючи потенційно небезпечному коду безперешкодно досягати етапу виконання.

Відмова від уникнення типізації на користь суворих контрактів обміну даними стає не просто питанням стилю, а фундаментальною вимогою до надійності розроблюваної медичної платформи.

Специфіка архітектури відкритих реєстрів фармаконагляду полягає у глибокій вкладеності та структурній варіативності об'єктів. Створення суворих декларативних інтерфейсів для базових сутностей – таких як «patient», «reaction» та «drug» – формує надійний захисний бар'єр між зовнішнім непередбачуваним середовищем та внутрішньою бізнес-логікою застосунку. Процес розробки передбачає детальне описування кожного вузла очікуваної відповіді. Наприклад, масив медикаментів вимагає чіткої специфікації вкладеного нормалізованого об'єкта «openfda», який, своєю чергою, повинен містити масиви рядків для ідентифікаторів «brand_name» чи «substance_name». Якщо компілятор не має інформації про точну структуру та типи цих об'єктів, будь-яка спроба ітерації або математичної агрегації перетворюється на потенційну точку відмови.

Клінічна реальність збору інформації про побічні ефекти диктує неминучу фрагментарність кінцевих датасетів. Далеко не кожен збережений репорт містить вичерпний демографічний профіль особи, тому критичні ключові поля часто надходять від API у стані «undefined» або «null». Вирішення цієї проблеми на рівні архітектури досягається завдяки активному використанню вбудованих утиліт TypeScript (Utility Types) [17], зокрема «Partial» та «Pick». Застосування типу «Partial» до базової моделі пацієнта дозволяє елегантно описати ситуацію, коли параметри віку («patientonsetage») або статі («patientsex») фізично відсутні в отриманому об'єкті. Це змушує інженера ще на етапі написання алгоритмів впроваджувати обов'язкові логічні перевірки на існування властивості, унеможливлюючи сліпе копіювання неповноцінних структур у стан React-компонентів.

На етапі безпосереднього кодування сувора типізація радикально змінює підхід до формування модулів аналітики. Статичний аналізатор безперервно перевіряє відповідність написаного коду задекларованим інтерфейсам,

перетворюючи середовище розробки на проактивного інспектора. Механізми інтелектуального автодоповнення позбавляють дослідника необхідності постійно звірятися з об'ємною технічною документацією FDA. При розробці логіки пошуку перетинів у модулі Drug Comparison редактор коду автоматично пропонує доступні властивості для масиву реакцій, гарантуючи безпомилкове звернення до поля «reactionmeddrapt». Будь-яка спроба звернутися до неіснуючого параметра або передати об'єкт у функцію, яка очікує примітивний рядок, миттєво блокується на рівні компіляції, ще до запуску застосунку в браузері.

Практична цінність описаного підходу найповніше розкривається під час рендерингу складних інтерактивних діаграм. Бібліотека Recharts вимагає ідеально підготовлених, плоских та однорідних масивів даних для коректної побудови візуальних осей та сегментів. Потрапляння значення «null» замість очікуваного числа або невідомої структури у властивості графіка гарантовано провокує помилку під час виконання (runtime error), що призводить до повного руйнування візуальної панелі на екрані користувача. Типобезпечна архітектура гарантує, що сирі відповіді від API проходять через суворі типізовані функції-мапери (mappers). У цих проміжних вузлах усі пропуски заповнюються значеннями за замовчуванням, а формати конвертуються до єдиного стандарту. Завдяки цьому візуальні компоненти отримують виключно валідовану інформацію, забезпечуючи безперебійну роботу вебплатформи навіть при обробці найскладніших та найменш структурованих клінічних вибірок.

1.5 Висновки до першого розділу

Проведений аналіз предметної області засвідчив незворотність переходу глобальної системи охорони здоров'я до концепції відкритих даних. Ініціативи рівня openFDA відіграють фундаментальну роль у процесі цифрової трансформації медицини, руйнуючи традиційну монополію державних

регуляторів на статистичну аналітику. Доступ до неагрегованих масивів Системи повідомлень про побічні ефекти через RESTful API формує принципово нове середовище для дослідницької діяльності. Машиночитані формати обміну інформацією дозволяють незалежним інженерам та дослідникам оминати етап складного ручного парсингу баз даних і безпосередньо інтегрувати актуальні клінічні звіти у спеціалізовані інформаційні панелі. Проте наявність самого лише доступу до сирих даних не вирішує проблеми їх швидкої клінічної інтерпретації. Для отримання реальної практичної користі ці масиви потребують застосування сучасних алгоритмів агрегації та динамічної візуалізації безпосередньо на стороні користувача. Саме тому виникає гостра необхідність у проєктуванні спеціалізованих клієнтських рішень, здатних миттєво трансформувати багаторівневі JSON-відповіді у зрозумілі статистичні зрізи. Створення такого інструменту дозволить спростити доступ до медичної аналітики, надавши зручний засіб для оперативного виявлення неочевидних побічних ефектів.

Дослідження наявних інструментів моніторингу безпеки лікарських засобів виявило суттєві архітектурні та функціональні прогалини на ринку медичного програмного забезпечення. Глобальні державні реєстри страждають від перевантажених інтерфейсів та низької швидкості обробки багатовимірних вибірок, тоді як комерційні портали надмірно спрощують статистику на догоду масовому споживачеві. Найбільш критичним недоліком сучасної інфраструктури фармаконагляду залишається відсутність інтуїтивних механізмів для прямого паралельного зіставлення профілів безпеки кількох діючих речовин. Виявлена проблематика чітко артикулює потребу у створенні легкого, вузькоспеціалізованого вебзастосунку, який би фокусувався виключно на швидкості перевірки клінічних гіпотез та візуальному виявленні перетинів симптомів у режимі Drug Comparison. Саме такий підхід дозволить дослідникам миттєво виявляти неочевидні кореляції між різними медикаментами без необхідності ручного зведення даних. Це, своєю чергою, значно оптимізує час на прийняття рішень під час аналізу потенційних клінічних ризиків.

Успішна реалізація сформульованого завдання вимагає застосування високопродуктивного та стійкого до помилок технологічного стека. Використання фреймворку Next.js у поєднанні з серверними компонентами React 19 ефективно розв'язує проблему повільного рендерингу важких клінічних масивів. Архітектура App Router нівелює надмірне навантаження на клієнтські пристрої завдяки кешуванню запитів та генерації базової розмітки на сервері. Водночас інтеграція суворої типізації TypeScript утворює надійний захисний контур проти структурної непередбачуваності сирих JSON-відповідей від урядових серверів. Створення декларативних інтерфейсів для об'єктів пацієнта та медикаменту повністю виключає ризик критичних збоїв під час побудови інтерактивних графіків, гарантуючи стабільне відображення валідованої статистичної інформації. Додатково використання утилітарного підходу Tailwind CSS забезпечить швидку адаптацію інтерфейсу під різні розміри екранів без втрати загальної швидкодії. Комплексне поєднання цих сучасних інструментів дозволить створити максимально чуйний та ергономічний аналітичний дашборд.

Сформований теоретичний та технологічний базис створює надійні передумови для переходу від загального аналізу концепції Open Data до практичної інженерної роботи. Досліджені обмеження існуючих платформ та визначена специфіка взаємодії з відкритим API формують чіткі вектори для подальшого проєктування програмного продукту. Наступним етапом розробки є безпосереднє формування вимог до аналітичних модулів, побудова UML-моделей поведінки дослідника та детальне опрацювання системної архітектури порівняльного застосунку, що буде комплексно реалізовано у другому розділі дослідження. Крім того, особлива увага приділятиметься оптимізації алгоритмічної складової для паралельної обробки великих обсягів медичних показників. Це забезпечить не лише функціональну повноту системи, але й її повну відповідність сучасним стандартам продуктивності вебзастосунків.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПОРІВНЯЛЬНОГО АНАЛІЗУ ДАНИХ OPENFDA

2.1 Формування вимог до системи та аналіз цільової аудиторії

Процес створення спеціалізованого програмного забезпечення для обробки медичної статистики вимагає чіткого розмежування архітектурних завдань на етапі планування. Формування вимог до аналітичної вебплатформи базується на розумінні специфіки роботи з відкритими реєстрами Управління з продовольства і медикаментів США. Гетерогенність та масивність сирих JSON-відповідей диктують необхідність впровадження жорстких функціональних та нефункціональних критеріїв, які гарантуватимуть стабільність, точність та зручність кінцевого продукту для дослідника.

Функціональне ядро системи будується навколо концепції єдиного аналітичного центру, що реалізується через повноцінний інтерактивний Dashboard. Головна вимога до цього модуля полягає у здатності приймати запити від користувача, звертатися до відкритого програмного інтерфейсу та трансформувати отримані статистичні сегменти (buckets) у репрезентативні візуальні панелі. Інтерфейс повинен забезпечувати миттєве відображення розподілу побічних ефектів за категоріями, віковими групами та статтю. Окремим критичним елементом сторінки є динамічний модуль сповіщень «Safety Alert Banner». Цей компонент має в реальному часі аналізувати відповіді сервера та автоматично генерувати попередження у разі виявлення критичної концентрації серйозних побічних реакцій або летальних випадків для обраного препарату. Така функціональність перетворює систему зі звичайного засобу візуалізації на проактивний інструмент моніторингу безпеки.

Взаємодія користувача з базою даних вимагає бездоганно налаштованого механізму пошуку. Враховуючи високу варіативність написання торгових назв медикаментів та наявність складних діючих речовин, застосунок повинен підтримувати інтелектуальне автодоповнення. Реалізація цієї вимоги передбачає

впровадження алгоритмів затримки виконання (debounce), що дозволяє надсилати пошукові запити до словників openFDA лише після завершення введення символів користувачем. Це запобігає вичерпанню квот на використання API та мінімізує ризик введення некоректних назв препаратів, які б призвели до повернення порожніх масивів даних.

Найбільш складною функціональною вимогою є реалізація ізольованого аналітичного середовища «Drug Comparison». Цей модуль має забезпечувати одночасний паралельний пошук інформації щодо двох різних лікарських засобів. Ключове завдання компонента – не просто вивести два набори графіків пліч-о-пліч, а програмно розрахувати перетини їхніх профілів безпеки. Система повинна алгоритмічно виявляти спільні побічні ефекти (Common Symptoms), зіставляти їхні частотні показники на основі поля «reactionmeddrapt» та виводити результати у вигляді порівняльних прогрес-барів і паралельних гістограм.

Специфіка роботи з медичними даними формує суворий перелік нефункціональних вимог, де на перше місце виходить продуктивність інтерфейсу. Аналітична панель повинна миттєво реагувати на дії користувача. Реалізація цієї вимоги повністю покладається на архітектурні можливості фреймворку Next.js App Router. Перенесення базової логіки маршрутизації та первинного завантаження каркаса сторінки на сторону сервера (Server-Side Rendering) дозволяє розвантажити браузер клієнта. Графіки Recharts повинні ініціалізуватися лише після повного отримання агрегованих JSON-масивів, забезпечуючи плавну анімацію та відсутність зависань інтерфейсу під час роботи з великими вибірками.

Відмовостійкість системи є критичною нефункціональною вимогою при інтеграції з відкритими урядовими реєстрами. Архітектура застосунку повинна бути стійкою до специфічних HTTP-відповідей екосистеми FAERS. Зокрема, повернення коду 404 зі статусом «NOT_FOUND» не має сприйматися системою як фатальний збій мережі. Застосунок зобов'язаний коректно обробляти такі ситуації, ізолювати помилку на рівні конкретного компонента та відображати

користувачеві коректний порожій стан, не блокуючи при цьому рендер інших частин сторінки.

Візуальна складова та ергономіка застосунку підпорядковуються вимогам до медичного програмного забезпечення. Інтерфейс має бути адаптивним, суворо типізованим візуально та позбавленим деструктивного декоративного шуму. Використання рушія Tailwind CSS 4 гарантує виконання цих умов завдяки утилітарному підходу до стилізації [18]. Система повинна зберігати високу контрастність типографіки, передбачуваність реакцій кнопок та стандартизовані відступи між графічними блоками, що суттєво знижує когнітивне навантаження на дослідника під час тривалих аналітичних сесій.

Проєктування логіки взаємодії вимагає глибокого розуміння професійного контексту кінцевого споживача програмного продукту. Цільовою аудиторією розроблюваної платформи виступають спеціалісти з фармаконагляду, клінічні дослідники, фармацевти-аналітики та медичні статистики. Це фахівці з високим рівнем доменної експертизи, чия повсякденна рутина пов'язана з обробкою колосальних обсягів розрізненої інформації про несприятливі клінічні події. Їхня основна мета полягає у швидкій верифікації сигналів небезпеки та об'єктивній оцінці співвідношення ризику та користі при призначенні специфічних курсів лікування.

Професійний «біль» такого дослідника криється в інструментальній недосконалості традиційних баз даних. За відсутності спеціалізованих вебзастосунків процес виявлення демографічних вразливостей перетворюється на виснажливу механічну роботу. Фахівець змушений експортувати сирі результати пошуку у форматах CSV або масивних JSON-файлах, завантажувати їх до табличних процесорів і вручну налаштовувати зведені таблиці. Агрегація даних за полем «patient.patientonsetage» для виділення вікових груп ризику або розбивка за кодами статі вимагає написання макросів або складних формул. Це не лише забирає години робочого часу, але й створює високу ймовірність

допущення критичних людських помилок при інтерпретації сирих кодів системи FAERS.

Впровадження розроблюваної платформи кардинально трансформує цей сценарій використання. Дослідник відкриває вебзастосунок і вводить назву діючої речовини у рядок пошуку на головному Dashboard. Система бере на себе всю алгоритмічну складність: самостійно формує коректний запит до API, виконує агрегацію «count», декодує числові ідентифікатори статі та конвертує вікові показники у стандартизовані діапазони. Замість безликих таблиць фахівець миттєво отримує набір інтерактивних візуалізацій. Це дозволяє одним поглядом оцінити, що певний кардіологічний препарат викликає ускладнення переважно у жінок віком понад шістдесят років, звільняючи когнітивні ресурси для безпосереднього клінічного аналізу проблеми.

Сценарій порівняння альтернативних медикаментів розкриває найвищу цінність інструменту. Класичний підхід вимагає від спеціаліста створення двох окремих ізольованих звітів та їхнього тривалого текстового зіставлення рядок за рядком. Наш гіпотетичний користувач стикається із завданням обрати безпечніший антидепресант для пацієнта з обтяженим анамнезом. Використовуючи модуль Drug Comparison, дослідник вводить дві різні діючі речовини у відповідні поля. Програмна архітектура генерує паралельні мережеві запити, а логіка TypeScript самостійно обчислює перетини у масивах побічних реакцій. Фахівець одразу бачить відсортований графік спільних симптомів, де довжина індикаторів прогресу наочно демонструє, який саме препарат частіше викликає тахікардію або безсоння. Процес, що раніше вимагав годинної концентрації, тепер зводиться до кількох кліків мишею та секундного очікування рендерингу компонентів. Така миттєва візуалізація критично важливих даних суттєво мінімізує ризик людської помилки під час ручної обробки багатовимірних клінічних масивів. У результаті фахівець отримує надійне аналітичне підґрунтя для прийняття обґрунтованих рішень щодо безпеки терапії, спираючись виключно на об'єктивну урядову статистику.

2.2 Побудова моделей системного аналізу (UML)

Проєктування поведінкової моделі програмного забезпечення вимагає формалізації взаємодії між кінцевим користувачем та функціональними вузлами системи. Цей процес дозволяє уникнути логічних хиб на етапі безпосереднього написання коду та формує чітке бачення архітектурних меж.

Візуалізація системних процесів на рівні високорівневих абстракцій починається з визначення головних дійових осіб. Головним актором у розроблюваній екосистемі виступає «Дослідник» – медичний статистик або фахівець із фармаконагляду, який ініціює більшість процесів в інтерактивному середовищі. Дослідник безпосередньо взаємодіє з графічним інтерфейсом для виконання своїх професійних завдань, серед яких базовим прецедентом є «Пошук лікарського засобу». Цей процес запускається через введення тексту у поле з підтримкою інтелектуального автодоповнення, після чого актор отримує доступ до прецеденту «Перегляд аналітики на Dashboard». У межах цієї панелі користувач аналізує розподіл несприятливих подій за віком та статтю, маніпулюючи інтерактивними компонентами згенерованих діаграм.

Специфічним та найбільш комплексним прецедентом виступає «Порівняння двох препаратів», який реалізується в ізольованому аналітичному модулі Drug Comparison. Цей сценарій використання вимагає від актора почергового або одночасного визначення двох цільових медикаментів для ініціалізації паралельного зіставлення. Додатковим пасивним прецедентом, який не потребує прямої дії користувача для свого запуску, є «Перегляд критичних сповіщень». Динамічний модуль Safety Alert функціонує у фоновому режимі та генерує візуальне попередження на екрані лише у разі програмного виявлення статистичних аномалій або високої концентрації летальних наслідків у завантаженій вибірці даних. Загальна архітектура взаємодії актора з функціональними модулями системи представлена на рисунку 2.1.

Рисунок 2.1 – Діаграма прецедентів інформаційної системи

Структурна цілісність моделі прецедентів забезпечується введенням стереотипних зв'язків між елементами системи. Будь-яка ініціація пошуку, перегляд дашборду або запуск порівняльного аналізу нерозривно пов'язані з мережевою взаємодією. З огляду на це прецеденти пошуку та порівняння включають базовий системний прецедент «Запит до відкритого API» через відношення стереотипу «include». Система гарантовано формує параметризований URL та звертається до серверів openFDA при кожній зміні критеріїв медичної вибірки. Водночас архітектура передбачає легітимні ситуації, коли зовнішній реєстр не містить інформації за вказаними параметрами. У такому разі базовий прецедент мережевого запиту розширюється системним прецедентом «Обробка помилки 404» через відношення «extend». Ця гілка логіки активується виключно за умови відсутності збігів у базі FAERS і відповідає за

відображення коректного стану порожнього екрана замість аварійного завершення роботи клієнтського застосунку. Такий підхід дозволяє чітко розмежувати нормальний потік виконання програми та обробку виняткових ситуацій ще на етапі високорівневого проєктування. Завдяки цьому користувач завжди отримує інформативний зворотний зв'язок від інтерфейсу, що суттєво підвищує загальну ергономічність програмного продукту. Формалізація цих процесів у вигляді об'єктно-орієнтованої моделі створює надійне фундаментальне підґрунтя для подальшої розробки стійкої архітектури клієнтської частини.

Моделювання динаміки обміну повідомленнями між об'єктами під час виконання конкретного сценарію є критично важливим для розуміння асинхронної природи вебзастосунків. Для розроблюваної платформи найбільш показовим є життєвий цикл комплексного запиту в межах модуля Drug Comparison. Процес ініціюється на боці клієнта, коли веббраузер реєструє подію підтвердження пошуку двох діючих речовин. Компонент React миттєво фіксує зміну локального стану, блокує поля введення для запобігання дублюванню дій та запускає асинхронну функцію обробки даних. На цьому етапі архітектура формує два незалежні параметризовані URL-запити до кінцевої точки «drug/event.json», передаючи відповідні нормалізовані ідентифікатори медикаментів та директиви агрегації «count». Для мінімізації мережевих затримок та оптимізації продуктивності ці виклики ініціюються паралельно, утворюючи два незалежні потоки виконання. Подальший рендеринг порівняльних графіків або перехід у стан помилки відбувається лише після успішної синхронізації та валідації відповідей від обох запитів. Такий строгий детермінований контроль за паралельними мережевими транзакціями повністю унеможливлює відображення неповних або розсинхронізованих медичних даних на екрані дослідника. Покроковий процес обміну повідомленнями між клієнтською частиною, контролером та зовнішнім API детально відображено на рисунку 2.2.

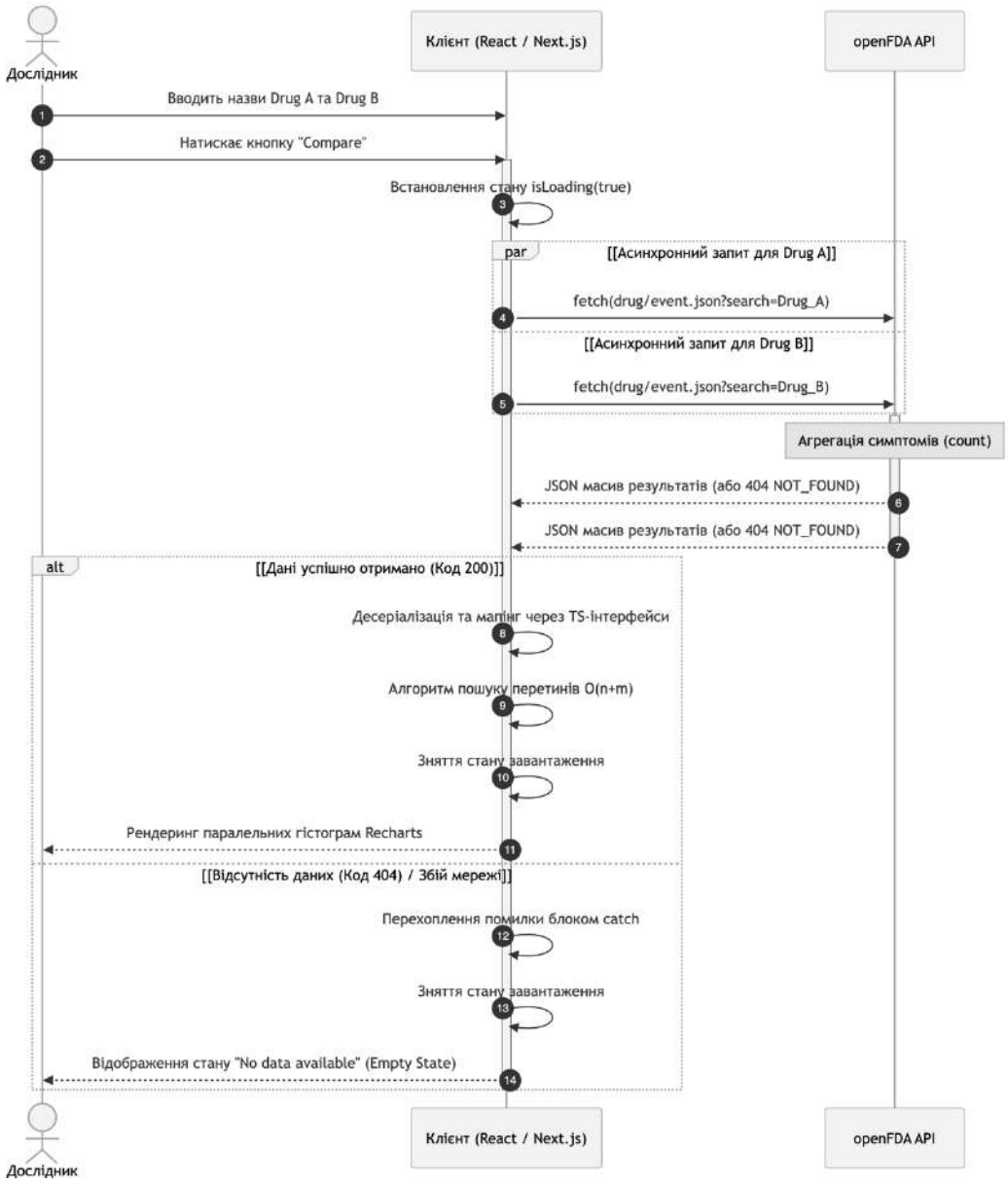


Рисунок 2.2 – Діаграма послідовності для модуля Drug Comparison

Виконання паралельних мережевих викликів суворо координується за допомогою вбудованої конструкції JavaScript «Promise.allSettled». Цей архітектурний вибір гарантує повну ізоляцію та незалежність потоків виконання один від одного. Клієнтський застосунок надсилає сформовані HTTP-запити до серверної інфраструктури Управління з продовольства і медикаментів США, переходячи у стан очікування (pending) та активуючи візуальні індикатори завантаження. Хмарний сервер приймає запити, здійснює пошук у колосальних масивах даних, виконує агрегацію частоти симптомів за полем

«reactionmeddrapt» із використанням точного матчингу та генерує дві відповіді у форматі JSON.

Після повернення пакетів даних на сторону клієнта починається критичний етап мапінгу результатів. Функція-контролер отримує сирі масиви та пропускає їх через декларативні TypeScript-інтерфейси. На цьому кроці логіка системи розділяє успішні результати від відхилених запитів. Якщо сервер повертає статус 404 через відсутність зареєстрованих побічних ефектів для одного з препаратів, проміс набуває стану «rejected», і застосунок готує відповідне інформаційне сповіщення. Валідовані об'єкти з успішних відповідей передаються до алгоритмічного блоку обчислення перетинів. Система ітерує через списки реакцій обох медикаментів, ідентифікує спільні симптоми (Common Symptoms) та формує нову зведену структуру даних, де кожному виявленому ускладненню відповідають одразу дві частотні метрики.

Фінальна стадія описаної послідовності полягає у передачі трансформованих масивів до візуального шару застосунку. Зведений об'єкт із перетинами надсилається у вигляді властивостей до екземплярів діаграм бібліотеки Recharts. Внутрішні механізми фреймворку фіксують оновлення вхідних даних, що запускає алгоритм узгодження (reconciliation) та подальший рендер оновленого віртуального дерева елементів. Браузер відмальовує паралельні гістограми та прогрес-бари, відображаючи досліднику готову аналітичну панель, що символізує успішне завершення життєвого циклу складного порівняльного прецеденту.

2.3 Архітектурна модель та структура даних застосунку

Проєктування архітектури сучасної інформаційної системи вимагає критичного переосмислення традиційних підходів до збереження та обробки даних. Класичні монолітні застосунки зазвичай спираються на власні реляційні бази даних на кшталт PostgreSQL або MySQL для управління станом системи та

зберігання накопиченої аналітики. Розроблювана платформа свідомо відмовляється від такої парадигми на користь архітектури безсерверного агрегатора. Вебзастосунок не має власного важкого бекенду для синхронізації чи дублювання клінічних репортів. Уся необхідна інформація витягується безпосередньо з віддаленого урядового сховища в режимі реального часу, що кардинально знижує вартість підтримки інфраструктури та нівелює ризики роботи із застарілими датасетами. Крім того, відсутність локального сховища автоматично мінімізує вектори потенційних кібератак, оскільки масиви медичних відомостей не накопичуються на проміжних вузлах системи. Натомість основне обчислювальне навантаження ефективно перерозподіляється між серверами відкритого API та клієнтським інтерфейсом, який відповідає за фінальний мапінг і візуалізацію показників. Як наслідок, формується стійка до високих навантажень екосистема, що здатна динамічно масштабуватися незалежно від кількості одночасних запитів від дослідників

У цій клієнт-серверній моделі хмарний реєстр openFDA виконує фундаментальну роль єдиного джерела правди. Відсутність проміжної локальної бази даних гарантує, що дослідник завжди працює з найактуальнішим зрізом медичної статистики, доступним на серверах Управління з продовольства і медикаментів США. Замість складних багатокрокових процесів трансформації для перенесення гігабайтів інформації у власну інфраструктуру, система діє як легкий інтелектуальний шлюз [**Error! Reference source not found.**]. Застосунок приймає параметри від користувача, конструює оптимізовані пошукові запити до ендпоінту «drug/event.json» та забезпечує виключно транзитну обробку повернених JSON-масивів без їхнього довготривалого збереження на жорсткому диску. Такий архітектурний підхід не лише оптимізує використання оперативної пам'яті клієнтського пристрою, але й підвищує загальну відмовостійкість платформи під час пікових навантажень. Водночас для безпечної десеріалізації та коректного мапінгу цих масивних ієрархічних відповідей необхідно мати чітке уявлення про організацію первинних даних регулятора. Логічну структуру,

взаємозв'язки та ключові атрибути компонентів цього зовнішнього масиву медичної інформації відображено на концептуальній ER-діаграмі предметної області (рисунок 2.3).

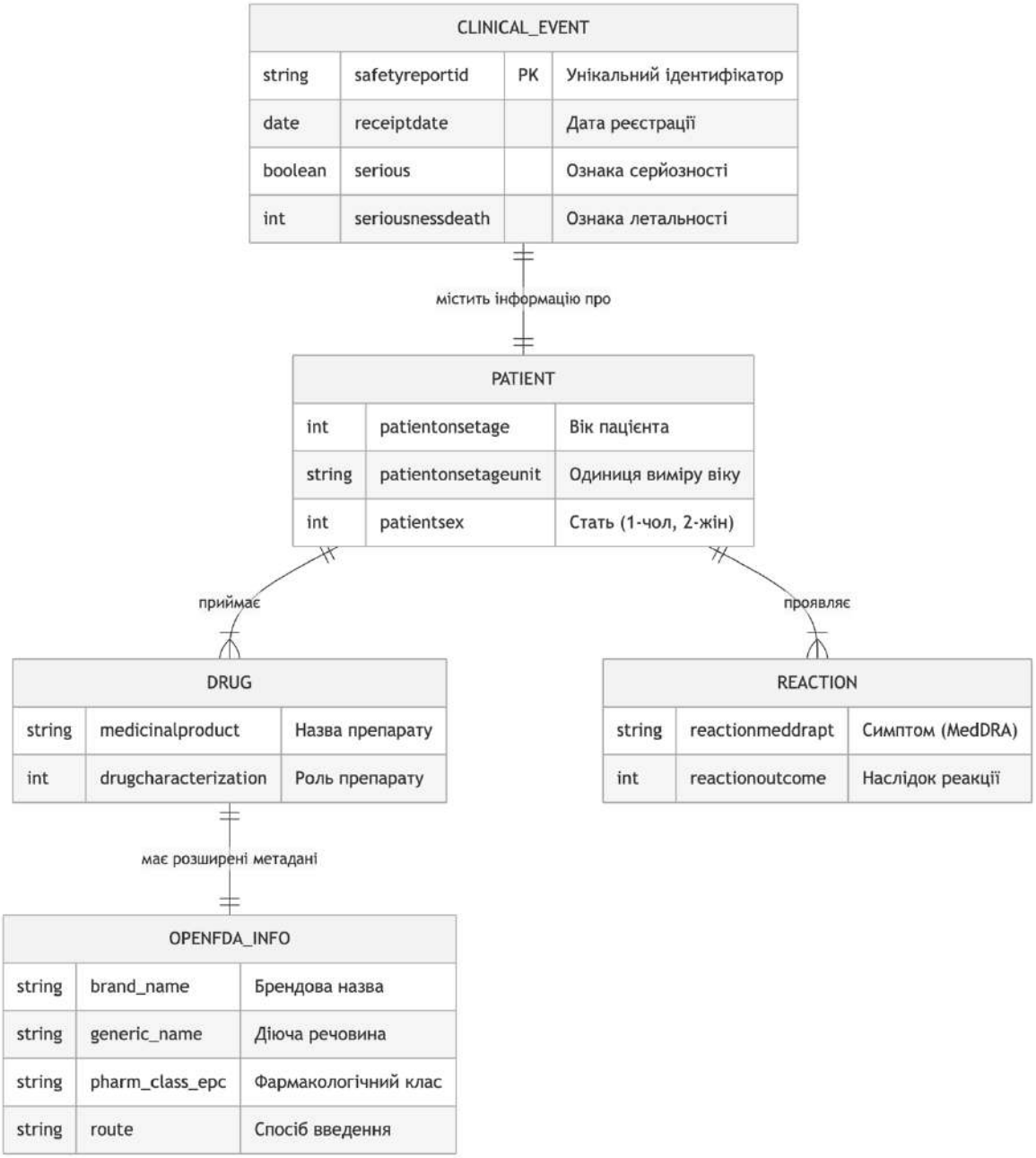


Рисунок 2.3 – Концептуальна ER-діаграма предметної області відкритих медичних даних системи FAERS

Структурна організація кодової бази спирається на можливості директорії «app/» у межах фреймворку Next.js, що забезпечує жорстке розділення відповідальності між компонентами системи. Кореневі файли маршрутизації, такі як «layout.tsx» та «page.tsx», функціонують виключно як серверні компоненти (Server Components). На цьому рівні інкапсулюється глобальна конфігурація застосунку, підключення стилів Tailwind CSS 4, налаштування метаданих та формування базового каркаса HTML-сторінки. Серверні компоненти здатні миттєво генерувати статичну розмітку та віддавати її браузеру без передачі зайвого JavaScript-коду. Це гарантує швидке відображення навігаційних панелей ще до того, як об'ємні масиви медичних даних будуть завантажені з віддаленого джерела. Взаємозв'язок, архітектурне розділення та логіку взаємодії між зазначеними серверними елементами, клієнтськими модулями візуалізації та зовнішнім інтерфейсом openFDA представлено на діаграмі компонентів (рисунок 2.4).

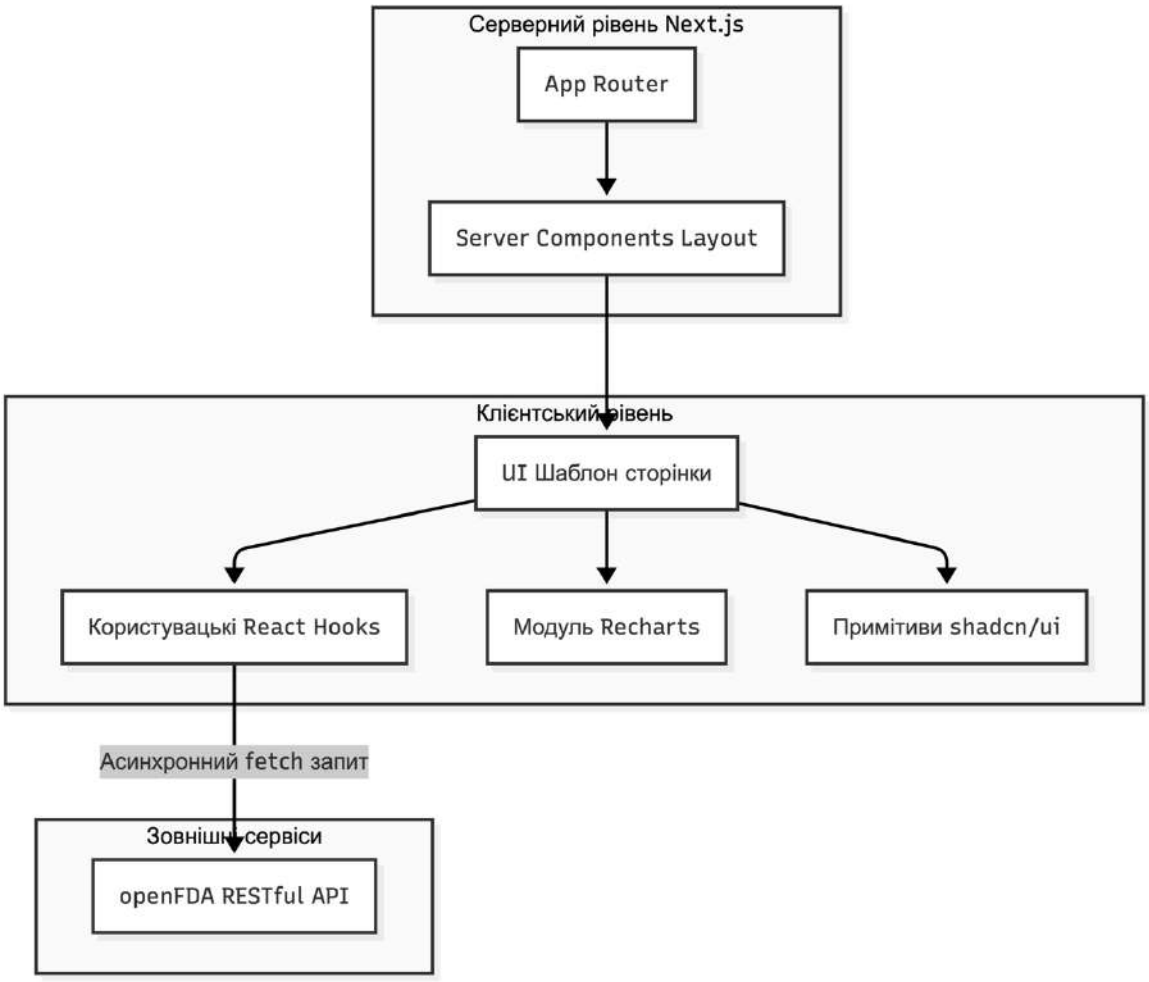


Рисунок 2.4 – Діаграма компонентів веб-орієнтованої інформаційної системи

Паралельно із серверним шаром, директорія «components/» містить набір ізольованих клієнтських компонентів, які маркуються директивою «use client». Саме ці вузли архітектури беруть на себе відповідальність за безпосередню асинхронну взаємодію з хмарним API та управління локальним станом інтерфейсу. Модулі пошуку, панелі автодоповнення та візуальні блоки на базі бібліотеки Recharts ініціюють мережеві fetch-запити [**Error! Reference source not found.**] до відкритого реєстру після монтування у віртуальне дерево елементів. Такий гранулярний підхід дозволяє виконувати ресурсомісткі обчислення, пов'язані з алгоритмічним виявленням перетинів симптомів або парсингом масивів «brand_name», виключно в межах конкретного аналітичного

віджета. Ізоляція логіки гарантує, що помилка обробки даних у клієнтському модулі порівняння препаратів не призведе до падіння глобального layout-шару.

Описаний архітектурний поділ створює безпрецедентні переваги у контексті оптимізації навантаження на серверну інфраструктуру. Застарілі монолітні підходи змушують бекенд не лише обробляти HTTP-звернення, а й виконувати складні маніпуляції з даними перед їхньою відправкою кінцевому споживачу. Делегування обов'язків із десеріалізації сирих JSON-відповідей та трансформації агрегацій безпосередньо клієнтським компонентам суттєво розвантажує хмарні потужності платформи. Хостинг-провайдер виконує функцію швидкої доставки статичної розмітки, тоді як подальший обмін даними та побудова графіків відбуваються фактично між локальним комп'ютером дослідника та серверами FDA.

Швидкість доставки контенту до кінцевого користувача є ключовим індикатором якості будь-якого медичного аналітичного інструменту. Відмова від класичної прив'язки до бази даних дозволяє розгорнути застосунок на периферійних вузлах мережі, максимально наблизивши точку входу до фізичного розташування спеціаліста з фармаконагляду. Дослідник отримує оптимізований інтерфейс за мілісекунди, після чого інтелектуальні компоненти платформи розпочинають точкову асинхронну взаємодію з відкритими реєстрами. Ця синергія серверного рендерингу для системного каркаса та реактивної клієнтської логіки для масивів даних формує еталонну модель, ідеально адаптовану під жорсткі вимоги сучасного візуального аналізу в медицині.

Інтеграція з відкритими реєстрами медичної статистики приховує фундаментальні загрози через надзвичайну гетерогенність та непередбачуваність вхідних даних. Сирий масив результатів, який повертає кінцева точка хмарного API, є алгоритмічно небезпечним для безпосередньої передачі у візуальні компоненти клієнтського застосунку. Відповідь сервера характеризується глибокою ієрархічною вкладеністю та масовою відсутністю

ключових вузлів, коли очікуваний об'єкт клінічного випадку може бути представлений як «null» або взагалі не міститися у загальній структурі JSON. Спроба прямої ін'єкції таких нестабільних об'єктів у логіку рендерингу інтерфейсу неминуче провокує каскадні збої, оскільки математичні агрегації або спроби ітерації по невизначеному масиву викликають фатальні помилки виконання коду. Ситуація ускладнюється змішаними типами даних, де числові ідентифікатори або параметри віку пацієнта часто передаються сервером у вигляді звичайних текстових рядків.

Вирішення цієї інженерної проблеми вимагає побудови надійного захисного шару на основі суворої типізації TypeScript [**Error! Reference source not found.**]. Процес розробки розпочинається зі створення ізольованих декларативних інтерфейсів, які описують ідеальну та повністю безпечну структуру даних для кожної конкретної аналітичної панелі. Замість оперування сирими відповідями, система впроваджує прикладні типи, адаптовані виключно під потреби візуалізації. Показовим прикладом є структура «AgeDataPoint», яка спеціально спроєктована для графіків розподілу демографічних ризиків і містить лише валідовані поля «ageGroup» у форматі рядка та «count» у вигляді суворого числа. Для модуля динамічних сповіщень розроблено тип «TopDrugResult», що жорстко фіксує ідентифікатор медикаменту та загальну кількість зафіксованих критичних наслідків. Це утворює надійний фундамент для роботи компонента Safety Alert Banner, повністю усуваючи ризик звернення до неіснуючих атрибутів під час аналізу активного препарату.

Проєктування архітектури для ізольованого середовища порівняльного аналізу вимагає ще вищого рівня абстракції. Модуль Drug Comparison функціонує в умовах одночасної обробки двох незалежних потоків даних, де ключовим алгоритмічним завданням є виявлення статистичних перетинів між побічними ефектами. Для збереження результатів цього складного зіставлення було розроблено спеціалізований тип «CommonSymptom». Цей інтерфейс діє як надійний інформаційний міст, об'єднуючи уніфіковану назву побічної реакції з

двома числовими властивостями, які відображають абсолютну частоту фіксації цього конкретного симптому для першого та другого препаратів відповідно. Застосування такої структури дозволяє консолідувати розрізнені серверні агрегації у єдину типізовану сутність, математично готову для відображення паралельних гістограм.

Перехід від нестабільного JSON-формату до задекларованих інтерфейсів реалізується через механізм типізованого мапінгу, який формує своєрідний конвеєр трансформації даних. Спеціально написані функції-мапери ітерують по масивах результатів, виконуючи роль суворих диспетчерів типів на етапі обробки запиту. Вони програмно перевіряють наявність кожного необхідного вузла, здійснюють безпечне приведення рядкових значень віку чи статі до числових форматів та безжально відсіюють пошкоджені або неповні записи. Головна мета цього процесу полягає у перетворенні глибоко вкладених ієрархій на ідеально плоскі масиви. Бібліотека Recharts алгоритмічно не здатна працювати з об'ємними багатовимірними деревами, тому математично правильне сплющення даних під час мапінгу є критичною передумовою для коректної побудови візуальних осей та обчислення пропорцій графічних сегментів.

Фінальна ланка цієї архітектурної моделі гарантує абсолютну безпеку на етапі рендерингу користувацького інтерфейсу. Сформовані та валідовані TypeScript-інтерфейси використовуються для суворої типізації властивостей, які передаються безпосередньо у візуальні компоненти клієнтського шару. Статичний аналізатор коду ще на етапі компіляції фізично блокує можливість передачі об'єкта, що не відповідає визначеному контракту, у компонент діаграми. Це повністю виключає сценарії, за яких значення «undefined», «null» або нечислові результати формату «NaN» могли б випадково потрапити у математичні функції обчислення координат на екрані. Завдяки такому детермінованому підходу вебзастосунок набуває максимальної відмовостійкості, а дослідник отримує гарантію, що навіть при надходженні екстремально

аномальних даних від серверів відкритого реєстру інтерфейс аналітичної платформи ніколи не зазнає раптового падіння під час виконання сесії.

2.4 Проєктування інтерфейсу користувача та UX-логіки

Проєктування візуального шару для медичних інформаційних систем вимагає фундаментального відходу від традиційних патернів вебдизайну. Архітектура інтерфейсу має підпорядковуватися суворим вимогам точності, доступності та мінімізації когнітивного навантаження на дослідника під час опрацювання масивів бази FAERS [22]. Реалізація цих принципів досягається шляхом інтеграції сучасних концепцій розробки UI-компонентів та науково обґрунтованого підходу до колірного кодування складних статистичних візуалізацій.

Сучасна екосистема розробки користувацьких інтерфейсів пропонує безліч готових бібліотек, однак їхнє використання у спеціалізованому медичному програмному забезпеченні часто супроводжується критичними недоліками. Класичні UI-фреймворки нав'язують власну візуальну стилістику та генерують надлишковий CSS-код, який вкрай важко перевизначити під специфічні потреби клінічної платформи. З огляду на це, архітектура розроблюваного вебзастосунку спирається на концепцію «headless» (нестилізованих) компонентів, реалізовану через екосистему Radix UI [**Error! Reference source not found.**]. Ця бібліотека надає виключно функціональну логіку, управління внутрішнім станом та базові примітиви доступності, повністю делегуючи відповідальність за зовнішній вигляд розробнику. Інженер отримує можливість проєктувати складні інтерактивні елементи, контролюючи кожен рядок згенерованої HTML-розмітки за допомогою атомарних класів Tailwind CSS 4.

Інтеграція колекції shadcn докорінно змінює традиційний підхід до управління залежностями у фронтенд-розробці [**Error! Reference source not**

found.]. На відміну від класичних NPM-пакетів, які інкапсулюють свій код у прихованих директоріях, shadcn пропонує патерн прямого копіювання вихідного коду безпосередньо у репозиторій проєкту. Завдяки цьому всі базові компоненти фізично розміщуються у локальній директорії «components/ui/», стаючи повноцінною частиною кодової бази застосунку. Такий рівень архітектурної свободи є критичним для медичної платформи. Розробник може безперешкодно модифікувати внутрішню логіку діалогових вікон, налаштовувати специфічні анімації для панелей автодоповнення пошуку або розширювати інтерфейси властивостей без необхідності очікувати на оновлення від сторонніх мейнтейнерів. Це гарантує, що інтерфейс системи може еволюціонувати синхронно зі змінами у структурі JSON-відповідей від openFDA.

Ключовим критерієм якості професійного медичного інструментарію є його безкомпромісна доступність (Accessibility або a11y). Фахівці з фармаконагляду працюють у різноманітних умовах, а інформаційна система повинна відповідати міжнародним стандартам інклюзивності WCAG [29]. Базові примітиви Radix UI автоматично беруть на себе вирішення найскладніших проблем доступності. При відкритті модальних вікон з деталями побічних реакцій система самостійно ізолює фокус клавіатури, не дозволяючи користувачеві випадково взаємодіяти з фоновими елементами. Усі випадаючі списки та складні форми пошуку ліків підтримують повноцінну навігацію за допомогою стрілок клавіатури. Вбудована генерація правильних ARIA-атрибутів гарантує, що екранні диктори (screen readers) коректно інтерпретують зміни в інтерфейсі, озвучуючи дослідникам появу нових сповіщень про безпеку або результати статистичних агрегацій.

Колір у контексті проєктування аналітичних дашбордів перестає бути виключно естетичним інструментом і набуває статусу повноцінного каналу передачі даних. Вибір колірної палітри для медичної платформи базується на принципах семантичного кодування, де кожен відтінок виконує суворо визначену інформаційну функцію. Непродумане використання яскравих

кольорів здатне катастрофічно погіршити користувацький досвід, зміщуючи акценти з реальних статистичних сигналів на другорядні елементи інтерфейсу. Базовий каркас застосунку використовує стриману монохромну гаму з відтінками сірого та синього, що створює нейтральне тло для роботи з масивами тексту та числовими метриками.

Особливої обережності вимагає застосування тривожних сигнальних кольорів. Використання насиченого червоного або яскраво-помаранчевого спектра жорстко обмежено архітектурою дизайн-системи. Ці кольори зарезервовані виключно для модуля Safety Alert Banner та відображення критичних клінічних наслідків (наприклад, летальних випадків, зафіксованих у масиві openFDA). Якщо інтерфейс почне підсвічувати червоним кольором кожну незначну побічну реакцію, таку як легка нудота або головний біль, дослідник неминуче зіткнеться з психологічним феноменом «втоми від сповіщень» (alert fatigue). Систематична гіперстимуляція призводить до того, що фахівець підсвідомо починає ігнорувати інтерфейсні попередження, що в умовах реального фармаконагляду може мати фатальні наслідки. Сувора економія тривожних відтінків гарантує, що їхня раптова поява на екрані миттєво приверне максимальну увагу користувача до справді небезпечних аномалій.

Проєктування візуальної логіки для модуля Drug Comparison висуває принципово інші виклики до кольоропередачі. Одночасне відображення статистичних зрізів для двох різних діючих речовин в єдиному компоненті Recharts створює високий ризик когнітивного перевантаження. Інженерна проблема полягає у необхідності чітко розмежувати дані, зберігши при цьому цілісність загальної картини. Вирішення цього завдання досягається через використання парних контрастних, але візуально неагресивних кольорів для паралельних гістограм. Застосування глибокого холодного синього відтінку для першого препарату (Drug A) та теплого нейтрального або приглушеного смарагдового для другого (Drug B) створює необхідний оптичний контраст.

Закріплення специфічного кольору за конкретним медикаментом на рівні локального стану React-компонента забезпечує наскрізну консистентність усієї аналітичної сесії. Дослідник, який перемикає увагу між графіком загальних симптомів, розподілом за статтю та віковою гістограмою, завжди бачить обрані препарати в одних і тих самих кольорах. Це позбавляє фахівця необхідності постійно звірятися з легендою графіка під час тривалого аналізу. Чітке колірне кодування дозволяє людському оку миттєво зчитувати різницю у довжині індикаторів прогресу (progress bars), інтуїтивно розуміючи, який саме препарат демонструє вищу частоту конкретного ускладнення, перетворюючи сирі JSON-метрики на прозорий і легко засвоюваний аналітичний продукт.

2.5 Алгоритмічне забезпечення модуля порівняння

Програмна реалізація аналітичного модуля Drug Comparison вимагає вирішення нетривіальної задачі злиття двох незалежних масивів медичних даних, отриманих від API Управління з продовольства і медикаментів США. Після успішного паралельного виконання HTTP-запитів клієнтський застосунок отримує два масиви об'єктів, які містять тисячі унікальних назв побічних реакцій («reactionmeddrapt») та відповідні показники їхньої абсолютної частоти («count») для першого та другого препаратів. Ключова мета алгоритмічного блоку полягає у швидкій ідентифікації спільних симптомів (перетинів) та формуванні єдиної консолідованої структури даних, яка безпосередньо передається до компонентів гістограм бібліотеки Recharts.

Наївний алгоритмічний підхід до вирішення задачі пошуку перетинів передбачає використання вкладених циклів. За такої моделі кожен елемент масиву побічних ефектів першого препарату послідовно порівнюється з кожним елементом масиву другого препарату. Математична часова складність такої операції визначається як $O(n*m)$, де n та m – кількості унікальних симптомів для відповідних медикаментів. У контексті обробки масштабних клінічних датасетів,

які часто налічують тисячі записів, застосування цього методу є критично неефективним. Виконання настільки важких синхронних обчислень у головному потоці JavaScript неминуче призводить до блокування циклу подій (Event Loop), що візуально проявляється як тривале зависання користувацького інтерфейсу та втрата чуйності платформи на дії дослідника.

Оптимізація обчислювального процесу досягається шляхом впровадження алгоритму на основі структури хеш-таблиць. У середовищі суворої типізації TypeScript ця концепція реалізується за допомогою вбудованого об'єкта «Map» або типізованого словника «Record». Логіка обробки розділяється на два незалежні лінійні проходи. На першому етапі система ітерує масив першого препарату, записуючи нормалізовану назву побічного ефекту як унікальний ключ словника, а його частоту – як значення. На другому етапі виконується ітерація масиву результатів другого медикаменту з одночасною перевіркою наявності поточного симптому в уже сформованій хеш-таблиці. Якщо ключ збігається, алгоритм миттєво конструює об'єкт перетину «CommonSymptom», який інкапсулює назву реакції та частотні показники одразу обох лікарських засобів.

Застосування індексації за ключем докорінно змінює продуктивність аналітичного модуля. Часова складність описаного алгоритму знижується до лінійного показника $O(n+m)$, оскільки операція читання з хеш-таблиці виконується за константний час $O(1)$ [**Error! Reference source not found.**]. З інженерної точки зору це означає, що розрахунок перетинів навіть для гігантських вибірок відбувається за лічені мілісекунди безпосередньо у браузері клієнта. Оптимізована логіка обчислень безпечно інкапсулюється всередині користувацьких хуків React, гарантуючи миттєвий рендер паралельних графіків без потреби винесення цієї задачі на сервер чи впровадження технології Web Workers [**Error! Reference source not found.**].

2.6 Висновки до другого розділу

Проведене системне проєктування сформувало надійний архітектурний базис для розробки медичної аналітичної платформи. Побудова моделей прецедентів та діаграм послідовності дозволила чітко окреслити межі взаємодії фахівця з візуальним інтерфейсом і визначити життєвий цикл складних асинхронних транзакцій. Детальний аналіз поведінкових патернів виявив критичну необхідність легітимної обробки порожніх відповідей API з кодом 404 на стороні клієнта, що лягло в основу механізмів загальної відмовостійкості вебзастосунку під час паралельних пошукових запитів.

Вибір безсерверної клієнт-серверної моделі з використанням віддаленого реєстру openFDA як єдиного джерела правди повністю усунув потребу у підтримці та синхронізації власних масивних баз даних. Делегування базової маршрутизації серверним компонентам Next.js та передача логіки асинхронних запитів ізольованим клієнтським віджетам забезпечили оптимальну швидкість доставки контенту. Водночас розробка суворих декларативних TypeScript-інтерфейсів для мапінгу відповідей утворила надійний програмний щит. Цей шар трансформації ефективно конвертує гетерогенні та неповні JSON-ієрархії на плоскі валідовані структури, гарантуючи безперебійну роботу системи візуалізації.

Інтеграція нестилізованих примітивів Radix UI та рушія Tailwind CSS 4 дозволила створити інклюзивний, семантично правильний інтерфейс з продуманою колірною логікою, що мінімізує ефект втоми від сповіщень. Алгоритмічна оптимізація модуля Drug Comparison через перехід до складності $O(n+m)$ остаточно розв'язала проблему клієнтського рендерингу об'ємних статистичних перетинів. Сформовані вимоги, обґрунтовані архітектурні рішення та спроєктовані алгоритмічні моделі утворюють цілісну технічну специфікацію, що дозволяє перейти до етапу безпосередньої програмної реалізації кодової бази застосунку в наступному розділі дослідження.

3 ІМПЛЕМЕНТАЦІЯ ТА ОЦІНКА ПРОДУКТИВНОСТІ АНАЛІТИЧНОГО ВЕБЗАСТОСУНКУ

3.1 Налаштування середовища розробки та конфігурація Tailwind CSS 4

Процес програмної реалізації аналітичної платформи базується на розгортанні сучасного робочого середовища з використанням фреймворку Next.js у конфігурації App Router [**Error! Reference source not found.**]. Ініціалізацію проєкту було здійснено із застосуванням мови TypeScript як основного інструменту забезпечення безпеки типів. Архітектурна топологія кодової бази передбачає суворе розділення відповідальності на рівні файлової системи. Директорія «app/» інкапсулює логіку серверної маршрутизації та глобальні конфігураційні файли, директорія «components/» виділена для збереження ізольованих клієнтських та серверних віджетів інтерфейсу, а в каталозі «lib/» зосереджено допоміжні утиліти для форматування даних та обробки класів стилів. Такий підхід формує детерміновану інфраструктуру, де кожен модуль має чітко визначене місце, що є критично важливою умовою для масштабування системи фармаконагляду.

Ключовим етапом налаштування візуального шару стала інтеграція четвертої версії компілятора Tailwind CSS. Цей реліз запроваджує фундаментальну зміну парадигми стилізації, повністю відмовляючись від традиційного конфігураційного файлу на базі JavaScript. Замість використання зовнішнього об'єкта налаштувань, конфігурацію системи було перенесено безпосередньо у головний файл стилів за принципом первинності каскадних таблиць (CSS-first configuration). Усі семантичні змінні, спеціалізовані кольорові палітри для розрізнення медикаментів у модулі порівняння та параметри типографіки тепер декларуються через нативні CSS-змінні у кореневому селекторі. Це інженерне рішення не лише радикально пришвидшує процес компіляції стильового бандла, але й дозволяє системі динамічно реагувати на зміну станів без необхідності перезавантаження дерева компонентів React.

Подальше формування інтерфейсу базувалося на інтеграції колекції нестилізованих примітивів shadcn. Процес ініціалізації цього середовища у зв'язці з новою архітектурою Tailwind вимагає специфічного підходу до підключення базових утиліт. Система не встановлюється як монолітна залежність через пакетний менеджер, а розгортається шляхом копіювання вихідного коду компонентів безпосередньо у робочу директорію проєкту. Це гарантує інженеру повний контроль над HTML-розміткою кожної діалогової панелі чи форми автодоповнення. Взаємодія атомарних класів Tailwind CSS 4 із внутрішньою логікою доступності від Radix UI створює міцний симбіоз, де візуальна гнучкість жодним чином не суперечить міжнародним стандартам інклюзивності для медичного програмного забезпечення.

Завершення базового налаштування передбачало встановлення спеціалізованих залежностей для візуалізації статистичних агрегацій та забезпечення консистентної іконографіки. Для побудови паралельних гістограм та динамічних панелей розподілу демографічних ризиків було інтегровано бібліотеку Recharts, яка нативно підтримує роботу з компонентною моделлю React 19 [**Error! Reference source not found.**]. Візуальна ідентифікація елементів управління, таких як кнопки ініціалізації пошуку або індикатори завантаження інформації від API, реалізована за допомогою пакета Lucide React [34]. Використання векторних графічних елементів з єдиною товщиною ліній та строгими пропорціями підтримує загальний мінімалістичний стиль клінічного інтерфейсу, не перевантажуючи дослідника зайвим візуальним шумом.

Проведений комплекс інфраструктурних налаштувань формує жорсткий, глибоко типізований та стилістично консистентний фундамент розроблюваної аналітичної платформи. Відмова від застарілих методів глобальної стилізації на користь CSS-first конфігурації у тісній синергії з компонентним підходом створює середовище, максимально стійке до візуальних регресій. Спроєктована клієнтська екосистема повністю готова до прийому гетерогенних масивів даних від віддалених серверів. Отримана архітектурна база дозволяє безпечно перейти

до етапу розробки складних асинхронних модулів взаємодії з відкритим програмним інтерфейсом FDA, де основний фокус розробки зміститься на створення типізованих запитів та алгоритмічну обробку відповідей.

3.2 Розробка модулів взаємодії з openFDA API

Програмна взаємодія з віддаленими серверами Управління з продовольства і медикаментів США побудована на використанні нативного браузерного інтерфейсу «fetch». Відмова від важких сторонніх бібліотек для виконання мережевих запитів зумовлена прагненням мінімізувати обсяг фінального збірочного файлу та оптимізувати продуктивність клієнтської частини. Застосування синтаксичних конструкцій «async» та «await» забезпечує неблокуюче виконання коду під час очікування відповіді від хмарного реєстру. Це дозволяє головному потоку JavaScript продовжувати обробку користувацьких подій, таких як скролінг сторінки або наведення курсора на інтерактивні елементи, поки мережевий рівень асинхронно здійснює обмін пакетами даних.

Управління життєвим циклом мережевого запиту в межах компонентів React вимагає проєктування багатовимірної архітектури станів. Використання базового хука «useState» дозволяє зберігати не лише трансформовані масиви медичної статистики, але й суворо контролювати проміжні фази транзакції. Логіка компонента передбачає обов'язкову ініціалізацію стану «isLoading» перед початком звернення до API. Цей маркер слугує тригером для відображення пульсуючих скелетонів (skeleton loaders) у системних блоках або інтерактивних індикаторів завантаження в полі пошуку. Паралельно архітектура резервує окремий стан «error» для перехоплення мережевих збоїв, котрий активується виключно всередині блоку «catch» конструкції «try/catch», гарантуючи безпечне опрацювання виняткових ситуацій. У разі виявлення помилки інтерфейс миттєво блокує рендеринг графіків і виводить інформативне повідомлення про відсутність збігів у базі або проблеми зі з'єднанням. Якщо ж транзакція

проходить успішно, стан завантаження знімається, а валідовані дані передаються до модулів побудови діаграм. Такий жорсткий контроль над усіма фазами запиту повністю виключає відображення порожніх або неробочих екранів. Візуалізацію проміжного стану очікування даних за допомогою пульсуючих індикаторів наведено на рисунку 3.1.

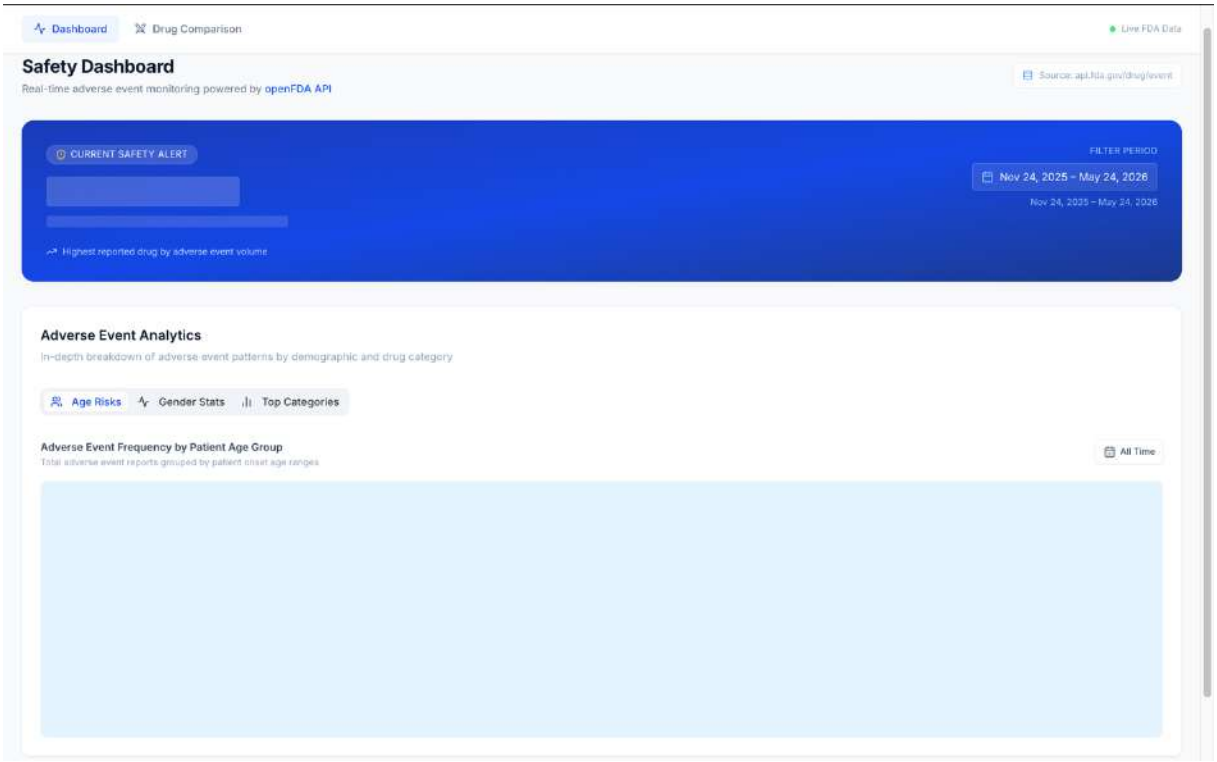


Рисунок 3.1 – Візуалізація стану завантаження (skeleton loader) під час виконання асинхронного запиту до сервера openFDA

Для забезпечення детермінованої поведінки графічного інтерфейсу та формалізації переходів між усіма описаними фазами асинхронного обміну даними було розроблено відповідну логіку. Вона чітко розмежовує стани очікування, обробки винятків та готовності масивів, що дозволяє повністю уникнути непередбачуваної поведінки застосунку під час тривалих мережевих затримок. Завдяки такому підходу клієнтська архітектура стає максимально відмовостійкою, гарантуючи безперервний та інформативний зворотний зв'язок

для кінцевого користувача. Весь цей життєвий цикл React-компонента від моменту початкової ініціалізації до успішного рендерингу медичної статистики наочно представлено на UML-діаграмі станів (рисунок 3.2).

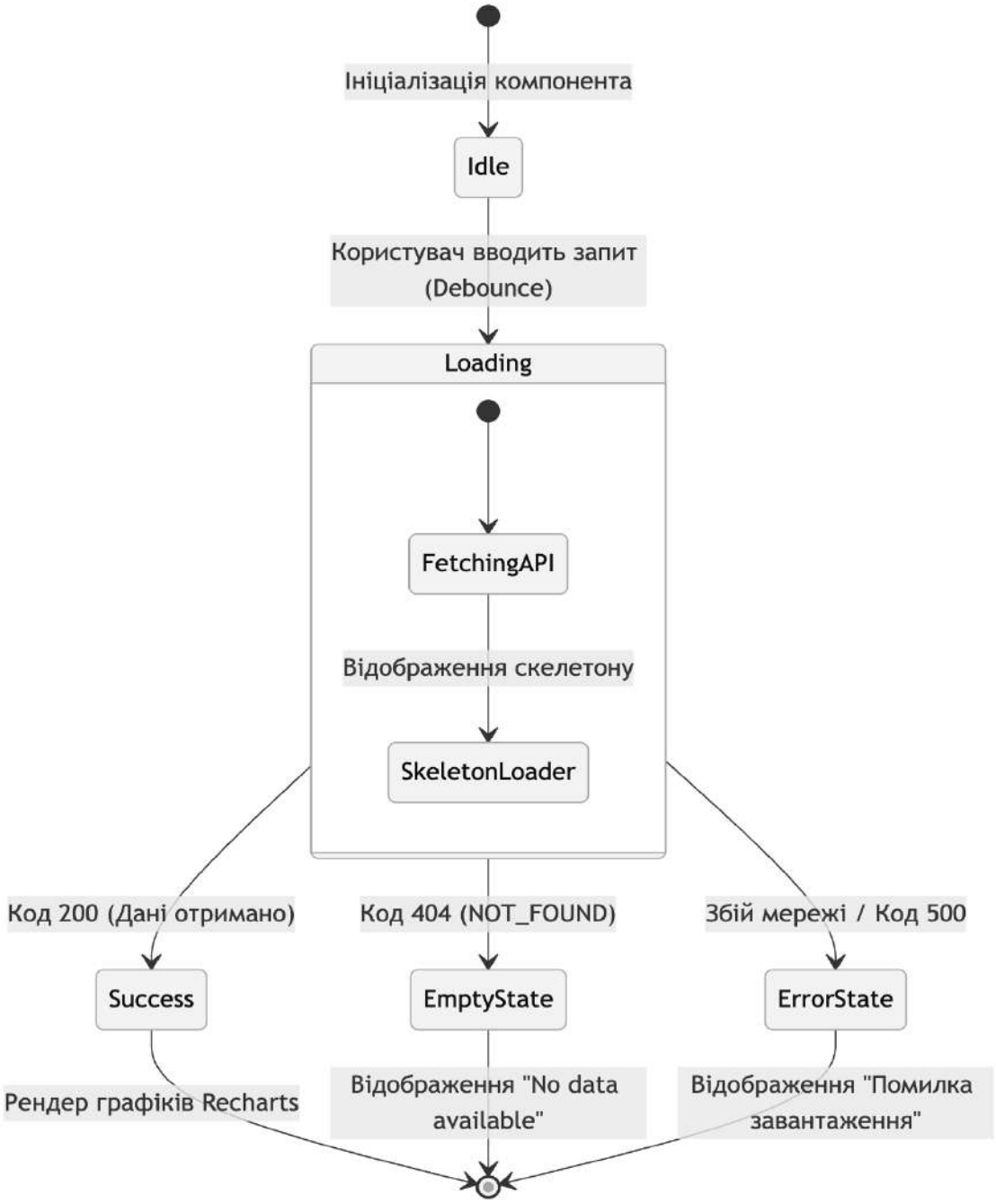


Рисунок 3.2 – UML-діаграма станів життєвого циклу компонента при взаємодії з віддаленим openFDA API

Комплексне управління цими станами формує безперервний та передбачуваний користувацький досвід. Навіть у випадках критичного уповільнення роботи урядових серверів або тимчасової втрати інтернет-з'єднання клієнтський інтерфейс ніколи не завмирає у стані невизначеності. Дослідник завжди отримує чіткий візуальний зворотний зв'язок: індикацію процесу пошуку, валідовану аналітичну панель або структуроване повідомлення про помилку мережевого обміну. Такий підхід повністю виключає сценарії раптового «зависання» сторінки браузера, зберігаючи високу чуйність платформи під час роботи з найважчими вибірками фармаконагляду. Приклад обробки стану відсутності статистичних даних для обраного критерію показано на рисунку 3.3

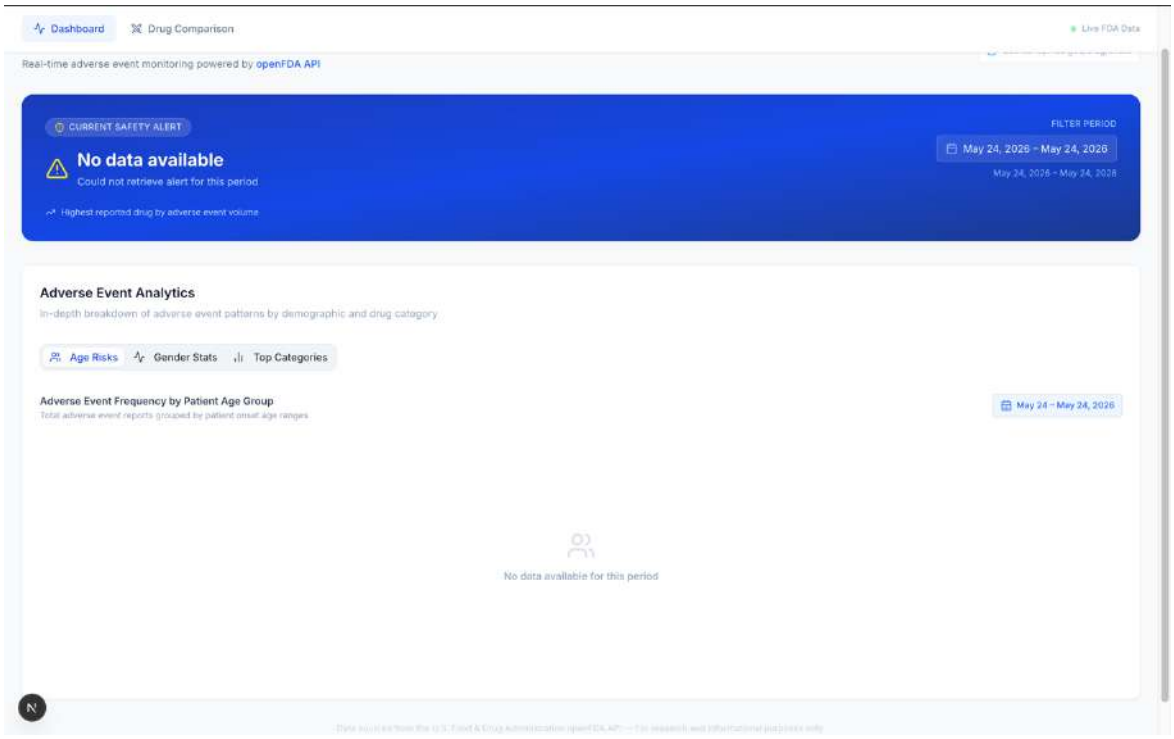


Рисунок 3.3 – Інтерфейс користувача за умови відсутності релевантних клінічних звітів (Empty State)

Проєктування інтелектуальних форм пошуку медикаментів зіштовхується з фундаментальною інженерною проблемою – жорсткими обмеженнями частоти звернень (rate limits) до відкритого API. Якщо архітектура застосунку

передбачатиме відправку нового HTTP-запиту при кожному натисканні клавіші користувачем, система згенерує десятки викликів за лічені секунди. Такий наївний алгоритм миттєво вичерпає виділені квоти FDA, що неминуче призведе до блокування IP-адреси клієнта або інвалідації авторизаційного ключа. Це перетворить аналітичний інструмент на повністю непрацездатний продукт ще на етапі введення назви діючої речовини.

Вирішення проблеми надлишкового мережевого трафіку досягається шляхом впровадження алгоритмічного патерну штучної затримки (Debounce) [34]. Програмна логіка перехоплює подію зміни текстового поля, проте свідомо відкладає фактичну ініціалізацію функції «fetch». Система активує таймер, налаштований зазвичай на п'ятсот мілісекунд. Якщо протягом цього вікна часу дослідник продовжує вводити наступні символи, попередній таймер скасовується і відлік починається заново. Мережевий запит формується і відправляється до реєстру лише після того, як користувач робить усвідомлену паузу. Ця проста оптимізація дозволяє замінити серію з десятків проміжних звернень на один цільовий запит із повноцінною назвою препарату.

Ефективність підсистеми автодоповнення безпосередньо залежить від синтаксичних можливостей самого пошукового рушія openFDA. Для реалізації гнучкого пошуку застосовується вбудований механізм символів підстановки (Wildcards). Під час програмного формування URL-запиту введений користувачем текст автоматично обгортається символами зірочки, перетворюючи стандартний критерій пошуку на конструкцію формату «brand_name:"*введений_текст*"». Такий синтаксис змушує сервер знаходити клінічні звіти навіть за частковим збігом або фрагментом складної хімічної назви. Інтеграція алгоритму затримки у тісній синергії з механізмом часткового збігу забезпечує спеціаліста миттєвими та релевантними підказками препаратів, повністю усуваючи ризик перевантаження урядової хмарної інфраструктури. езультат роботи підсистеми автодоповнення з використанням описаних алгоритмів наведено на рисунку 3.3.

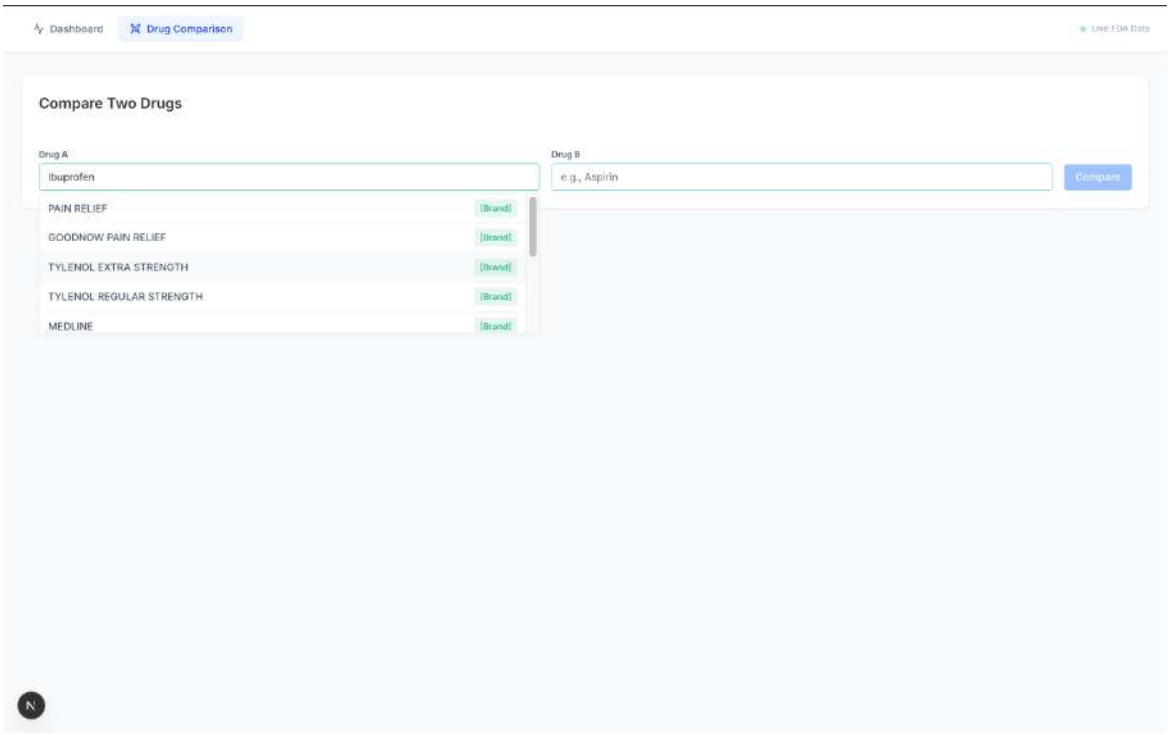


Рисунок 3.3 – Робота модуля розумного пошуку та автодоповнення назв медичних препаратів

3.3 Технічна реалізація аналітичних компонентів

Процес трансформації оброблених медичних даних у наочні візуальні панелі спирається на можливості екосистеми Recharts. Ця бібліотека використовує декларативний підхід до створення SVG-графіки, що ідеально узгоджується з компонентною моделлю фреймворку React. Технічна реалізація візуального шару починається з передачі попередньо сплющених (flat arrays) та суворо типізованих масивів інформації через властивості (props) безпосередньо до кореневих компонентів діаграм, таких як «BarChart» або «PieChart». Завдяки впровадженню TypeScript-інтерфейсів на попередніх етапах архітектура гарантує, що графічні примітиви отримують виключно валідовані об'єкти. Структури, які надходять до діаграм, вже позбавлені аномалій формату «null» або вкладених нерозпізнаних об'єктів, що повністю виключає ризик виникнення

помилок виконання (runtime errors) під час математичного розрахунку координат візуальних осей.

Сучасний клінічний інтерфейс вимагає бездоганної адаптивності під різноманітні пристрої, від широкоформатних моніторів у дослідницьких лабораторіях до екранів портативних ноутбуків. Реалізація цієї вимоги досягається шляхом інкапсуляції кожної діаграми у спеціалізований компонент-обгортку «ResponsiveContainer». Цей структурний елемент безперервно відстежує доступну ширину та висоту батьківського блоку в об'єктній моделі документа. При зміні розмірів вікна браузера обгортка автоматично перераховує фізичні габарити внутрішнього SVG-полотна та ініціює плавне перемальовування графіка. Такий інженерний підхід дозволяє відмовитися від написання складних медіа-запитів у каскадних таблицях стилів, гарантуючи, що аналітичні панелі Dashboard завжди зберігатимуть правильні пропорції незалежно від роздільної здатності екрана дослідника.

Окремим викликом при проєктуванні щільних статистичних інтерфейсів є проблема надмірного візуального шуму. Відображення абсолютних числових значень частоти побічних ефектів безпосередньо на секторах кругових діаграм або над кожним стовпчиком гістограми створює когнітивне перевантаження. Вирішення цієї проблеми реалізовано через розробку кастомних підказок (Custom Tooltips) [**Error! Reference source not found.**]. Програмна логіка системи приховує точні математичні метрики та повні медичні терміни з класифікатора MedDRA всередині спеціального інтерактивного компонента. Цей блок динамічно монтується у віртуальне дерево елементів виключно у момент наведення курсора миші на конкретний сегмент графіка. Кастомна підказка використовує утилітарні класи Tailwind CSS 4 для дотримання загальної стилістики застосунку, надаючи фахівцю вичерпну інформацію у спливаючому вікні, залишаючи при цьому основний простір діаграми мінімалістичним та зручним для швидкого зчитування загальних тенденцій.

Перехід від теоретичної алгоритмічної моделі до практичного React-компонента в модулі Drug Comparison вимагає побудови чіткої архітектури управління локальним станом. Фізична структура інтерактивного блоку базується на наявності двох незалежних текстових полів введення, кожне з яких прив'язане до власного стану через хук «useState». Ця ізоляція гарантує, що процес автодоповнення та вибору діючої речовини для першого медикаменту жодним чином не впливає на конфігурацію другого. Ініціалізація паралельного мережевого запиту відбувається виключно після натискання спільної кнопки порівняння. Цей архітектурний паттерн запобігає передчасній відправці неповних запитів до серверів openFDA та синхронізує початок асинхронної транзакції для обох обраних препаратів.

Після успішного отримання та базової десеріалізації двох масивів JSON-відповідей запускається оптимізований алгоритм пошуку перетинів із лінійною часовою складністю $O(n+m)$. Програмна логіка, інкапсульована у відповідному користувацькому хуку, формує хеш-таблицю на основі масиву першого препарату, де ключем виступає поле «reactionmeddrapt». Під час ітерації масиву другого препарату алгоритм миттєво ідентифікує збіги та конструює зведений плоский масив об'єктів формату «CommonSymptom». Кожен елемент цього фінального масиву містить уніфіковану назву побічного ефекту та дві суворо типізовані числові метрики, що відображають абсолютну частоту фіксації цього симптому для першого та другого лікарських засобів відповідно.

Сформований масив перетинів передається безпосередньо у властивості графіка порівняння, який програмно конструюється як єдиний екземпляр компонента «BarChart». Ключова особливість цієї реалізації полягає у використанні одразу двох візуальних елементів «Bar» всередині спільної координатної сітки. Кожен із цих елементів динамічно прив'язується до відповідного ключа даних (Drug A та Drug B). На етапі рендерингу система автоматично призначає стовпчикам заздалегідь визначені семантичні кольори, що формує наочну паралельну гістограму. Дослідник отримує можливість

миттєво візуально оцінити різницю у профілях безпеки. Довжина парних індикаторів на єдиній осі чітко демонструє, який із двох аналізованих медикаментів має вищу ймовірність викликати специфічний спільний симптом, трансформуючи складну математичну агрегацію у зрозумілий та ефективний інструмент клінічної аналітики.

3.4 Опис функціональних можливостей розробленого продукту

Фінальний етап розробки програмного продукту супроводжується комплексним аналізом користувацького досвіду та перевіркою коректності функціонування ключових модулів. Сценарій взаємодії фахівця з системою починається на головній сторінці, де дослідник вводить назву цільового лікарського засобу в інтелектуальне поле пошуку. Після завершення введення та відпрацювання алгоритму затримки застосунок ініціює асинхронне звернення до відкритого реєстру. Успішне отримання масивів даних запускає процес десеріалізації, після чого система миттєво відмальовує інтерактивний дашборд. Візуальна панель структурує отриману статистику, розділяючи її на логічні блоки демографічних ризиків та категорій побічних ефектів, що дозволяє користувачеві безперешкодно переходити від загального огляду до глибокого аналізу конкретних клінічних аномалій.

Критичним елементом проактивного моніторингу в межах згенерованого дашборду виступає підсистема «Safety Alert Banner». Її алгоритм функціонує у фоновому режимі, здійснюючи безперервний аналіз трансформованих результатів на наявність специфічних клінічних маркерів у режимі реального часу. Програмна логіка сканує масив побічних реакцій, розраховуючи питому вагу серйозних ускладнень або летальних наслідків відносно загальної кількості зафіксованих репортів. Якщо обчислений показник перевищує заздалегідь встановлений статистичний поріг, система динамічно генерує тривожне візуальне сповіщення. Цей компонент закріплюється у верхній частині екрана,

використовуючи сигнальні кольори для миттєвого привернення уваги фахівця до потенційної небезпеки медикаменту. Такий інженерний підхід трансформує застосунок із пасивного засобу візуалізації на інтелектуального помічника, здатного самостійно виявляти приховані загрози.

Процес технічної апробації вебзастосунку підтвердив високу ефективність обраних архітектурних рішень. Проведення незалежного аудиту продуктивності за допомогою інструментарію Google Lighthouse продемонструвало максимальні показники за критеріями швидкості завантаження та чуйності інтерфейсу [37]. Досягнення таких результатів стало прямим наслідком стратегічного перенесення обчислювального навантаження на серверну інфраструктуру завдяки використанню серверних компонентів фреймворку Next.js. Браузер клієнта отримує мінімально необхідний обсяг JavaScript-коду, що критично важливо при роботі з масивними обсягами медичної статистики. Свій вагомий внесок у загальну оптимізацію зробила й інтеграція компілятора Tailwind CSS 4, який згенерував екстремально легкий стильовий бандл, відсікши всі невикористані CSS-правила.

Окрема увага під час тестування приділялася здатності платформи витримувати нетипові сценарії та граничні випадки у процесі мережевої взаємодії. Звернення до відкритих урядових серверів неминуче супроводжується ситуаціями, коли інформація про специфічний препарат фізично відсутня в базі даних, через що API Управління з продовольства і медикаментів США повертає статус 404. Архітектура розробленого застосунку успішно проходить цей стрес-тест – система не перериває цикл виконання і не демонструє користувачеві неробочий інтерфейс. Програмна логіка коректно перехоплює мережеву помилку на рівні ізольованого компонента і плавно відображає інтерактивний порожній стан. Це дозволяє досліднику миттєво зрозуміти причину відсутності графіків та продовжити пошук інших діючих речовин без необхідності перезавантаження сторінки.

Фінальним етапом апробації стала перевірка надійності механізмів типізації під час обробки екстремально великих та фрагментованих датасетів. Механізм валідації на основі декларативних інтерфейсів TypeScript довів свою абсолютну ефективність в умовах стрес-тестування. Під час штучної подачі на вхід системи пошкоджених JSON-об'єктів, де ключові поля демографії або назви медикаментів були замінені на значення «null» або «undefined», алгоритми мапінгу безпомилково фільтрували невалідні записи. Жоден пошкоджений елемент не зміг подолати захисний бар'єр і потрапити до властивостей графічних компонентів. Це повністю усунуло ризик виникнення критичних помилок під час рендерингу SVG-графіки, підтвердивши експлуатаційну стійкість розробленої платформи для інтенсивного повсякденного використання.

3.5 Оцінка якості та перспективи подальшого розвитку системи

Розроблена вебплатформа для аналізу безпеки лікарських засобів на основі відкритого програмного інтерфейсу успішно пройшла всі етапи технічної валідації та повністю готова до практичної експлуатації. Система демонструє високу стабільність під час безперервної роботи з масивними масивами клінічних репортів бази FAERS. Реалізований модуль візуального порівняння медикаментів ефективно вирішує головну дослідницьку проблему, миттєво трансформуючи складні багатовимірні JSON-структури в інтуїтивно зрозумілі паралельні гістограми. Сувора типізація даних на всіх етапах трансформації гарантує повну відсутність помилок виконання, що підтверджує високу архітектурну якість створеного програмного продукту. Фахівець із фармаконагляду отримує надійний інструмент, здатний проактивно сигналізувати про небезпечні клінічні маркери без зависань чи структурних збоїв.

Поточна архітектура діє як надійний фундамент для подальшого масштабування аналітичної екосистеми. Пріоритетним вектором розвитку

проєкту є розширення джерел медичної статистики для забезпечення крос-континентального аналізу побічних реакцій. Інтеграція платформи з відкритими реєстрами Європейського агентства з лікарських засобів (EMA – European Medicines Agency) [38] становить логічний наступний крок у цьому напрямку. Об'єднання американських та європейських клінічних датасетів у межах єдиного інтерфейсу дозволить дослідникам виявляти глобальні тренди безпеки препаратів, нівелюючи регіональні аномалії або специфічні генетичні вразливості окремих популяцій хворих. Такий гібридний підхід суттєво підвищить рівень об'єктивності статистичних зрізів.

Функціональне масштабування платформи передбачає впровадження повноцінної системи автентифікації користувачів. Ця модернізація надасть клініцистам можливість створювати персональні облікові записи для безпечного збереження історії складних пошукових запитів та результатів багатокрокових порівняльних аналізів. Додатково клінічна практика диктує необхідність інтеграції модуля експорту згенерованих звітів. Формування готових аналітичних витягів у форматі PDF або вивантаження відфільтрованих «сирих» даних у вигляді CSV-файлів дозволить спеціалістам безперешкодно інтегрувати отримані статистичні висновки у медичні презентації, наукові статті або офіційну регуляторну документацію. Розширення експортних можливостей остаточно перетворить вебзастосунок на комплексну робочу станцію дослідника.

3.6 Висновки до третього розділу

Програмна реалізація аналітичної платформи підтвердила інженерну доцільність обраного технологічного стека. Ініціалізація середовища розробки на базі фреймворку Next.js з архітектурою App Router забезпечила оптимальний баланс між швидким серверним рендерингом базової розмітки та гнучкою клієнтською інтерактивністю. Перехід до парадигми конфігурації стилів

безпосередньо через CSS-змінні у Tailwind CSS 4, у поєднанні з інтеграцією нестилізованих примітивів shadcn, дозволив побудувати суворий, інклюзивний та стилістично консистентний медичний інтерфейс. Відмова від традиційних громіздких UI-бібліотек критично зменшила обсяг збірочного файлу, гарантуючи миттєве завантаження візуальних панелей.

Організація мережевої взаємодії з хмарним реєстром openFDA продемонструвала високу ефективність застосованих асинхронних патернів. Впровадження алгоритмічної затримки Debounce докорінно знизило навантаження на урядові сервери, запобігаючи вичерпанню квот під час активного використання поля пошуку дослідником. Синергійне застосування цього механізму разом із синтаксисом Wildcards забезпечило функціонування гнучкого автодоповнення, здатного знаходити препарати за частковим збігом назви. Система надійно обробляє HTTP-запити, елегантно перехоплюючи статус 404 та відмальовуючи інформативні порожні стани замість аварійного завершення роботи застосунку.

Інтеграція декларативної бібліотеки Recharts дозволила перетворити сирі статистичні агрегації на чуйні візуальні панелі. Оптимізація модуля порівняння через використання хеш-таблиць знизила алгоритмічну складність пошуку спільних симптомів до $O(n+m)$, забезпечивши миттєвий рендер паралельних гістограм безпосередньо у браузері клієнта. Комплексна технічна апробація підтвердила абсолютну стійкість платформи до пошкоджених JSON-структур завдяки жорсткому механізму мапінгу через декларативні TypeScript-інтерфейси. Розроблений програмний продукт повністю задовольняє поставлені функціональні та нефункціональні вимоги, виступаючи надійним високопродуктивним інструментом для фахівців із фармаконагляду.

ВИСНОВКИ

Виконане кваліфікаційне дослідження становить цілісну науково-практичну роботу, у межах якої було успішно розв'язано актуальну інженерну задачу створення спеціалізованого вебзастосунку для аналізу медичних даних. Поставлена на початку роботи мета щодо розробки високоефективної платформи для інтерактивного порівняння профілів безпеки лікарських засобів досягнута повною мірою. Усі сформульовані дослідницькі та технічні завдання виконані в повному обсязі, що дозволило перетворити теоретичну концепцію на готовий до експлуатації програмний продукт.

Проведений глибокий аналіз предметної області переконливо засвідчив наявність системної проблеми перевантаження фахівців неструктурованою медичною інформацією. Відкриття доступу до урядових реєстрів рівня openFDA кардинально змінило парадигму фармаконагляду, проте оголило критичну нестачу сучасних інструментів для обробки таких масивів. Дослідження підтвердило, що сирі відповіді у форматі JSON вимагають принципово нових підходів до візуалізації, здатних мінімізувати когнітивне навантаження на клініциста. Вибір технологічного стека на базі Next.js, React 19 та TypeScript виявився найбільш перспективним та архітектурно обґрунтованим рішенням для розробки медичних інформаційних систем нового покоління. Використання інноваційного компілятора Tailwind CSS 4 забезпечило створення мінімалістичного медичного інтерфейсу без надлишкового стильового коду, що є критичним для фокусування уваги на клінічних сигналах.

Процес системного проєктування дозволив сформувати жорсткі функціональні вимоги та перевести їх у площину чітких інженерних абстракцій. Побудовані моделі уніфікованої мови моделювання, зокрема діаграми прецедентів та послідовності [**Error! Reference source not found.**], забезпечили безпомилкове розуміння життєвого циклу складних асинхронних транзакцій. Стратегічним рішенням на цьому етапі стала відмова від розгортання власної монолітної бази даних на користь архітектурної моделі безсерверного

агрегатора. Віддалений хмарний реєстр Управління з продовольства і медикаментів США був успішно інтегрований як єдине джерело правди. Розробка суворих декларативних TypeScript-інтерфейсів відіграла фундаментальну роль у цій конфігурації. Створені типи та функції-мапери утворили надійний контур безпеки, який унеможливлює потрапляння критичних помилок виконання у візуальні компоненти під час обробки непередбачуваних ієрархічних структур.

Безпосередня програмна реалізація аналітичних модулів підтвердила життєздатність закладених архітектурних ідей. Справжнім інженерним проривом у межах проєкту стала розробка та імплементація оптимізованого алгоритму для модуля порівняння препаратів. Перехід від наївного обчислення до використання механізмів на основі хеш-таблиць дозволив знизити часову складність пошуку спільних симптомів до лінійного показника $O(n+m)$. Це алгоритмічне рішення гарантує миттєве формування масивів перетинів безпосередньо у браузері клієнта, повністю виключаючи зависання інтерфейсу. Паралельно було успішно розв'язано проблему надмірного мережевого трафіку. Впровадження патерну штучної затримки у комбінації із синтаксисом часткового збігу захистило платформу від вичерпання серверних квот, забезпечивши при цьому безперебійну роботу інтелектуального автодоповнення.

Комплексна технічна апробація розробленої системи продемонструвала видатні експлуатаційні характеристики. Незалежний аудит за допомогою інструментарію Google Lighthouse зафіксував максимальні метрики продуктивності, що є прямим наслідком раціонального використання серверних компонентів Next.js та легкої архітектури відмальовування сторінок. Платформа довела свою абсолютну стійкість до стресових сценаріїв під час обробки граничних випадків. Отримання статусу про відсутність даних від відкритого API коректно перехоплюється системою з подальшим виведенням інтерактивного порожнього стану. Водночас механізми статичного аналізу типів безпомилково фільтрують пошкоджені JSON-об'єкти з пропущеними полями

демографії чи найменувань, виключаючи ризик руйнування SVG-діаграм бібліотеки Recharts.

Спроєктована архітектура володіє значним технологічним потенціалом для подальшого масштабування та розширення аналітичних можливостей. Найбільш перспективним вектором розвитку системи вбачається інтеграція з європейським реєстром медичних даних для забезпечення крос-континентального моніторингу профілів безпеки лікарських засобів. Такий крок дозволить виявляти глобальні епідеміологічні тренди, незалежні від регіональних особливостей протоколів лікування. Паралельно планується впровадження спеціалізованих модулів експорту згенерованих звітів у стандартизовані формати, що надасть фахівцям змогу інтегрувати отримані візуалізації безпосередньо у клінічні презентації або нормативну документацію.

Розроблена аналітична вебплатформа є повністю завершеним, високопродуктивним та відмовостійким програмним продуктом. Застосування передових методів алгоритмічної оптимізації у зв'язці з сучасним технологічним стеком дозволило створити унікальне середовище для паралельного зіставлення медикаментів. Результати цього дипломного проєкту мають беззаперечне практичне значення як для інформаційних технологій у контексті обробки великих даних, так і для медичної галузі, надаючи спеціалістам із фармаконагляду надійний інструмент для швидкого та об'єктивного прийняття критичних клінічних рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1 openFDA API Endpoint Reference. URL: https://open.fda.gov/apis/drug/event/ (дата звернення: 11.01.2026).

2 Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

3 Introducing JSON / JSON.org. URL: https://www.json.org/json-en.html (дата звернення: 02.02.2026).

4 What is a RESTful API? / Amazon Web Services. URL: https://aws.amazon.com/what-is/restful-api/ (дата звернення: 02.02.2026).

5 Flanagan D. JavaScript: The Definitive Guide. O'Reilly Media.

6 Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma та ін. Addison-Wesley.

7 MedDRA (Medical Dictionary for Regulatory Activities) / International Council for Harmonisation (ICH). URL: https://www.meddra.org/ (дата звернення: 05.02.2026).

8 ECMAScript Language Specification / ECMA International. URL: https://tc39.es/ecma262/ (дата звернення: 05.02.2026).

9 Node.js Documentation / OpenJS Foundation. URL: https://nodejs.org/en/docs/ (дата звернення: 20.02.2026).

10 React Hooks API Reference / Meta Open Source. URL: https://react.dev/reference/react (дата звернення: 20.02.2026).

11 Promise.allSettled() / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/allSettled (дата звернення: 05.03.2026).

12 Recharts: A composable charting library built on React components. URL: https://recharts.github.io/en-US/ (дата звернення: 05.03.2026).

13 Chart.js Documentation / Chart.js. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 05.03.2026).

14 Getting started with D3 / D3 by Observable. URL: https://d3js.org/getting-started (дата звернення: 05.03.2026).

15 Fetch API / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 11.03.2026).

16 HTTP response status codes / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (дата звернення: 11.03.2026).

17 Documentation – Utility Types / TypeScript. URL: https://www.typescriptlang.org/docs/handbook/utility-types.html (дата звернення: 11.03.2026).

18 Tailwind CSS v4.0. CSS-first configuration / Tailwind Labs. URL: https://tailwindcss.com/blog/tailwindcss-v4 (дата звернення: 14.03.2026).

19 What is serverless computing? / Cloudflare. URL: https://www.cloudflare.com/learning/serverless/what-is-serverless/ (дата звернення: 14.03.2026).

20 TypeScript Handbook. Everyday Types / Microsoft. URL: https://www.typescriptlang.org/docs/handbook/2/everyday-types.html (дата звернення: 15.03.2026).

21 Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability.

22 Grigorik I. High Performance Browser Networking. O'Reilly Media.

23 FDA Adverse Event Reporting System (FAERS) Public Dashboard. URL: https://www.fda.gov/drugs/questions-and-answers-fdas-adverse-event-reporting-system-faers/fda-adverse-event-reporting-system-faers-public-dashboard (дата звернення: 15.03.2026).

24 The event loop / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Event_loop (дата звернення: 15.03.2026).

25 Introduction to the DOM / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

26 Radix UI: Primitives for building accessible, high-quality web apps. URL: https://www.radix-ui.com/primitives/docs/overview/introduction (дата звернення: 15.03.2026).

27 shadcn/ui. Beautifully designed components that you can copy and paste into your apps. URL: https://ui.shadcn.com/docs (дата звернення: 15.03.2026).

28 Vercel Documentation / Vercel. URL: https://vercel.com/docs (дата звернення: 15.03.2026).

29 Web Content Accessibility Guidelines (WCAG) 2.1 / W3C. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 21.03.2026).

30 Introduction to Algorithms / T. H. Cormen та ін. MIT Press.

31 Using Web Workers – Web APIs / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers (дата звернення: 21.03.2026).

32 Next.js App Router Documentation / Vercel. URL: https://nextjs.org/docs/app (дата звернення: 21.03.2026).

33 React 19 Reference Overview / Meta Open Source. URL: https://react.dev/blog/2024/04/25/react-19 (дата звернення: 21.03.2026).

34 Lucide React Guide / Lucide. URL: https://lucide.dev/guide/react/ (дата звернення: 01.06.2026).

35 Debouncing and Throttling Explained Through Examples / CSS-Tricks. URL: https://css-tricks.com/debouncing-throttling-explained-examples/ (дата звернення: 21.03.2026).

36 Tufte E. R. The Visual Display of Quantitative Information. Graphics Press.

37 Google Lighthouse Configuration and Performance Scoring / Google for Developers. URL: https://developer.chrome.com/docs/lighthouse/performance (дата звернення: 22.03.2026).

38 European Medicines Agency (EMA). Access to EudraVigilance data. URL: https://www.ema.europa.eu/en/human-regulatory/research-development/pharmacovigilance/eudravigilance/access-eudravigilance-data (дата звернення: 04.04.2026).

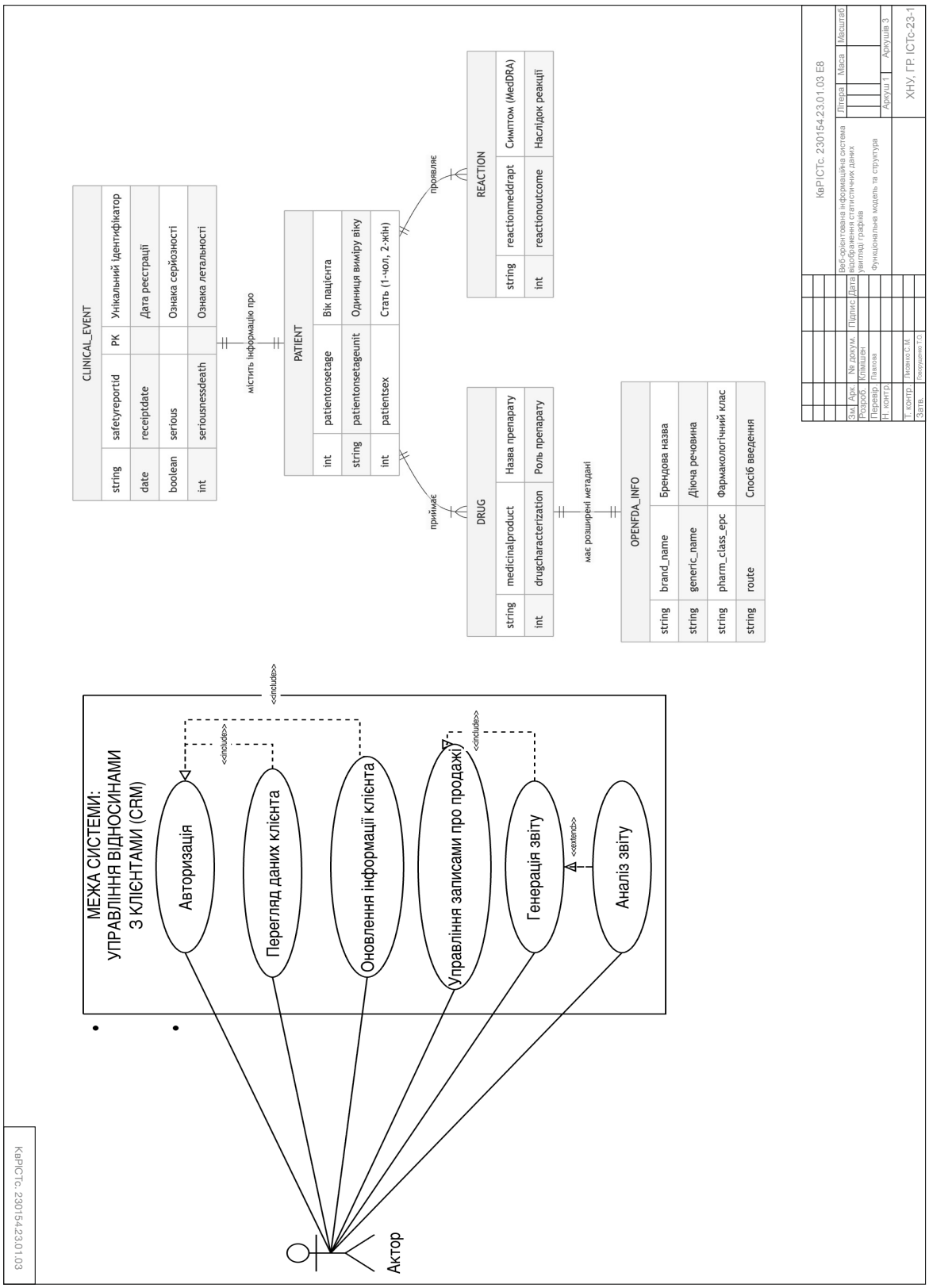
39 Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional.

40 10 Usability Heuristics for User Interface Design / Nielsen Norman Group. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 04.04.2026).

ДОДАТОК А

(обов’язковий)

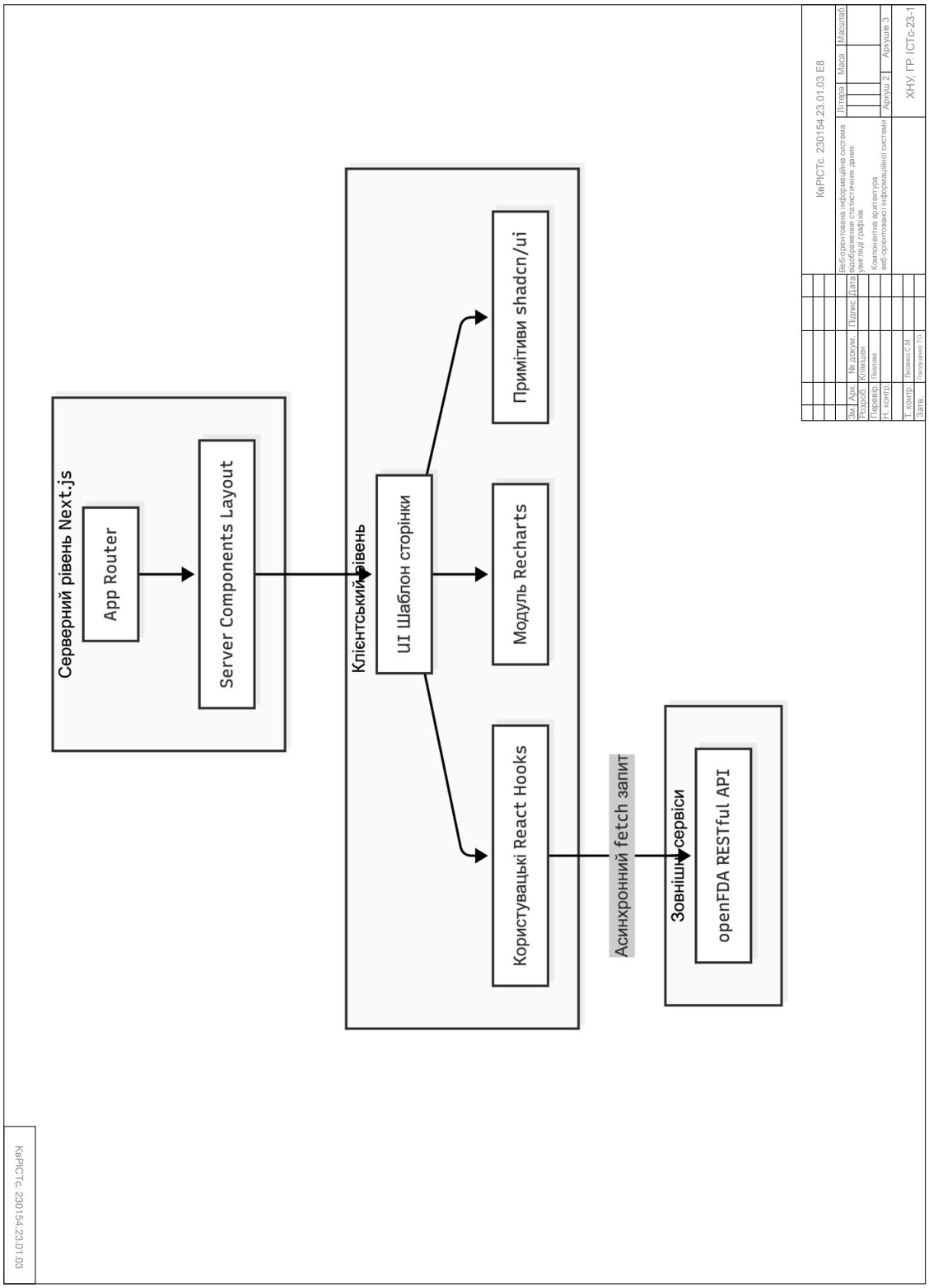
Функціональна модель та структура



ДОДАТОК Б

(обов’язковий)

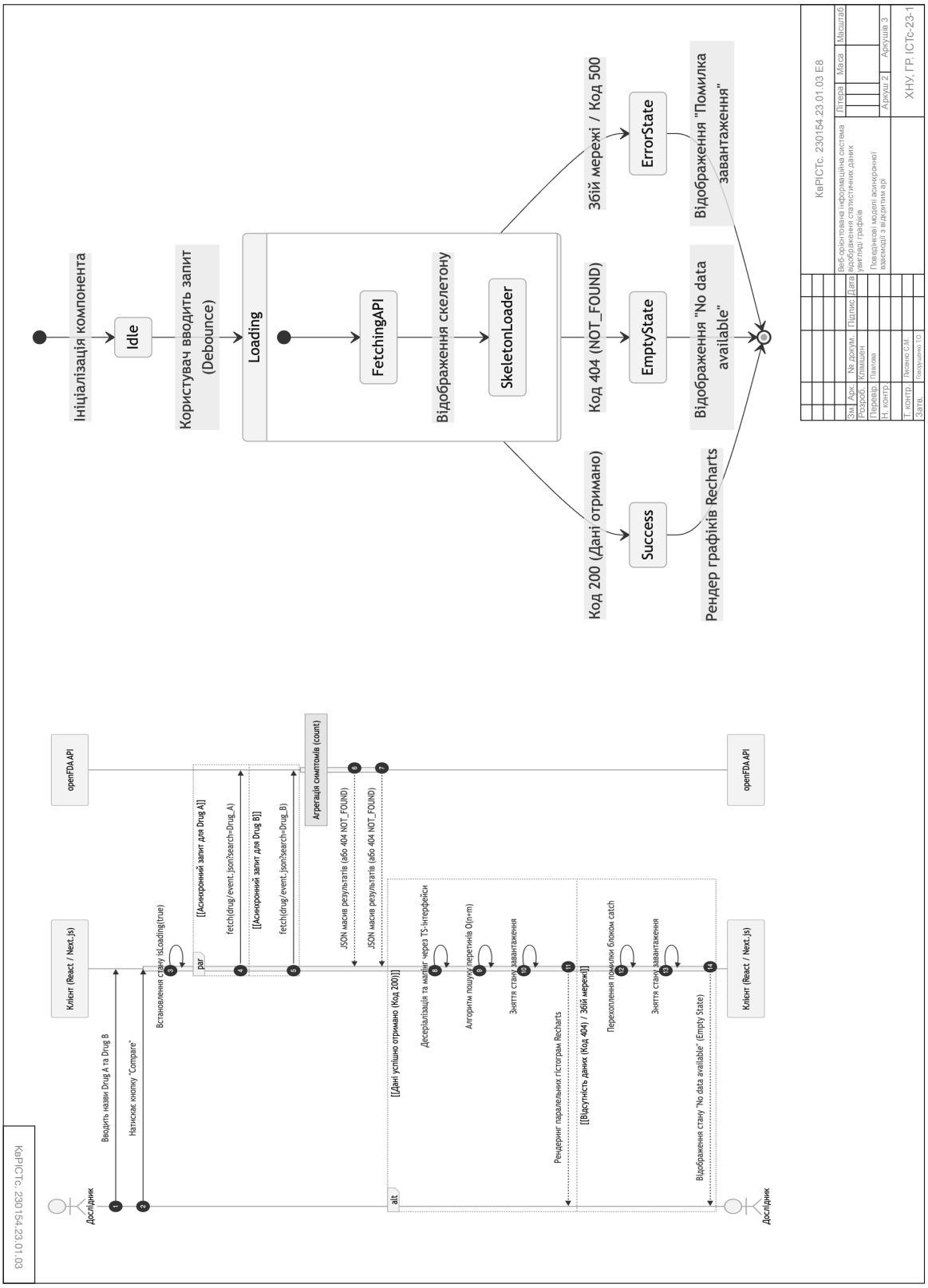
Компонентна архітектура веб-орієнтованої інформаційної системи



ДОДАТОК В

(обов’язковий)

Поведінкові моделі асинхронної взаємодії з відкритим арі



ДОДАТОК Г

(обов’язковий)

Візуалізація результатів програмної реалізації інформаційної системи

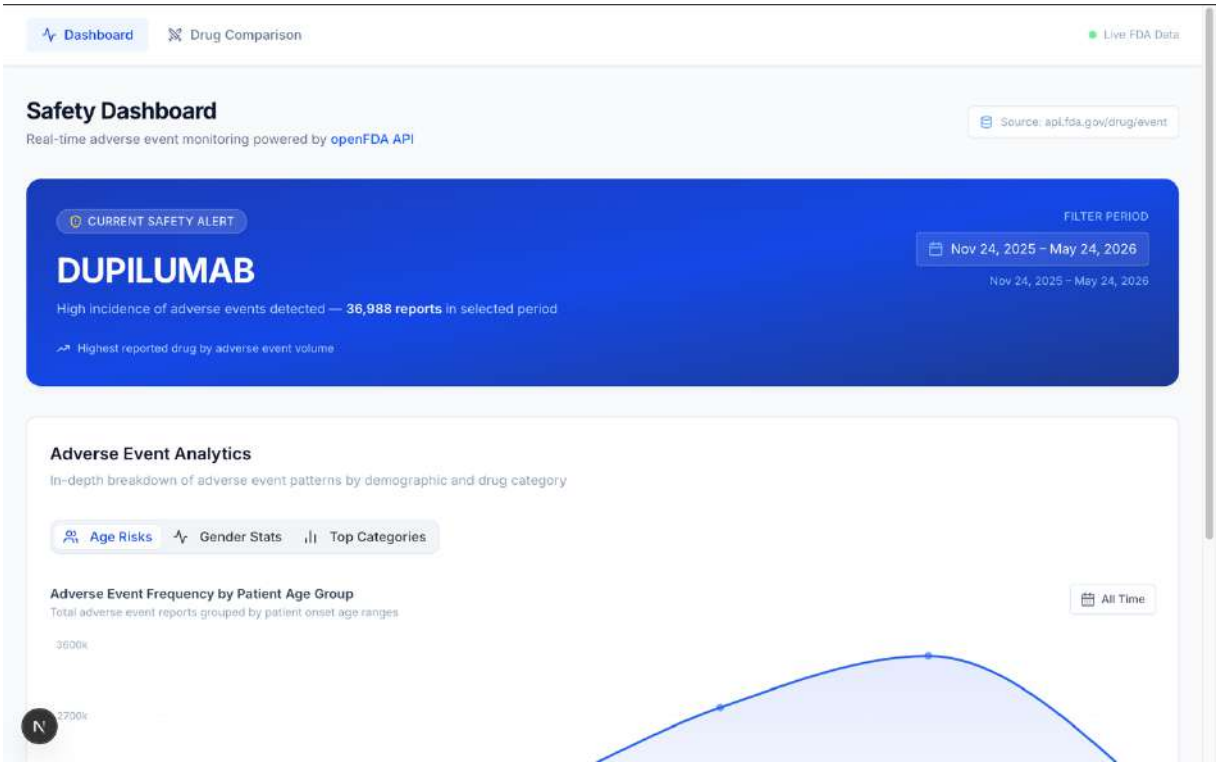


Рисунок Г.1 – Головна панель інформаційної системи з активним модулем сповіщень про критичні ризики безпеки

Рисунок Г.2 – Аналітичний модуль: візуалізація частоти виникнення побічних реакцій залежно від вікової групи пацієнтів

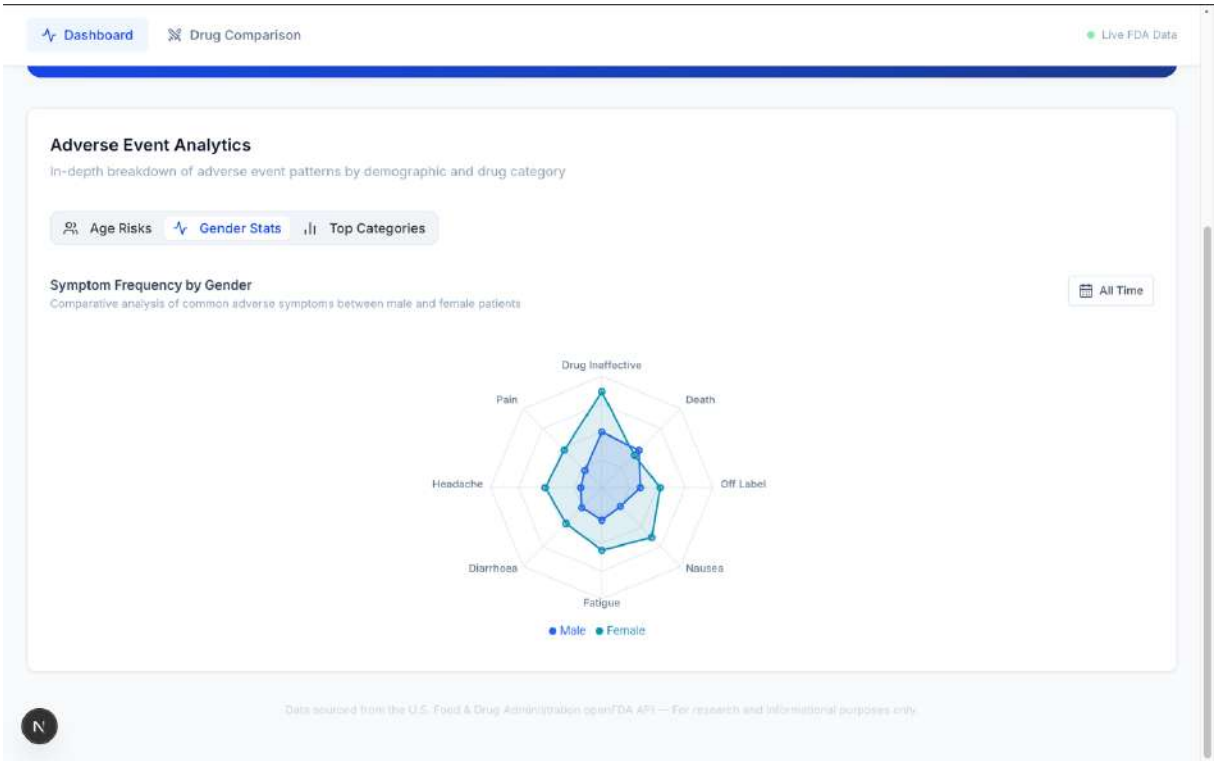


Рисунок Г.3 – Аналітичний модуль: порівняльний аналіз профілів побічних симптомів за статтю

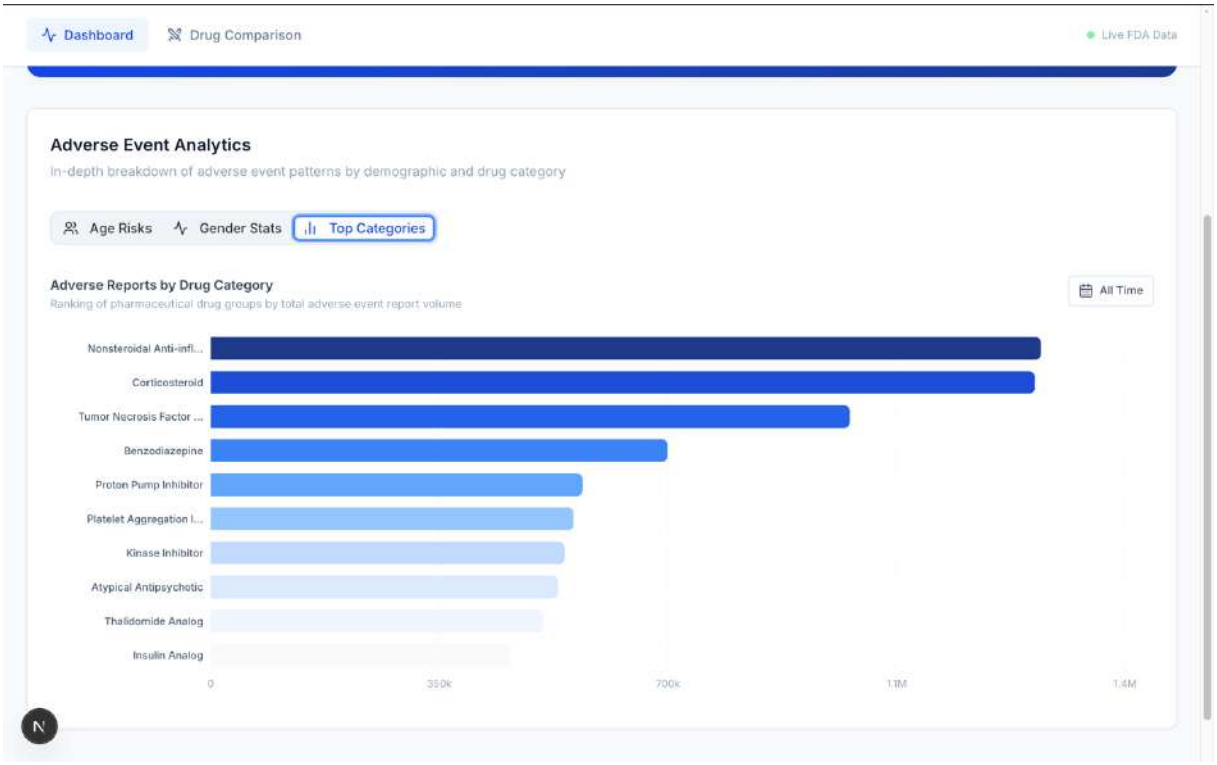


Рисунок Г.4 – Аналітичний модуль: рейтинг фармакологічних груп лікарських засобів за загальною кількістю зареєстрованих клінічних звітів

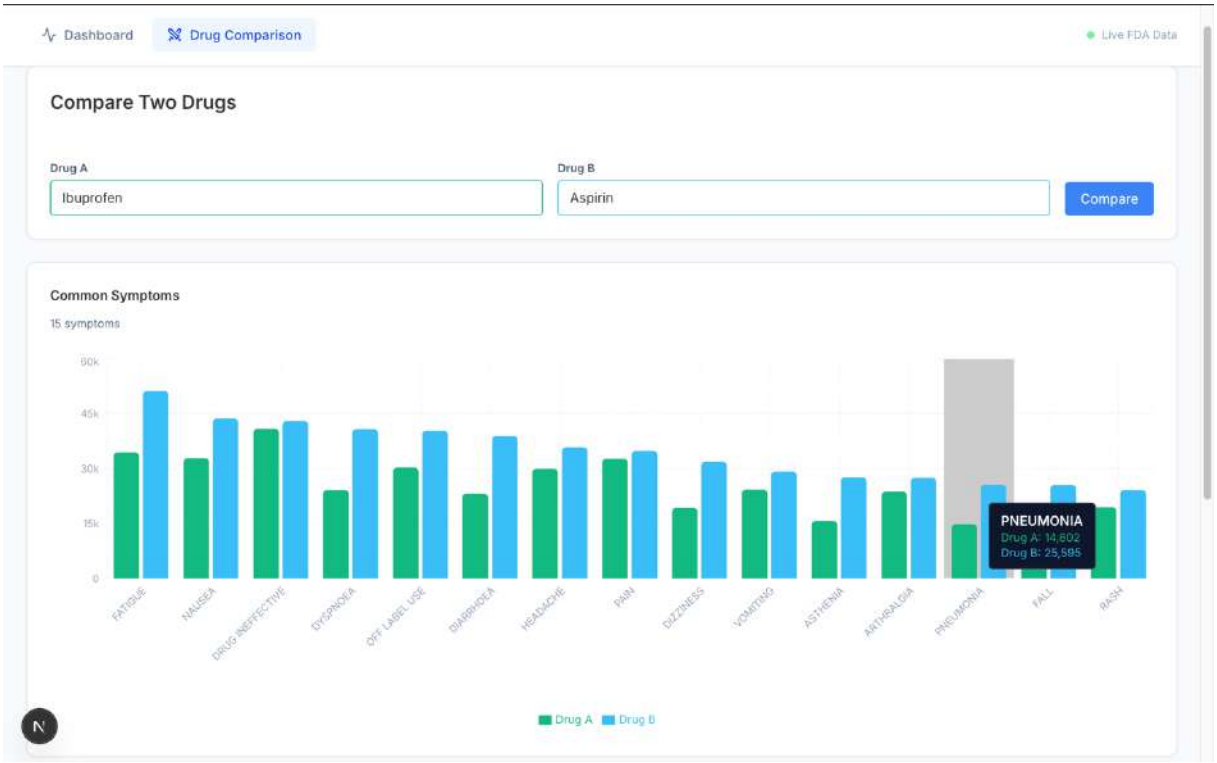


Рисунок Г.5 – Модуль порівняння препаратів: інтерактивна гістограма перетину спільних побічних реакцій (на прикладі Ibuprofen та Aspirin)

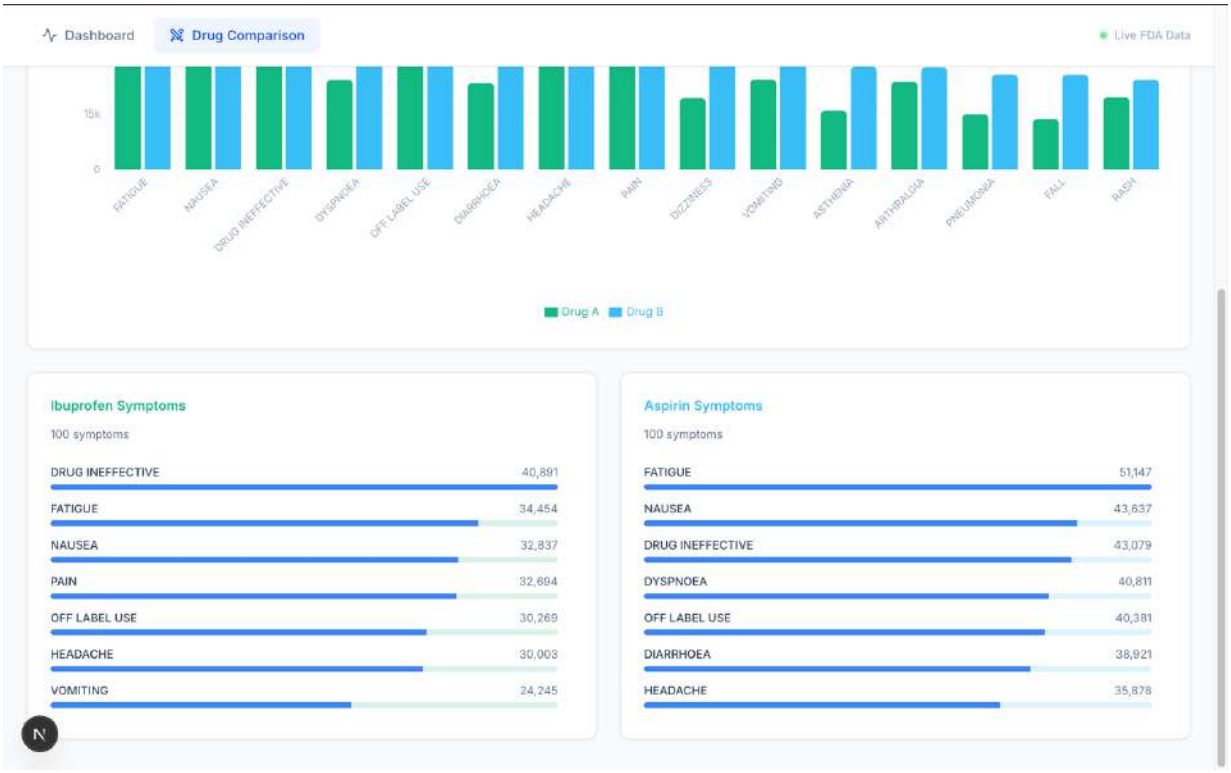


Рисунок Г.6 – Модуль порівняння препаратів: відображення деталізованих кількісних показників специфічних симптомів для кожного лікарського засобу

ДОДАТОК Д

(обов'язковий)

Фрагмент програмного коду

Лістинг Д.1. Кореневий layout застосунку (app/layout.tsx)

```
import type { Metadata } from "next";
import { Analytics } from "@vercel/analytics/next";
import "./globals.css";
import { Inter, DM_Sans } from "next/font/google";
const inter = Inter({
  subsets: ["latin"],
  variable: "--font-inter",
  weight: ["300", "400", "500", "600", "700"],
});
const dmSans = DM_Sans({
  subsets: ["latin"],
  variable: "--font-dm-sans",
  weight: ["400", "500", "600", "700"],
});
export const metadata: Metadata = {
  title: "Drug Safety Analytics",
  description:
    "Medical analytics platform for drug adverse event monitoring and safety data
visualization, powered by openFDA.",
  generator: "v0.app",
  icons: {
    icon: [
      {
        url: "/icon-light-32x32.png",
        media: "(prefers-color-scheme: light)",
      },
      {
        url: "/icon-dark-32x32.png",
        media: "(prefers-color-scheme: dark)",
      },
      {
        url: "/icon.svg",
        type: "image/svg+xml",
      },
    ],
    apple: "/apple-icon.png",
  },
};
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
```

```
    <html
      lang="en"
      className={`${inter.variable} ${dmSans.variable} bg-slate-50`}
    >
      <body className="font-sans antialiased text-slate-900">
        {children}
        {process.env.NODE_ENV === "production" && <Analytics />}
      </body>
    </html>
  );
}
```

Лістинг Д.2. Навігація між модулями (components/navigation.tsx)

```
"use client";
import Link from "next/link";
import { usePathname } from "next/navigation";
import { Activity, Swords, Search } from "lucide-react";
import { cn } from "@/lib/utils";
const navLinks = [
  { href: "/", label: "Dashboard", icon: Activity },
  { href: "/drug-comparison", label: "Drug Comparison", icon: Swords },
];
export function Navigation() {
  const pathname = usePathname();
  return (
    <header className="fixed top-0 left-0 right-0 z-50 bg-white border-b border-
slate-200 shadow-sm">
      <div className="mx-auto px-6 h-16 flex items-center justify-between">
        <nav className="flex items-center gap-1">
          {navLinks.map(({ href, label, icon: Icon }) => {
            const isActive = pathname === href;
            return (
              <Link
                key={href}
                href={href}
                className={cn(
                  "flex items-center gap-2 px-4 py-2 rounded-lg text-sm font-medium
transition-colors",
                  isActive
                    ? "bg-blue-50 text-blue-700"
                    : "text-slate-500 hover:text-slate-900 hover:bg-slate-50",
                )}
              >
                <Icon className="w-4 h-4" strokeWidth={2} />
                {label}
              </Link>
            );
          })}
        </nav>
        <div className="flex items-center gap-2 text-xs text-slate-400">
          <span className="w-2 h-2 rounded-full bg-green-400 inline-block animate-
pulse" />
```

```
          <span>Live FDA Data</span>
        </div>
      </div>
    </header>
  );
}
```

Лістинг Д.3. Сторінка дашборду - композиція модулів аналітики (app/page.tsx)

```
import { Navigation } from "@/components/navigation";
import { SafetyAlertBanner } from "@/components/safety-alert-banner";
import { AgeRisksChart } from "@/components/charts/age-risks-chart";
import { GenderStatsChart } from "@/components/charts/gender-stats-chart";
import { TopCategoriesChart } from "@/components/charts/top-categories-chart";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import {
  Card,
  CardContent,
  CardDescription,
  CardHeader,
  CardTitle,
} from "@/components/ui/card";
import { Users, Activity, BarChart2, Database } from "lucide-react";
export default function DashboardPage() {
  return (
    <div className="min-h-screen bg-slate-50">
      <Navigation />
      <main className="pt-16">
        <div className="max-w-9xl mx-auto px-6 py-8 space-y-8">
          <div className="flex items-center justify-between">
            <div>
              <h2 className="text-2xl font-bold text-slate-900 tracking-tight">
                Safety Dashboard
              </h2>
              <p className="text-slate-500 text-sm mt-1">
                Real-time adverse event monitoring powered by{" "}
                <span className="text-blue-600 font-medium">openFDA API</span>
              </p>
            </div>
            <div className="flex items-center gap-2 text-xs text-slate-400 bg-white
border border-slate-200 rounded-lg px-3 py-2">
              <Database className="w-3.5 h-3.5 text-blue-400" />
              <span>Source: api.fda.gov/drug/event</span>
            </div>
          </div>
          <SafetyAlertBanner />
          <Card className="border-slate-200 shadow-sm">
            <CardHeader className="pb-0">
              <CardTitle className="text-lg font-semibold text-slate-900">
                Adverse Event Analytics
              </CardTitle>
```

```
              <CardDescription className="text-slate-400 text-sm mt-1">
                In-depth breakdown of adverse event patterns by demographic and
                drug category
              </CardDescription>
            </CardHeader>
            <CardContent className="px-6">
              <Tabs defaultValue="age-risks">
                <TabsList  className="bg-slate-100  border  border-slate-200  p-1
rounded-xl mb-6 gap-1">
                  <TabsTrigger value="age-risks">
                    <Users className="w-4 h-4 mr-1.5" />
                    Age Risks
                  </TabsTrigger>
                  <TabsTrigger value="gender-stats">
                    <Activity className="w-4 h-4 mr-1.5" />
                    Gender Stats
                  </TabsTrigger>
                  <TabsTrigger value="top-categories">
                    <BarChart2 className="w-4 h-4 mr-1.5" />
                    Top Categories
                  </TabsTrigger>
                </TabsList>
                <TabsContent value="age-risks">
                  <AgeRisksChart
                    title="Adverse Event Frequency by Patient Age Group"
                    description="Total  adverse  event  reports  grouped  by  patient
onset age ranges"
                  />
                </TabsContent>
                <TabsContent value="gender-stats">
                  <GenderStatsChart
                    title="Symptom Frequency by Gender"
                    description="Comparative  analysis  of  common  adverse  symptoms
between male and female patients"
                  />
                </TabsContent>
                <TabsContent value="top-categories">
                  <TopCategoriesChart
                    title="Adverse Reports by Drug Category"
                    description="Ranking  of  pharmaceutical  drug  groups  by  total
adverse event report volume"
                  />
                </TabsContent>
              </Tabs>
            </CardContent>
          </Card>
        </div>
      </main>
    </div>
  );
}
```

Лістинг Д.4. Банер безпеки - запит до openFDA з фільтром за датою (components/safety-alert-banner.tsx)

```
interface TopDrugResult {
  name: string;
  count: number;
}
function buildDateParams(range: DateRange | undefined): string {
  if (!range?.from || !range?.to) return "";
  const from = format(range.from, "yyyyMMdd");
  const to = format(range.to, "yyyyMMdd");
  return `${from}+TO+${to}`;
}
const fetchTopDrug = useCallback(async () => {
  setLoading(true);
  setError(false);
  try {
    const dateParams = buildDateParams(dateRange);
    const                                    url                                    =
`https://api.fda.gov/drug/event.json?search=receivedate:[${dateParams}]&count=pati
ent.drug.openfda.substance_name.exact&limit=1`;
    const res = await fetch(url);
    if (!res.ok) throw new Error("API error");
    const data = await res.json();
    const top = data?.results?.[0];
    if (top) {
      setTopDrug({ name: top.term, count: top.count });
    } else {
      setTopDrug(null);
    }
  } catch {
    setError(true);
    setTopDrug(null);
  } finally {
    setLoading(false);
  }
}, [dateRange]);
```

Лістинг Д.5. Графік ризиків за віком - паралельні запити до openFDA (components/charts/age-risks-chart.tsx)

```
const AGE_GROUPS: { label: string; min: number; max: number }[] = [
  { label: "0–17", min: 0, max: 17 },
  { label: "18–29", min: 18, max: 29 },
  { label: "30–44", min: 30, max: 44 },
  { label: "45–59", min: 45, max: 59 },
  { label: "60–74", min: 60, max: 74 },
  { label: "75+", min: 75, max: 120 },
];
function buildDateParams(range: DateRange | undefined): string {
```

```
  if (!range?.from || !range?.to) return "";
  const from = format(range.from, "yyyyMMdd");
  const to = format(range.to, "yyyyMMdd");
  return `+AND+receivedate:[${from}+TO+${to}]`;
}
const fetchData = useCallback(async () => {
  setLoading(true);
  setEmpty(false);
  try {
    const dateParams = buildDateParams(dateRange);
    const results: AgeDataPoint[] = [];
    await Promise.all(
      AGE_GROUPS.map(async ({ label, min, max }) => {
        const search = `patient.patientonsetage:[${min}+TO+${max}]${dateParams}`;
        const                                  url                                  =
`https://api.fda.gov/drug/event.json?search=${search}&limit=1`;
        const res = await fetch(url);
        if (!res.ok) {
          results.push({ ageGroup: label, count: 0 });
          return;
        }
        const json = await res.json();
        results.push({
          ageGroup: label,
          count: json?.meta?.results?.total ?? 0,
        });
      }),
    );
    const ordered = AGE_GROUPS.map(
      (g) => results.find((r) => r.ageGroup === g.label)!,
    );
    const total = ordered.reduce((s, d) => s + d.count, 0);
    if (total === 0) {
      setEmpty(true);
      setData([]);
    } else {
      setData(ordered);
    }
  } catch {
    setEmpty(true);
    setData([]);
  } finally {
    setLoading(false);
  }
}, [dateRange]);
```

Лістинг Д.6. Графік статистики за статтю - порівняльна агрегація симптомів (components/charts/gender-stats-chart.tsx)

```
function buildDateSearch(range: DateRange | undefined): string {
  if (range?.from && range?.to) {
    const from = format(range.from, "yyyyMMdd");
```

```
    const to = format(range.to, "yyyyMMdd");
    return `receivedate:[${from}+TO+${to}]`;
  }
  const today = format(new Date(), "yyyyMMdd");
  return `receivedate:[20040101+TO+${today}]`;
}
const fetchData = useCallback(async () => {
  setLoading(true);
  setEmpty(false);
  try {
    const dateSearch = buildDateSearch(dateRange);
    const [maleRes, femaleRes] = await Promise.all([
      fetch(

`https://api.fda.gov/drug/event.json?search=(${dateSearch})+AND+patient.patientsex
:1&count=patient.reaction.reactionmeddrapt.exact&limit=10`,
      ),
      fetch(

`https://api.fda.gov/drug/event.json?search=(${dateSearch})+AND+patient.patientsex
:2&count=patient.reaction.reactionmeddrapt.exact&limit=10`,
      ),
    ]);
    if (!maleRes.ok && !femaleRes.ok) {
      setEmpty(true);
      setData([]);
      return;
    }
    const maleJson: FDACountResponse = maleRes.ok
      ? await maleRes.json()
      : { results: [] };
    const femaleJson: FDACountResponse = femaleRes.ok
      ? await femaleRes.json()
      : { results: [] };
    const maleResults = maleJson.results ?? [];
    const femaleResults = femaleJson.results ?? [];
    const symptomMap = new Map<string, { male: number; female: number }>();
    for (const item of maleResults) {
      const existing = symptomMap.get(item.term) ?? { male: 0, female: 0 };
      existing.male = item.count;
      symptomMap.set(item.term, existing);
    }
    for (const item of femaleResults) {
      const existing = symptomMap.get(item.term) ?? { male: 0, female: 0 };
      existing.female = item.count;
      symptomMap.set(item.term, existing);
    }
    const results: GenderDataPoint[] = Array.from(symptomMap.entries())
      .map(([term, counts]) => ({
        symptom: term
          .split(" ")
          .slice(0, 2)
          .map((w) => w.charAt(0).toUpperCase() + w.slice(1).toLowerCase())
          .join(" "),
        male: Math.round(counts.male / 1000),
```

```
        female: Math.round(counts.female / 1000),
      }))
      .sort((a, b) => b.male + b.female - (a.male + a.female))
      .slice(0, 8);
    if (results.length === 0) {
      setEmpty(true);
      setData([]);
    } else {
      setData(results);
    }
  } catch {
    setEmpty(true);
    setData([]);
  } finally {
    setLoading(false);
  }
}, [dateRange]);
```

Лістинг Д.7. Графік топ категорій препаратів (components/charts/top-categories-chart.tsx)

```
const fetchData = useCallback(async () => {
  setLoading(true);
  setEmpty(false);
  try {
    let searchParam = "";
    if (dateRange?.from && dateRange?.to) {
      const from = format(dateRange.from, "yyyyMMdd");
      const to = format(dateRange.to, "yyyyMMdd");
      searchParam = `search=receivedate:[${from}+TO+${to}]&`;
    }
    const                                    url                                    =
`https://api.fda.gov/drug/event.json?${searchParam}count=patient.drug.openfda.phar
m_class_epc.exact&limit=${TOP_N}`;
    const res = await fetch(url);
    if (!res.ok) {
      setEmpty(true);
      setData([]);
      return;
    }
    const json = await res.json();
    const results: { term: string; count: number }[] = json.results ?? [];
    if (results.length === 0) {
      setEmpty(true);
      setData([]);
      return;
    }
    const chartData: CategoryDataPoint[] = results.map((item) => ({
      category: item.term.replace(/\s*\[EPC\]$/, ""),
      shortLabel: createShortLabel(item.term),
      count: item.count,
    }));
```

```
    setData(chartData);
  } catch {
    setEmpty(true);
    setData([]);
  } finally {
    setLoading(false);
  }
}, [dateRange]);
```

Лістинг Д.8. Порівняння препаратів - автодоповнення назв через openFDA (app/drug-comparison/page.tsx)

```
interface SymptomData {
  term: string;
  count: number;
}
interface CommonSymptom {
  symptom: string;
  countA: number;
  countB: number;
}
interface DrugSuggestion {
  name: string;
  type: "Brand" | "Substance";
}
const fetchSuggestions = useCallback(async (query: string) => {
  if (!query.trim() || query.length < 2) {
    setSuggestions([]);
    return;
  }
  setLoading(true);
  try {
    const                                  brandUrl                                        =
`https://api.fda.gov/drug/event.json?search=patient.drug.openfda.brand_name:"${que
ry}*"+OR+patient.drug.openfda.substance_name:"${query}*"&count=patient.drug.openfd
a.brand_name.exact&limit=10`;
    const [brandRes] = await Promise.all([fetch(brandUrl)]);
    const brandData = brandRes.ok ? await brandRes.json() : { results: [] };
    const combined: DrugSuggestion[] = [
      ...(brandData.results || []).slice(0, 10).map((r: { term: string }) => ({
        name: r.term,
        type: "Brand" as const,
      })),
    ];
    setSuggestions(combined);
    setShowSuggestions(true);
  } catch {
    setSuggestions([]);
  } finally {
    setLoading(false);
  }
}, []);
```

```
const handleInputChange = (v: string) => {
  onChange(v);
  if (debounceTimer.current) clearTimeout(debounceTimer.current);
  debounceTimer.current = setTimeout(() => {
    fetchSuggestions(v);
  }, 300);
};
```

Лістинг Д.9. Порівняння препаратів - отримання симптомів і обчислення спільних реакцій (app/drug-comparison/page.tsx)

```
const fetchDrugSymptoms = useCallback(
  async (drugName: string): Promise<SymptomData[]> => {
    try {
      const search =
`patient.drug.openfda.substance_name:"${drugName}"+OR+patient.drug.openfda.brand_name:"${drugName}"+OR+patient.drug.openfda.generic_name:"${drugName}"`;
      const url =
`https://api.fda.gov/drug/event.json?search=${search}&count=patient.reaction.reactionmeddrapt.exact&limit=100`;
      const res = await fetch(url);
      if (!res.ok) throw new Error("API error");
      const data = await res.json();
      return (data?.results || [])
        .slice(0, 100)
        .map((r: { term: string; count: number }) => ({
          term: r.term,
          count: r.count,
        }));
    } catch {
      return [];
    }
  }, [],
);
const handleCompare = useCallback(async () => {
  if (!drugA.trim() || !drugB.trim()) return;
  setLoading(true);
  setError(false);
  try {
    const [dataA, dataB] = await Promise.all([
      fetchDrugSymptoms(drugA),
      fetchDrugSymptoms(drugB),
    ]);
    setSymptomsA(dataA);
    setSymptomsB(dataB);
    const termSetB = new Set(dataB.map((s) => s.term));
    const common = dataA
      .filter((s) => termSetB.has(s.term))
      .map((s) => ({
        symptom: s.term,
        countA: s.count,
        countB: dataB.find((b) => b.term === s.term)?.count || 0,
      }))
```

```
      .sort(
        (a, b) => Math.max(b.countA, b.countB) - Math.max(a.countA, a.countB),
      )
      .slice(0, 15);
    setCommonSymptoms(common);
  } catch {
    setError(true);
    setSymptomsA([]);
    setSymptomsB([]);
    setCommonSymptoms([]);
  } finally {
    setLoading(false);
  }
}, [drugA, drugB, fetchDrugSymptoms]);
```

Лістинг Д.10. Порівняння препаратів - візуалізація спільних симптомів (app/drug-comparison/page.tsx)

```
export default function DrugComparisonPage() {
  const [drugA, setDrugA] = useState("");
  const [drugB, setDrugB] = useState("");
  const [symptomsA, setSymptomsA] = useState<SymptomData[]>([]);
  const [symptomsB, setSymptomsB] = useState<SymptomData[]>([]);
  const [commonSymptoms, setCommonSymptoms] = useState<CommonSymptom[]>([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(false);
  // fetchDrugSymptoms, handleCompare - див. лістинг 9
  return (
    <div className="min-h-screen bg-slate-50">
      <Navigation />
      <main className="pt-16">
        <div className="max-w-full px-6 py-8">
          <div className="max-w-9xl mx-auto space-y-6">
            <DrugSearchHeader
              drugA={drugA}
              drugB={drugB}
              onDrugAChange={setDrugA}
              onDrugBChange={setDrugB}
              onDrugASelect={setDrugA}
              onDrugBSelect={setDrugB}
              onCompare={handleCompare}
              loading={loading}
            />
            {error && (
              <Card className="bg-red-50 border-red-100">
                <CardContent className="pt-6">
                  <p className="text-sm text-red-700">
                    Error fetching data. Please try again.
                  </p>
                </CardContent>
              </Card>
            )}
```

```
            {(symptomsA.length > 0 || symptomsB.length > 0) && (
              <>
                <ComparisonChart
                  commonSymptoms={commonSymptoms}
                  loading={loading}
                />
                <div className="flex flex-1 gap-6 h-screen max-h-96">
                  <SymptomsList
                    title={`${drugA} Symptoms`}
                    symptoms={symptomsA}
                    loading={loading}
                    color={COLORS.drugA}
                  />
                  <SymptomsList
                    title={`${drugB} Symptoms`}
                    symptoms={symptomsB}
                    loading={loading}
                    color={COLORS.drugB}
                  />
                </div>
              </>
            )}
          </div>
        </div>
      </main>
    </div>
  );
}
function ComparisonChart({
  commonSymptoms,
  loading,
}: {
  commonSymptoms: CommonSymptom[];
  loading: boolean;
}) {
  // ...
  return (
    <ResponsiveContainer width="100%" height={400}>
      <BarChart data={commonSymptoms}>
        <Bar dataKey="countA" fill={COLORS.drugA} name="Drug A" />
        <Bar dataKey="countB" fill={COLORS.drugB} name="Drug B" />
      </BarChart>
    </ResponsiveContainer>
  );
}
```

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Дмитро Клімішен

Співавтор:

Назва: Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків

Експерт: Ольга ПАВЛОВА

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:1.52%

Коефіцієнт подібності 2:0.27%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-26 20:45:04.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-27 Доцент Андрій Нічепорук

Дата експерт

ANTI-PLAGIARISM (HTTP://AP.KM.UA) V-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
ID: 272405 Назва: БКР Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків Додано в БД: 2026-05-27 Автора: Дмитро Клімішен Керівники: Ольга ПАВЛОВА Консультанти: Опоненти:	111583	774	1044 (1%)	15 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Клімішен Дмитро Віталійович

Тема: Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___65___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування, програмна реалізація та тестування спеціалізованої веб-орієнтованої інформаційної системи (аналітичної платформи) для моніторингу безпеки лікарських засобів та візуального порівняння побічних ефектів на основі відкритих урядових даних (API openFDA).

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі проаналізовано предметну область фармаконагляду та концепцію відкритих медичних даних, виявлено недоліки існуючих реєстрів та обґрунтовано вибір сучасного технологічного стека (Next.js, React, TypeScript, Tailwind CSS) для побудови швидкої клієнтської архітектури. В другому розділі виконано системне проєктування: розроблено загальну компонентну архітектуру безсерверного агрегатора, побудовано концептуальну ER-діаграму предметної області FAERS, а також створено комплекс UML-діаграм (прецедентів, послідовності та станів) для моделювання логіки роботи модуля Drug Comparison. У третьому розділі здійснено практичну програмну реалізацію вебзастосунку, інтеграцію з віддаленим RESTful API, налаштовано транзитну асинхронну обробку масивів JSON та впроваджено інтерактивну візуалізацію статистичних медичних зрізів

4. Позитивні сторони роботи: Створено функціонуючий аналітичний вебзастосунок, який вирішує задачу зіставлення профілів безпеки діючих речовин.

5. Негативні сторони роботи: Бажано було б розширити функціонал системи можливістю експорту згенерованих порівняльних графіків та звітів у формати PDF або CSV для зручнішого використання результатів аналізу в офлайн-документообігу.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре 175С

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

к.т.н. доцент Федина Ю.В.

доцент кафедри ОТВ

"5" червня 2026 р. _____ (підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Дмитро Клімішен

ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи ІСТс-23-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Веб-орієнтована інформаційна система відображення статистичних даних у вигляді графіків
Автор Дмитро КЛІМІШЕН
Освітня програма Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 Інформаційні системи та технології
Науковий керівник: Павлова Ольга Олександрівна, д.ф., доцент

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.
4) значна частина знайденого плагіату відноситься до списку використаних джерел
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 1,52%; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Єлизавета ГНАТЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ