Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

КВАЛІФІКАЦІЙНА РОБОТА
за освітнім ступенем «бакалавр»
на тему:

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ВАКАНСІЙ НА ОСНОВІ КОМПЛЕКСНОГО АНАЛІЗУ РЕЗЮМЕ КОРИСТУВАЧА

Виконав:
студент IV-го курсу
групи ІПЗ-41
спеціальності 121 «Інженерія програмного забезпечення»
Тарногурський Станіслав Андрійович
Керівник:
к.п.н., доц. Петренко С.В.

Рівне – 2026

ЗМІСТ

ВСТУП

Актуальність роботи. У сучасному світі динаміка ринку праці, особливо у сфері інформаційних технологій, вимагає принципово нових підходів до пошуку та відбору персоналу. Традиційні методи скринінгу резюме та моніторингу вакансій стають малоефективними через величезні обсяги неструктурованої інформації. Впровадження інтелектуальних систем на основі великих мовних моделей та семантичного пошуку дозволяє трансформувати процес рекрутингу, забезпечуючи точну відповідність між компетенціями кандидата та вимогами роботодавця.

Ефективність таких систем безпосередньо залежить від гнучкості архітектури та швидкості обробки даних. Використання мікросервісного підходу у поєднанні з сучасними протоколами передачі даних (gRPC) та брокерами повідомлень (RabbitMQ) дозволяє створювати масштабовані платформи, здатні аналізувати складні професійні профілі в режимі реального часу. Інтеграція штучного інтелекту для розрахунку показника відповідності (Match Score) стає критично важливим інструментом, що мінімізує вплив людського фактору та оптимізує витрати часу для кандидатів.

Мета роботи: проєктування та реалізація архітектури розподіленої інтелектуальної системи для автоматизованого аналізу резюме та підбору релевантних вакансій на основі AI-технологій.

Об’єктом дослідження є процес автоматизованого збору, обробки та семантичного аналізу професійної інформації в інтелектуальних системах рекрутингу.

Інструментом дослідження є мови програмування TypeScript (Node.js/NestJS), середовище Next.js для фронтенд-частини, векторна база даних Qdrant, реляційна БД PostgreSQL, брокер повідомлень RabbitMQ, протокол gRPC для міжсервісної взаємодії, AWS S3 сховище для резюме та платіжний шлюз Stripe.

Предметом дослідження є методи, моделі та архітектурні рішення побудови мікросервісної AI-платформи для семантичного пошуку вакансій, розрахунку Match Score за допомогою LLM та управління платіжними циклами для SaaS-моделі.

Завдання дослідження:

1. Провести аналіз сучасних технологій штучного інтелекту та методів векторного пошуку в контексті HR-технологій.
2. Спроектувати мікросервісну архітектуру системи з використанням gRPC та RabbitMQ для забезпечення відмовостійкості та масштабованості.
3. Реалізувати модуль AI-аналізу резюме для вилучення ключових навичок та оцінки відповідності вакансіям.
4. Розробити платіжну інфраструктуру на базі Stripe для підтримки різних рівнів підписок та моделі Pay-as-you-go.
5. Провести тестування системи (unit, інтеграційні та E2E тести) та оцінити економічну ефективність використання AI-ресурсів.

Апробація результатів кваліфікаційної роботи. Результати дослідження та основні технічні рішення подані для оприлюднення на І Міжнародну науково-практичну конференцію «Сучасні інформаційні технології: від теорії до практики» (м. Луцьк, 22-23 травня 2026 р.) [1].

Публікації. Подано дослідження у вигляді тез доповідей на І Міжнародну науково-практичну конференцію «Сучасні інформаційні технології: від теорії до практики» у Луцький національний технічний університет.

Структура роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто стан ринку IT-рекрутингу та роль AI у підборі персоналу. У другому проведено огляд інструментарію та технологічного стеку проєкту. У третьому розділі описано архітектуру системи, логіку взаємодії компонентів через API Gateway та інтеграцію зі Stripe. У четвертому розділі представлено результати тестування, аналіз точності AI-алгоритмів та техніко-економічне обґрунтування розробленого рішення. Список використаних джерел містить 24 джерела.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПІДХОДІВ ДО ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ВАКАНСІЙ

1.1. Аналіз сучасного стану та проблематики процесів працевлаштування на основі професійних резюме

Сучасний ринок праці характеризується високим рівнем цифровізації та значними обсягами неструктурованих даних. Трансформація сегменту HRTech, призвела до переходу від статичних баз даних вакансій до інтелектуальних систем підтримки прийняття рішень. Проте, незважаючи на технологічний прогрес, процес взаємодії між пошуковцем та роботодавцем залишається обтяженим низкою системних проблем.

На сучасному етапі розвитку ринку існує критична інформаційна прогалина. Основним інструментом представлення професійного досвіду залишається резюме (CV), яке в більшості випадків є неструктурованим текстовим документом. Згідно зі звітом World Economic Forum «The Future of Jobs Report 2025», формальні ознаки кваліфікації, такі як назви попередніх посад або наявність дипломів, поступово втрачають пріоритетність порівняно з реальним набором динамічних компетенцій кандидата, поданих в таблиці 1.1 [2].

Таблиця 1.1

Порівняльний аналіз підходів до обробки резюме та підбору вакансій

Параметр порівняння	Традиційний підхід (Keyword Matching)	Інтелектуальний підхід (AI Semantic Analysis)
Принцип аналізу	Пошук точних збігів символьних рядків.	Аналіз контексту та векторних значень слів.
Обробка синонімів	Відсутня (не розрізняє "Java" та "JVM-based development").	Повна (розуміє ієрархію та спорідненість навичок).

Гнучкість фільтрів	Бінарна (відповідає / не відповідає).	Градуальна (визначення відсотка релевантності досвіду).
Обробка помилок	Жорстке відхилення при друкарських помилках або нетиповому форматуванні.	Стійкість до шумів у тексті завдяки когнітивному парсингу.
Ефект для кандидата	Високий ризик потрапляння у «чорну діру» відгуків.	Персоналізований зворотний зв’язок та поради.

Проте, попри цей концептуальний зсув, технічна інфраструктура більшості рекрутингових платформ залишається застарілою, що створює значний розрив між потенціалом шукача та потребами роботодавця.

Ключові «больові точки» суб'єктів ринку:

- **Ефект інформаційної прірви:** відсутність зворотного зв'язку через автоматичне відсіювання системами керування кандидатами (ATS), що не здатні коректно інтерпретувати складні формати документів [3].
- **Проблема нерелевантної самопрезентації:** використання уніфікованих резюме для різних функціональних ролей, що нівелює унікальні компетенції фахівця.
- **Труднощі професійної ідентифікації:** складність об'єктивної оцінки власного рівня (Junior/Middle/Senior) та відповідності суміжним галузям.
- **Низька конверсія відгуків:** висока частка «шумових» відгуків, що не відповідають базовим вимогам вакансії.
- **Часові витрати на первинний скринінг:** за статистикою, рекрутер витрачає в середньому 6–8 секунд на первинний перегляд одного CV, що підвищує ризик суб'єктивних помилок.

Однією з ключових проблем галузі є так званий «семантичний розрив» — невідповідність між природною мовою, якою кандидат описує свій досвід у резюме, та жорсткими алгоритмічними запитами, що використовують рекрутери.

Статистичні дані за 2025–2026 роки вказують на те, що понад 75% резюме відсіюються системами керування кандидатами (ATS) ще до моменту їх перегляду людиною [3]. Основними причинами цього є не лише технічні помилки парсингу неструктурованих файлів (PDF, DOCX), а й відсутність прямих лексичних входжень ключових слів. Класичні методи пошуку, що базуються на синтаксичному порівнянні рядків (Keyword Matching), демонструють низьку ефективність: за даними аналітичного агентства Incruiter (2026), традиційні фільтри пропускають до 40% релевантних фахівців, оскільки не здатні розпізнати синонімію навичок або суміжні технологічні стеки.

Традиційні підходи до підбору вакансій, що використовуються на таких платформах, як LinkedIn, Indeed або Work.ua, базуються на алгоритмах Keyword Matching (пошук за ключовими словами). Конкуренція за людей зросла, а кандидати більше не шукають роботу так, як раніше. Щоб залишатися ефективними, рекрутери використовують методи підбору персоналу, які адаптовані до нової реальності: автоматизації, соцмереж, швидкої комунікації [4]. Дослідження показують, що цей підхід є дефіцитним з кількох причин:

- **Відсутність семантичного контексту:** класичні фільтри не враховують синонімію та ієрархію навичок. Наприклад, система може не ідентифікувати спеціаліста з «NestJS» як релевантного для вакансії «Node.js Developer», якщо пряме входження слова відсутнє.
- **Жорстка бінарна логіка:** традиційні системи працюють за принципом суворої відповідності (наприклад, досвід рівно 5 років). Це призводить до відсіювання потенційно сильних кандидатів, чий досвід становить 4.5 роки, але є більш інтенсивним або релевантним за технологічним стеком.
- **Ігнорування динаміки професійного розвитку:** лінійні фільтри не здатні оцінити «вектор розвитку» кандидата, базуючись лише на минулих посадах, і не враховують потенціал до перекваліфікації (рис. 1.1.).

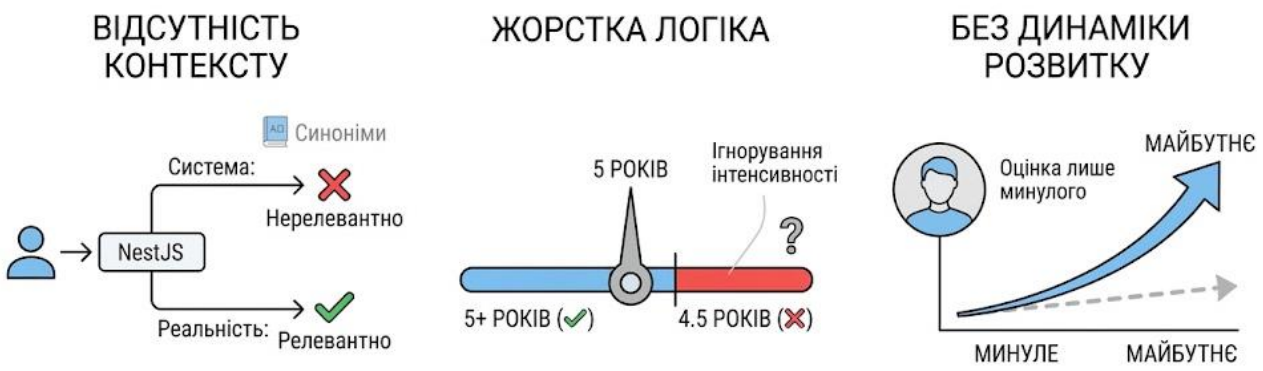

Рисунок 1.1. Недоліки пошуку за ключовими словами.

Застосування технологій генеративного штучного інтелекту та великих мовних моделей (LLM) дозволяє перейти від синтаксичного аналізу тексту до семантичного аналізу компетенцій. Інтелектуальна система, на відміну від класичних агрегаторів, функціонує як персоналізований агент, що виконує глибоку екстракцію сутностей з неструктурованих файлів, робить глибокий аналіз відповідності замість простого порівняння слів та прогностичне моделювання успішного проходження інтерв'ю на основі виявлених прогалин у знаннях [5]. При цьому великі мовні моделі не розуміють слів у звичному для нас сенсі. Замість цього вони працюють з токенами – базовими одиницями інформації для статистичного аналізу. Токеном може бути символ, ціле слово або його частина Таким чином, розробка платформи, що поєднує функції аналітичного центру для резюме та інтелектуального агрегатора вакансій, є актуальним науково-практичним завданням, спрямованим на оптимізацію витрат ресурсів на ринку праці.

Застосування технологій генеративного штучного інтелекту та великих мовних моделей (LLM) станом на 2026 рік стає єдиним ефективним рішенням для подолання цих бар’єрів. Інтеграція інтелектуальних агентів дозволяє скоротити час на первинний скринінг на 67%, одночасно підвищуючи точність прогнозу успішності кандидата з 35–40% до 78%. Практичний підхід доводить, що перехід до семантичного аналізу на основі векторних ембеддінгів дозволяє системі «розуміти» контекст професійних досягнень, а не просто констатувати наявність термінів у тексті. Таким чином, розробка платформи, яка не лише агрегує вакансії,

а й виконує функцію інтелектуального когнітивного помічника що є критично необхідною. Така система вирішує триєдине завдання: усуває технічну вразливість резюме перед ATS, забезпечує високу релевантність підбору через Skill-based matching та підвищує конкурентоспроможність кандидата шляхом автоматизованого аналізу прогалин у знаннях та надання рекомендацій до інтерв'ю.

1.2. Методи та технології аналізу резюме і вакансій

Процес автоматизованого аналізу професійної документації на сучасному етапі базується на комплексному використанні методів комп'ютерної лінгвістики та штучного інтелекту. Ключовим завданням таких систем є трансформація неструктурованого тексту резюме та описів вакансій у структуровані дані, придатні для математичного порівняння та обробки. Еволюція цих методів пройшла шлях від простого порівняння символьних рядків до складних когнітивних моделей, здатних інтерпретувати професійний контекст.

Найбільш базовим методом, який тривалий час залишався стандартом у системах керування кандидатами, є зіставлення ключових слів. Ця технологія базується на пошуку точних входжень термінів, визначених рекрутером, у тексті документа. Попри свою обчислювальну простоту, даний підхід має суттєві обмеження, оскільки не враховує синонімію, полісемію та морфологічні варіації слів. У сучасних інтелектуальних системах цей метод використовується лише як допоміжний компонент для первинної грубої фільтрації даних.

Для глибинної обробки тексту застосовуються методи обробки природної мови (Natural Language Processing, NLP) [6]. На етапі препроцесингу здійснюється токенізація, лематизація та видалення стоп-слів, що дозволяє очистити текст від мовного шуму. Центральне місце в аналізі резюме посідає технологія розпізнавання іменованих сутностей NER (рис 1.2.).

Рисунок 1.2. Глибинна обробка тексту резюме за допомогою NLP

За допомогою спеціально навчених моделей система ідентифікує в тексті специфічні категорії: назви технологій, рівні володіння мовами, заклади освіти, географічні локації та часові інтервали професійної діяльності. Це дозволяє перетворити довільний текст на чіткий профіль компетенцій кандидата.

Кроки процесу підготовки тексту резюме при обробці природної мови (NLP):

- **Токенізація:** поділ неструктурованого тексту на окремі лексичні одиниці (слова, фрази).
- **Лематизація та стемінг:** зведення слів до їхньої канонічної форми або кореня для нівелювання морфологічних розбіжностей.
- **Видалення стоп-слів:** виключення сполучників, часток та інших службових частин мови, що не несуть змістовного навантаження.
- **Розпізнавання іменованих сутностей (NER):** автоматичне виділення з тексту специфічних категорій даних, таких як назви технологій, періоди роботи та освітні ступені.

Наступний крок – з'ясувати, як усі слова в тексті пов'язані одне з одним. Мета полягає в тому, щоб побудувати дерево, яке кожному слову в реченні призначає одне батьківське слово.

Незважаючи на прогрес, у багатьох системах досі використовується метод зіставлення ключових слів, проте його технічна реалізація суттєво відрізняється від сучасних методів векторного аналізу як показано в таблиці 1.2.

Таблиця 1.2 Порівняння критеріїв синтаксичного та сематичного аналізу

Технічний критерій	**Keyword Matching (Синтаксичний аналіз)**	**Vector Embeddings (Семантичний аналіз)**
Структура даних	Плоскі списки термінів.	Багатовимірні числові вектори.
Обробка контексту	Відсутня (аналізується лише наявність слова).	Висока (враховується оточення слова та його сенс).
Математична модель	Булева логіка (True/False).	Косинусна подібність, Евклідова відстань.
Гнучкість	Низька (потребує ідентичного написання).	Висока (розпізнає концептуально близькі поняття).
Обчислювальна складність	Мінімальна (пошук у рядку).	Середня/Висока (матричні обчислення).

Якісний стрибок у точності підбору вакансій став можливим завдяки впровадженню векторних представлень слів та текстів — ембеддінгів (Embeddings). Технологія полягає у відображенні слів або цілих речень у багатовимірний векторний простір, де семантично схожі поняття розташовуються на близькій математичній відстані. На відміну від ключових слів, ембеддінги дозволяють системі «розуміти», що поняття «розробка інтерфейсів» та «Front-end development» є спорідненими. Оцінка релевантності вакансії щодо резюме в такому випадку зводиться до обчислення косинусної подібності між їхніми векторами, що забезпечує високу точність навіть за відсутності прямих текстових збігів.

Особливе місце в сучасній архітектурі посідають великі мовні моделі (LLM). Вони виконують роль інтелектуального агента, що забезпечує виконання наступних функцій:

1. **Когнітивний аудит:** аналіз резюме на предмет логічності викладу, повноти опису досягнень та відповідності стандартам галузі.

2. **Генерація рекомендацій:** автоматичне формування порад щодо виправлення помилок або розширення певних розділів документа.
3. **Симуляція співбесіди:** на основі зіставлення профілю вакансії та резюме, модель формує перелік найбільш ймовірних питань, що дозволяє кандидату підготувати обґрунтовані відповіді.
4. **Синтез супровідних листів:** створення адаптованих текстів, що підкреслюють саме ті навички кандидата, які є критичними для конкретного роботодавця.

Останнім етапом розвитку технологій у даній сфері є використання великих мовних моделей (LLM) та генеративного штучного інтелекту [5]. Ці технології дозволяють не лише аналізувати дані, а й створювати когнітивні надбудови над ними. Генеративні моделі здатні проводити комплексний аудит резюме, виявляючи логічні прогалини у викладі досвіду та надаючи стилістичні поради щодо покращення змісту. Більш того, на основі порівняльного аналізу вектора кандидата та вектора ідеального профілю вакансії, система може генерувати персоналізовані сценарії інтерв'ю, прогнозуючи потенційні запитання рекрутера та готуючи кандидата до найбільш складних аспектів технічної співбесіди.

Таким чином, сучасна архітектура систем підбору персоналу являє собою гібридну модель. Вона поєднує надійність класичних NLP-методів для структурування даних, потужність векторних представлень для забезпечення семантичного пошуку та аналітичні можливості генеративного штучного інтелекту для надання консультаційної підтримки користувачам. Такий інтегрований підхід дозволяє мінімізувати вплив суб’єктивних факторів та значно підвищити ефективність взаємодії між пошуковцем та працедавцем.

Для забезпечення високої швидкодії при роботі з великими масивами даних (вакансіями з різних джерел) застосовуються спеціалізовані векторні бази даних. Вони оптимізовані для проведення операцій пошуку за подібністю в реальному часі. У межах розробки AI-агрегатора це дозволяє миттєво зіставляти завантажене резюме з тисячами актуальних пропозицій, ранжуючи їх за ступенем семантичної відповідності. Такий підхід базується на архітектурі Retrieval-Augmented

Generation (RAG), де зовнішні дані (вакансії) інтегруються в контекст мовної моделі для формування точних та актуальних відповідей користувачеві.

RAG (Retrieval Augmented Generation) - це метод роботи з великими мовними моделями, коли користувач пише свої питання, а ви програмно до цього питання "підмішуєте" додаткову інформацію з якихось зовнішніх джерел і подаєте все на вхід мовної моделі. Іншими словами ви додаєте до контексту запиту до мовної моделі додаткову інформацію, на основі якої мовна модель може дати користувачеві більш повну та точну відповідь [7]. Основна ідея RAG-системи - розділити завдання відповіді на два етапи : пошук релевантної інформації та генерація відповіді з урахуванням знайдених даних. Таким чином, RAG-архітектура складається з двох ключових компонентів: **retriever** (пошуковий модуль) **та generator** (генеративна LLM-модель).

Моделі RAG також можуть підключатися до Інтернету за допомогою API та отримувати доступ до стрічок соціальних мереж у режимі реального часу та відгуків споживачів для кращого розуміння настроїв ринку. Тим часом доступ до термінових новин та пошукових систем може призвести до точніших відповідей, оскільки моделі включають отриману інформацію в процес генерації тексту [8].

Оскільки RAG пов'язує модель із зовнішніми джерелами знань, а не включає ці знання до навчальних даних моделі, він підтримує розрив між моделлю та цими зовнішніми знаннями. Підприємства можуть використовувати RAG для збереження даних власних джерел, одночасно надаючи моделям доступ до них — доступ, який можна скасувати будь-коли.

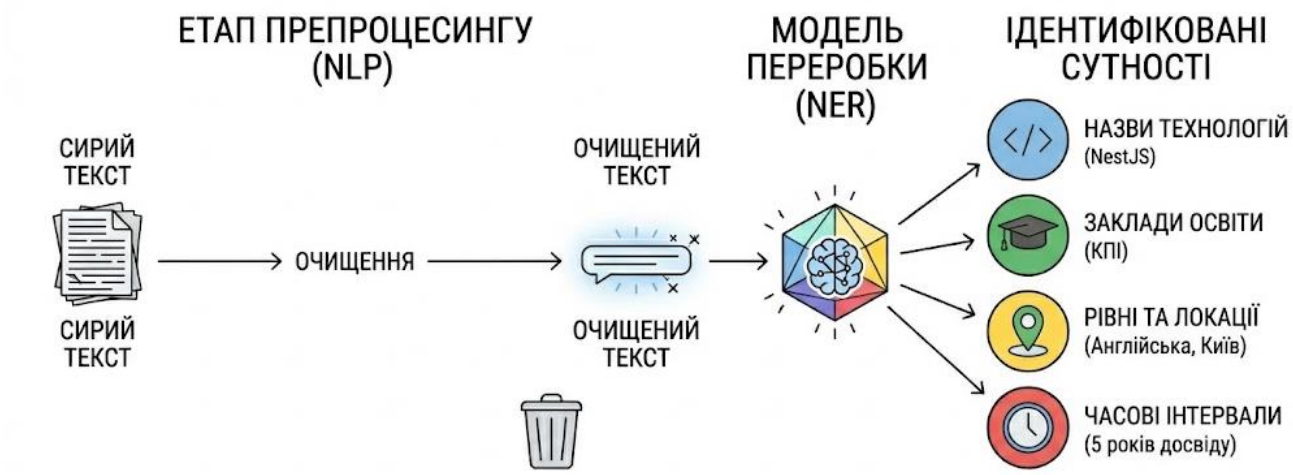

Рисунок 1.3. Мінімалістична модель RAG

Узагальнюючи технологічний стек, можна виділити три рівні обробки інформації в проекті:

- **Інфраструктурний рівень:** зберігання та швидкий пошук векторних представлень у спеціалізованих сховищах.
- **Аналітичний рівень:** використання LLM для глибокого семантичного розбору та виявлення прихованих зв’язків.
- **Інтерфейсний рівень:** генерація персоналізованого контенту (порад, питань до співбесіди) у формі природного діалогу.

1.3. Огляд існуючих систем та конкурентний аналіз

Сучасний ринок автоматизованих систем працевлаштування представлений широким спектром інструментів, які можна класифікувати за їхньою функціональною спрямованістю. Попри активне впровадження штучного інтелекту, більшість існуючих рішень залишаються вузькоспеціалізованими, що змушує користувача використовувати декілька розрізнених сервісів для досягнення єдиної мети — успішного працевлаштування.

На сьогодні основними гравцями ринку є платформи, що фокусуються на окремих етапах життєвого циклу пошуку роботи. Нижче наведено порівняльний аналіз найбільш розповсюджених систем у таблиці 1.3.

Таблиця 1.3 Порівняльна характеристика існуючих сервісів для пошуку роботи

Назва системи	**Тип сервісу**	**Основні переваги**	**Основні недоліки**
Rezi	Оптимізатори резюме	Глибокий аналіз ATS-сумісності, виявлення ключових слів.	Обмеженість лише текстовим аналізом; відсутність пошуку вакансій.
LinkedIn або Indeed	Глобальні агрегатори	Величезна база вакансій, соціальне підтвердження.	Низька релевантність «розумних» рекомендацій; високий рівень спаму.
Teal або Huntr	Трекери заявок	Ефективне керування воронкою відгуків, зручний інтерфейс.	Слабкі аналітичні можливості щодо покращення змісту резюме.
Final Round AI	Інтерв'ю-коучі	Глибока підготовка до співбесід у реальному часі.	Повна ізоляція від етапу пошуку вакансій та аналізу резюме.

Глобальні платформи-агрегатори володіють найбільшими масивами даних, проте їхні алгоритми ранжування часто базуються на комерційних пріоритетах (просування платних оголошень) або застарілих методах пошуку за ключовими словами. Це призводить до низької релевантності видачі, де кандидат отримує пропозиції, що відповідають лише назві посади, але не враховують специфіку його технічного стеку або професійних амбіцій. Крім того, такі системи не надають інструментів для глибокого аналізу файлу резюме самого користувача, залишаючи процес адаптації документа повністю на стороні кандидата.

Інша категорія − сервіси оптимізації резюме фокусуються виключно на технічній сумісності документів із системами ATS. Їхнім головним недоліком є «статичність» аналізу: вони вказують на помилки у форматуванні або відсутність

ключових слів, але не пропонують прямого виходу на ринок праці. Користувач отримує «ідеальний» документ, проте змушений самостійно шукати вакансії на сторонніх ресурсах, що знову призводить до втрати часу та контексту.

При постановці задачі на основі проведеного аналізу предметної області та існуючих рішень, можна констатувати, що на ринку відсутній доступний інтегрований інструмент, який би поєднував глибоку аналітику файлів резюме з релевантним пошуком та персоналізованою підготовкою до інтерв'ю в єдиному інтерфейсі.

Аналіз продемонстрував, що головною проблемою існуючих рішень є функціональна фрагментація. Користувач змушений вручну переносити дані з одного сервісу в інший, що призводить до втрати контексту та зниження ефективності.

На відміну від конкурентів, система забезпечує безперервний цикл підтримки на рисунку 1.4:

- **Інтегрований фідбек-луп:** зміни, внесені в резюме на основі AI-порад, миттєво впливають на алгоритм ранжування вакансій.
- **Семантичний агрегатор:** на відміну від LinkedIn, система шукає вакансії не за назвою посади, а за вектором компетенцій.
- **Контекстна підготовка:** перелік питань до співбесіди генерується не абстрактно, а на основі конкретного зіставлення «Резюме — Вакансія», що дозволяє підготувати відповіді на найбільш критичні запитання щодо досвіду кандидата.

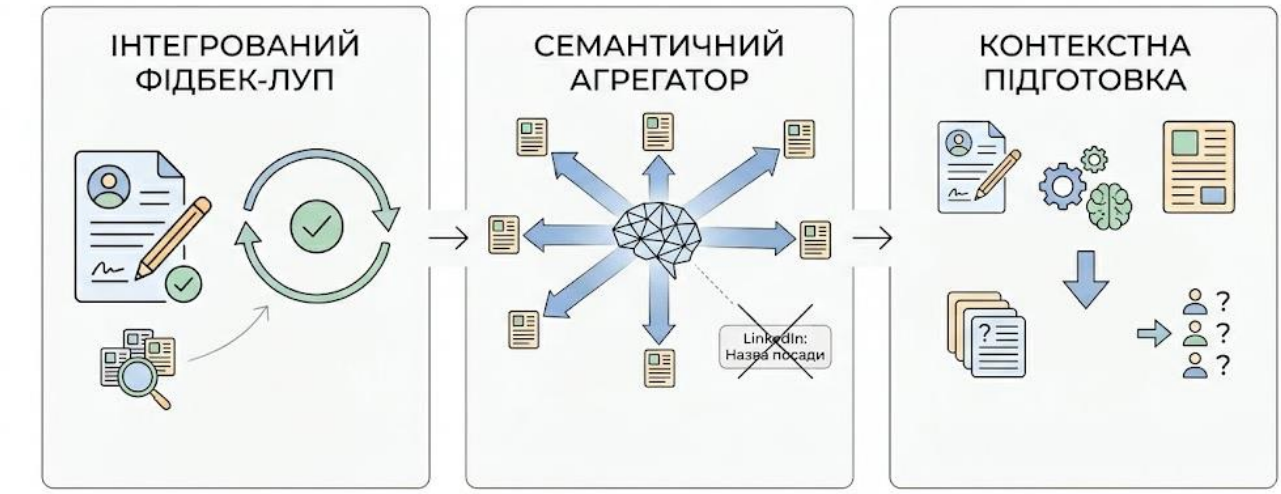

Рисунок 1.4. Система забезпечує безперервний цикл підтримки

Метою даної роботи є розробка інтелектуальної платформи для автоматизації процесів працевлаштування, яка повинна вирішувати пов’язані між собою завдання з глибоким мисленням, а саме:

1. Розробка модуля когнітивного парсингу неструктурованих файлів резюме (PDF/DOCX) з використанням LLM для екстракції професійних сутностей.
2. Створення алгоритму семантичного матчингу на основі векторних ембеддінгів для пошуку вакансій за реальною відповідністю компетенцій.
3. Реалізація системи генерації персоналізованих порад щодо оптимізації CV та сценаріїв інтерв’ю.
4. Побудова зручного веб-інтерфейсу для агрегації результатів аналізу та взаємодії користувача з AI-агентом.

Виконання поставлених завдань дозволить створити сервіс, що значно скорочує часові витрати на працевлаштування та підвищує якість взаємодії між кандидатом і ринком праці. Запропонований у даній роботі сервіс розробляється як універсальний інтелектуальний хаб (Career Co-pilot). Його ключова наукова та практична цінність полягає у подоланні вузькоспеціалізованого підходу шляхом створення наскрізного процесу: від когнітивної інтерпретації досвіду кандидата до фінальної імітації співбесіди на основі реальних вимог ринку. Система не просто надає інструменти, а формує динамічну модель професійної особистості, яка транслюється на всі етапи працевлаштування.

Для підвищення конкурентоспроможності та глибини аналізу сервісу, у межах даної роботи передбачено впровадження додаткового інноваційного функціоналу:

- **Інтеграція зовнішніх професійних профілів:** можливість аналізу не лише файлів резюме, а й автоматичного збору даних з GitHub або LinkedIn для верифікації заявлених навичок через реальний код або соціальне підтвердження. Це базовий елемент, який дозволяє чітко ідентифікувати, для якої саме ролі створюється документ. Назва посади повинна бути зрозумілою як для рекрутера, так і для потенційного кандидата [9].
- **Модуль аналізу ринкових трендів у реальному часі:** додавання функції прогнозування затребуваності певних навичок, що дозволяє системі надавати поради не лише щодо виправлення резюме, а й щодо необхідності освоєння нових технологій (Upskilling). Основне завдання цих технологій у бізнесі – навчитися розуміти поведінку клієнта в умовах, що постійно змінюються [10].
- **Мультимовна адаптація контенту:** автоматична трансформація резюме та підготовка до інтерв'ю для міжнародних ринків з урахуванням культурних та професійних особливостей конкретних регіонів. Рішення зробити профіль користувача різними мовами однозначно посприяє розвитку вашого бізнесу, виходу на нові джерела та новий ринок.

На основі виявлених дефіцитів ринку було сформульовано мету роботи — розробку інтегрованого сервісу, що забезпечує когнітивний парсинг неструктурованих файлів, семантичний пошук вакансій та генеративну підтримку на етапі підготовки до співбесіди.

Таким чином, теоретичний аналіз підтверджує актуальність та науково-практичну значущість обраного напрямку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ВАКАНСІЙ

2.1. Вимоги до системи та сценарії використання

Процес проєктування інтелектуальної платформи розпочинається з деталізації вимог, що висуваються до її функціональності, надійності та взаємодії з користувачем. Специфікації розробки базується на поєднанні класичних принципів програмної інженерії та специфічних підходів до побудови систем штучного інтелекту. Оскільки система базується на опрацюванні конфіденційних даних резюме та взаємодії з великими мовними моделями, особлива увага приділяється точності обробки запитів та швидкодії інтерфейсу.

Функціональні вимоги визначають конкретні сервіси та можливості, які система повинна надавати кінцевому користувачу:

- **Модуль управління даними**: система має підтримувати завантаження файлів у форматах PDF та DOCX, забезпечуючи коректне вилучення тексту без втрати структури документа.
- **Інтелектуальний аналіз:** автоматичне розпізнавання професійних сутностей (стек технологій, досвід, освіта) та їх трансформація у структурований JSON-формат.
- **Семантичний пошук**: реалізація механізму зіставлення профілю користувача з базою вакансій на основі косинусної подібності векторів, а не простого входження ключових слів.
- **Звичайний розширений пошук**: механізм скрепінгу вакансій з основних джерел на основі наданої інформації досить добре знаходить релевантні вакансії за проаналізованим резюме, що є безкоштовною версією пошуку для користувачів Free-tier рівня.

- **Генеративний модуль порад**: формування персоналізованого звіту з рекомендаціями щодо покращення змісту резюме (виявлення слабких місць, стилістичні правки).
- **Модуль підготовки до інтерв'ю**: генерація списку ймовірних питань на основі специфіки вакансії та досвіду кандидата.

Проте, нефункціональні вимоги визначають обмеження та стандарти якості роботи платформи:

- **Продуктивність:** час первинної обробки резюме не повинен перевищувати 10–15 секунд (враховуючи затримки API мовних моделей).
- **Масштабованість:** архітектура системи повинна дозволяти горизонтальне масштабування при збільшенні кількості одночасних сесій та обсягу бази вакансій.
- **Безпека та приватність:** забезпечення шифрування персональних даних та автоматичне видалення тимчасових файлів після завершення аналізу.
- **Юзабіліті:** мінімалістичний інтерфейс, що не потребує спеціального навчання користувача для початку роботи.

Для формалізації взаємодії користувача з платформою розроблено сценарії використання, які охоплюють основні бізнес-процеси сервісу. Головним актором системи є Кандидат, а допоміжним — AI-інструментарій. Для деталізації логіки взаємодії розроблено набір сценаріїв у таблиці 2.1, що описують шлях користувача від моменту завантаження даних до отримання фінального результату.

Важливою частиною сценаріїв є обробка виняткових ситуацій. Наприклад, якщо файл резюме містить лише зображення без текстового шару (скан-копія), система повинна ініціювати модуль OCR або запропонувати користувачеві завантажити текстову версію. Цей паттерн проєктування гарантує стійкість системи до неякісних вхідних даних та забезпечує позитивний користувацький досвід.

В цілому оптичне розпізнавання символів дає досить широкий спектр переваг, які можуть використовувати компанії всіх галузей і розмірів. Окрім

підвищення ефективності пошуку даних, OCR лежить також в основі інших процесів, наприклад машинного навчання [12].

Таблиця 2.1 Опис основних сценаріїв використання системи

Назва сценарію	Опис та послідовність дій	Очікуваний результат
Аналіз та оптимізація CV	Користувач завантажує файл резюме. Система проводить парсинг, порівнює дані зі стандартами галузі та видає перелік порад.	Структурований звіт із вказаними зонами росту та помилками.
Семантичний підбір вакансій	Система використовує векторизовані дані резюме для пошуку найбільш релевантних оголошень у підключеній базі.	Список вакансій, ранжованих за відсотком відповідності (Match Score).
Симуляція технічного інтерв'ю	Користувач обирає конкретну вакансію. Система генерує топ-10 питань, які базуються на «прогалинах» між вимогами вакансії та досвідом у CV.	Перелік питань та орієнтовні вектори правильних відповідей.
Керування профілем компетенцій	Користувач переглядає вилучені системою навички, може редагувати їх або додавати посилання на GitHub/LinkedIn.	Актуалізований цифровий профіль спеціаліста.

Визначені вимоги та сценарії використання стають фундаментом для наступного етапу проєктування – вибору архітектурних паттернів та розробки структурної схеми бази даних, яка повинна підтримувати як реляційні зв'язки, так і векторні операції. Такий підхід мінімізує помилки «холодного старту» та забезпечує високу релевантність відповідей AI-агента.

Аналіз сценаріїв використання дозволяє змоделювати поведінку системи в умовах реальної експлуатації. Нижче представлено розширений опис ключових прецедентів.

2.2. Архітектура системи та взаємодія компонентів

Проєктування архітектури системи базується на принципах мікросервісного підходу, що дає змогу забезпечити високу масштабованість, відмовостійкість та ізоляцію контекстів обробки даних. Вибір децентралізованої структури обумовлений різнорідністю завдань: від швидкої синхронної авторизації до тривалих асинхронних процесів генеративного аналізу резюме. Центральним елементом взаємодії із зовнішнім світом є API Gateway, який виконує роль єдиної точки входу. Він відповідає за маршрутизацію запитів, агрегацію відповідей від мікросервісів, лімітування запитів (Rate Limiting) та первинну валідацію токенів доступу.

Відповідно до функціональних вимог, систему розділено на п'ять спеціалізованих мікросервісів, кожен з яких оперує власною базою даних для забезпечення принципу Database per Service.

- Модуль авторизації, протокол комунікацій gRPC
  - *База даних:* Реляційна БД (наприклад, PostgreSQL) для зберігання облікових записів користувачів, хешованих паролів та профілів [22].
  - *Роль:* Управління реєстрацією, входом та генерацією JWT-токенів.
- Модуль оброки за допомогою LLM, протокол комунікацій Rabbit MQ
  - *База даних:* In-memory DB (Redis) для тимчасового зберігання статусів обробки черги [23].
  - *Роль:* Безпосередня взаємодія з мовними моделями, виконання операцій парсингу та формування векторних ембеддінгів.
- Допоміжний модуль обробки згенерованих даних, протокол комунікацій gRPC

- *База даних:* Спеціалізована база даних (PostgreSQL).
- *База даних:* Спеціалізована база даних із підтримкою векторного пошуку (Qdrant) [24].
- *Роль:* Зберігання конфігурацій LLM моделей, кешування проаналізованих резюме у структурованому форматі та ведення детальних логів аналітики.

- Модуль сповіщень, протокол комунікацій Rabbit MQ
  - *База даних:* БД для зберігання шаблонів листів та історії відправлень.
  - *Роль:* Відправка Email-підтверджень, сповіщень про завершення аналізу резюме та маркетингових розсилок.
- Модуль збереження файлів, протокол комунікацій Rabbit MQ
  - *База даних:* БД для зберігання метаданих файлів (MIME-типи, посилання на власника), у той час як самі файли зберігаються в об'єктному сховищі (S3).
  - *Роль:* Управління життєвим циклом файлів та забезпечення безпечного доступу до них.
- Модуль платежнів, протокол комунікацій gRPC
  - *База даних:* БД для зберігання підписок, транзакцій.
  - *Роль:* Управління життєвим циклом транзакцій та забезпечення безпечного доступу до усіх транзакцій та системи підписок.

Взаємодія між компонентами системи реалізована за допомогою двох взаємодоповнюючих паттернів gRPC та Rabbit MQ:

- **gRPC -** високопродуктивний каркас для взаємодії між сервісами, що дає можливість згенерувати код клієнта й сервера на основі файлів з розширенням **.proto**. Застосовується в критичних вузлах, де потрібна миттєва відповідь. Використання протоколу HTTP/2 та бінарної серіалізації Protocol Buffers значно скорочує обсяг трафіку між сервісами порівняно з традиційним REST JSON [13], див рисунок 2.1.

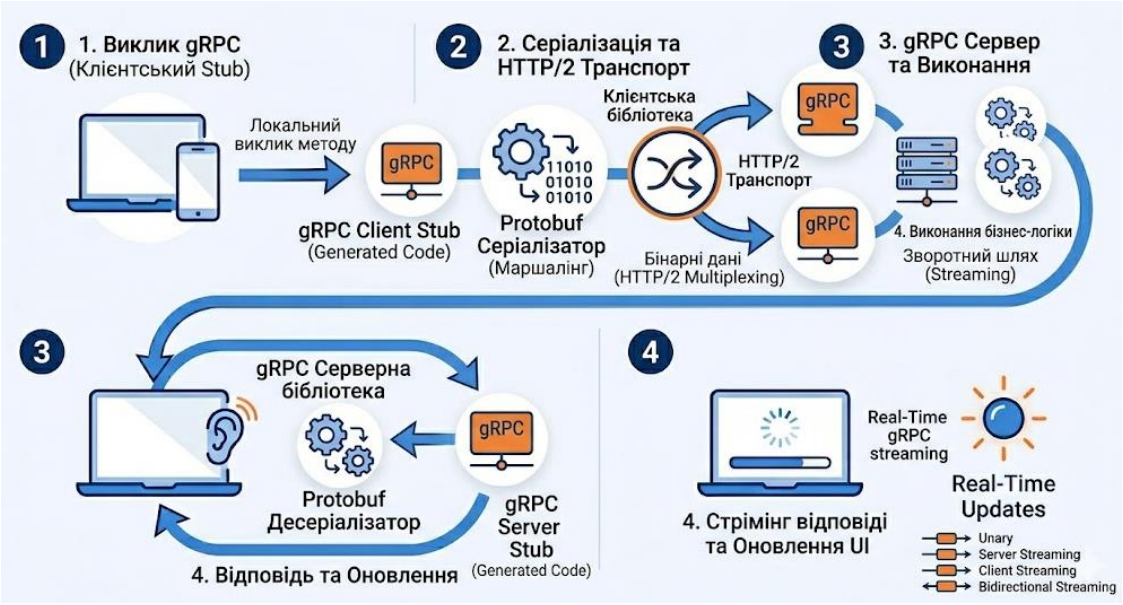

Рисунок 2.1. Комунікація за допомогою gRPC

- **RabbitMQ -** брокер повідомлень для складних сценаріїв. Його завдання спростити маршрутизацію між модулями програми та зробити її більш ефективною. Реалізує подієво-орієнтовану модель. Наприклад, після завантаження резюме через кінцеву точку, у чергу RabbitMQ відправляється повідомлення, яке згодом підхоплює мікросервіс для аналізу. Це гарантує, що навіть при пікових навантаженнях система не втратить дані, а черга поступово буде опрацьована вільними воркерами [14].

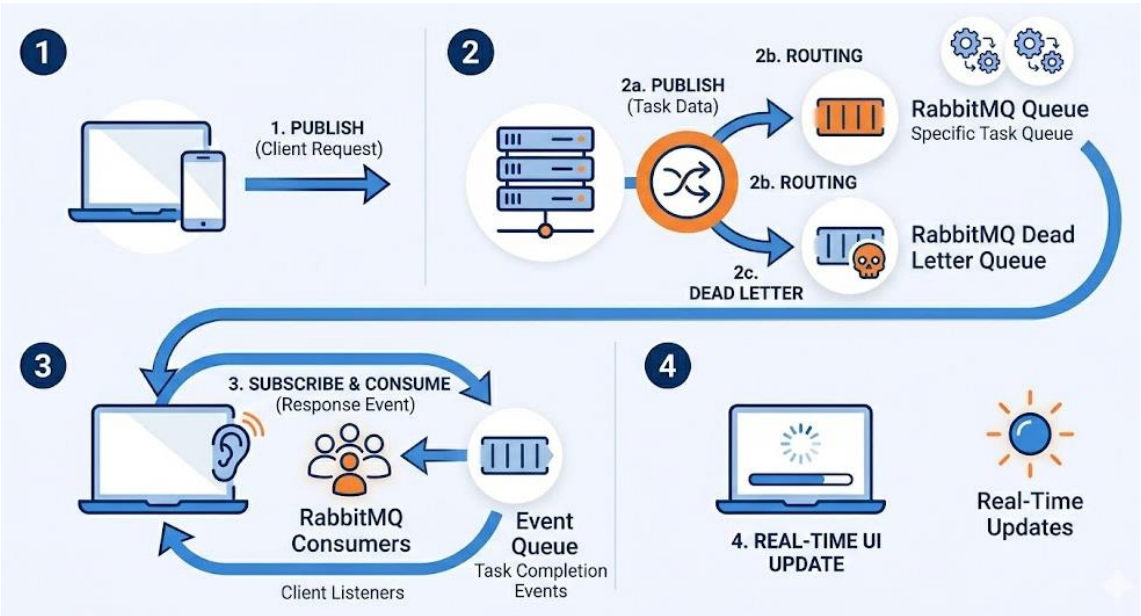

Рисунок 2.2. Комунікація за допомогою RabbitMQ

Вибір між gRPC та RabbitMQ для конкретних вузлів системи базується на аналізі характеру даних та вимог до часу відповіді. В межах проєкту архітектурне рішення спрямоване на усунення недоліків обох підходів шляхом їхнього взаємодоповнення. Детальне порівняння характеристик обраних технологій наведено у таблиці 2.2.

Таблиця 2.2 Порівняння gRPC та RabbitMQ в архітектурі системи

Критерій порівняння Моделей	**gRPC (Синхронна)**	**RabbitMQ (Асинхронна)**
Парадигма	Запит-відповідь	Черги, підписник-слухач
Тип зв’язку	Тісний зв’язок	Слабкий зв’язок
Затримка (Latency)	Мінімальна (HTTP/2, Binary)	Вища (залежить від стану черги)
Гарантія доставки	Залежить від доступності сервісу	Максимальна, залишається у черзі доки не виконається
Формат даних	Protocol Buffers (бінарний)	Довільний (зазвичай JSON)
Основна перевага	Продуктивність та типізація	Відмовостійкість та балансування навантаження

Використання gRPC є критичним для сервісів авторизації та доступу до ядра AI-даних, оскільки будь-яка затримка на цих етапах безпосередньо впливає на швидкість відображення сторінок користувачеві. Бінарна серіалізація через Protobuf дає змогу зменшити обсяг переданих даних на 30–50% порівняно з JSON, що критично при високій інтенсивності міжсервісних запитів.

З іншого боку, RabbitMQ виступає стабілізатором системи при виконанні «важких» операцій. Наприклад, процес аналізу резюме мовною моделлю може тривати від декількох секунд до хвилини. Використання асинхронного повідомлення дає змогу Gateway одразу повернути користувачеві статус

«Обробляється», не блокуючи з’єднання. Брокер повідомлень гарантує, що запит не буде втрачено у разі короткочасного збою мікросервісу обробки, оскільки повідомлення залишиться в черзі до моменту успішного підтвердження.

Оскільки навантаження на систему є нерівномірним (наприклад, модуль генерування AI споживає значно більше CPU та пам'яті під час роботи LLM, ніж модуль авторизації), мікросервісна архітектура на основі гібридного типу комунікацій між сервісами дозволяє застосовувати горизонтальне масштабування як для мікросервісу так і для воркерів. При збільшенні черги в RabbitMQ система може автоматично запускати додаткові екземпляри-воркери саме того сервісу, який потребує ресурсів у цей момент.

2.3. Моделювання даних та логіки підбору вакансій

Ефективність функціонування інтелектуальної платформи персоналізованого підбору вакансій критично залежить від якості спроєктованої моделі даних. На етапі проєктування логічної структури необхідно вирішити завдання трансформації неструктурованих текстових масивів у чітко типізовані цифрові профілі компетенцій, а також забезпечити можливість виконання математичних операцій над векторними представленнями цих даних. Також варто зауважити, що файл резюме може в себе включати будь-який стиль оформлення даних та різну кількість специфікацій, зібраних на основі досвіду у будь-якій сфері. Такий широкий діапазон даних змушує генерувати чітку але уніфіковану структуру даних, яка може бути згенерована без серйозних промахів, які можут вплинути на пошук вакансій.

Логічна структура бази даних побудована за сутнісно-зв’язною моделлю (Entity-Relationship), яка враховує специфіку мікросервісної архітектури та потребу в гібридному зберіганні інформації. Оскільки система оперує як традиційними реляційними даними (облікові записи, логи), так і складними векторами (ембеддінгами), де було інтегровано векторну базу даних для забезпечення високої швидкості семантичного пошуку.

Логічна структура системи охоплює декілька критичних доменів, кожен з яких представлений власним набором сутностей та зв’язків. Першим фундаментальним доменом є Сервіс авторизації, який відповідає за ідентифікацію та розмежування прав доступу користувачів. Основними сутностями цього сервісу є:

- Користувач (User): базова сутність, що зберігає унікальний ідентифікатор, електронну пошту та зашифрований пароль. Вона є точкою входу для всіх інших сервісів системи через свій ID.
- Роль (Role): визначає рівень доступу користувача (наприклад, «User», «Admin»), що дозволяє гнучко керувати правами в межах всієї платформи.
- Провайдер авторизації: сутність, що дозволяє реалізувати багатофакторну або сторонню авторизацію (наприклад, Google, GitHub). Вона пов’язана з користувачем через логічний ідентифікатор, що забезпечує можливість використання декількох методів входу для одного облікового запису.

Сервіс штучного інтелекту. Цей сервіс не лише виконує аналіз даних, а й керує конфігураціями мовних моделей та відстежує витрати ресурсів. Структура даних тут включає:

- LM модель: описує конкретну велику мовну модель (наприклад, GPT-4o, Claude 3.5), її параметри, версію та провайдера.
- Пресет: набір налаштувань для конкретної моделі, включаючи системні промпти, температуру генерації та ліміти токенів. Пресет має прямий зв’язок із сутністю LM моделі.
- Користувацькі пресети: дають користувачу можливіть використовуваті різні AI моделей для більш глибокого або абсолютно не передбачуваного результату аналізу, який може виявити щось унікальне.
- Профілі проаналізованих вакансій: зберігає результат розбору вакансій. Тут міститься структуровані дані у форматі JSON, вилучені з тексту оголошення.

- Репорт аналізу: критично важлива сутність для моніторингу працездатності та економіки системи. Вона фіксує кожен запит на аналіз, кількість використаних токенів, статус виконання та загальну вартість операції.

Сервіс платежів, який забезпечує монетизацію платформи та керує лімітами користувачів. Сутності цього домену включають:

- План: опис тарифних планів (Free, Pro, Ultimate), що включає ціну, доступні ліміти на кількість аналізів та додаткові функції.
- Користувацька підписка: сутність, що зв'язує користувача з певним планом, фіксуючи дати початку та закінчення дії послуги.
- Транзакція: історія фінансових операцій, що зберігає статус платежу, суму, валюту та посилання на зовнішній платіжний шлюз.

Ключовою функціональною перевагою розробленої системи є багатоетапний алгоритм персоналізованого підбору вакансій, який поєднує методи цільового збору даних (web scraping) та глибокого семантичного аналізу. На відміну від стандартних систем, що обмежуються пошуком у заздалегідь підготовленій базі, дана реалізація використовує динамічний підхід, де процес пошуку ініціюється безпосередньо результатами AI-аналізу конкретного резюме.

Після вилучення ключових пошукових даних з проаналізованого резюме, відбувається динамічний збір даних. Реалізація цього модуля базується на використанні інструментів автоматизації браузера, що дозволяє в реальному часі отримувати актуальні оголошення з провідних джоб-бордів. Зібрані сирі дані вакансій проходять етап очищення від HTML-розмітки та нормалізації тексту для подальшої обробки.

Для кожної вилученої вакансії створюється векторне представлення. На цьому етапі текст вакансії перетворюється у математичний об'єкт - ембеддінг. Він дозволяє системі оперувати значеннями та контекстом професійних вимог, а не просто послідовністю символів.

Фінальне зіставлення відбувається за рахунок порівняння вектора резюме користувача з масивом векторів знайдених вакансій.

Окрім косинусної подібності векторів, алгоритм враховує додаткові параметри, вилучені під час аналізу резюме:

1. Рівень очікуваної заробітної плати.
2. Відповідність рівня кваліфікації.
3. Тип зайнятості.
4. Галузева специфіка попереднього досвіду.

Такий підхід до моделювання даних дозволяє вирішити проблему вузькоспеціалізованості традиційних систем. Завдяки сутності репорту аналізу, система може динамічно підбирати найбільш ефективну модель для кожного конкретного завдання, балансуючи між якістю аналізу та вартістю токенів. Наприклад, для первинного скринінгу великої бази вакансій може використовуватися швидша та дешевша модель, тоді як для фінальної генерації питань до інтерв’ю – найбільш потужна LLM.

Особливістю даної моделі є також механізм роботи з користувацькими пресетами. Це дозволяє системі адаптуватися під специфічні запити користувача, наприклад, фокусуватися виключно на вакансіях для віддаленої роботи або пріоритезувати стартапи. Логічний зв'язок за userId дає змогу сервісу AI миттєво підтягувати ці налаштування без необхідності дублювання персональних даних користувача в кожній базі.

Узагальнюючи логіку проєктування на рисунку 2.3, можна виділити наступні етапи обробки інформації в межах описаної моделі сутностей:

1. Етап ідентифікації: модуль авторизації підтверджує особу користувача та передає його ідентифікатор далі по ланцюжку.
2. Етап підписки: модуль платежів перевіряє активність підписки та наявність доступних лімітів за допомогою системи підписок.
3. Етап аналізу: модуль AI використовує обраний пресет для аналізу резюме.
4. Етап пошуку: На основі проаналізованих даних за допомогою модулю пошуку робіт, був виконаний пошук по основним ключам та зіставлення векторів знайдених вакансій та ключових навичок.

5. Етап видачі: Користувач отримує відранжований список вакансій разом із персоналізованими. Проте, якщо відповідних вакансій не знайдено, користувачу натомість надається список найбільш популярних вакансій які можуть бути не пов’язані з його ключовими навичками.

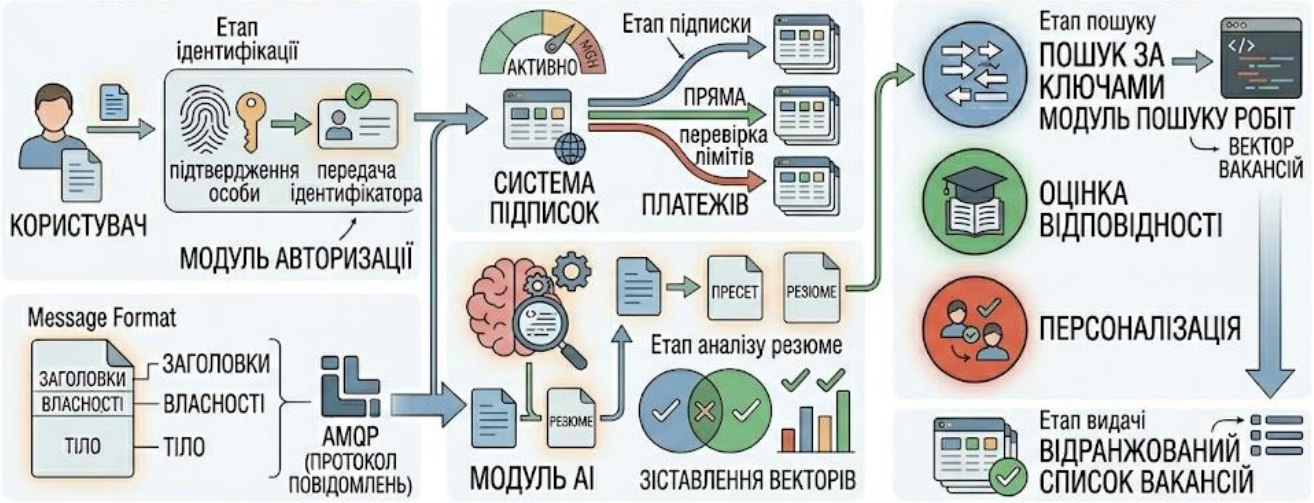

Рисунок 2.3. Процес аналізу резюме та ранжування вакансій

Це робить платформу стійкою до технологічних змін у сфері штучного інтелекту та дозволяє безперервно покращувати якість підбору вакансій для кінцевого користувача. Таким чином, спроєктована модель сутностей та логіка їхньої взаємодії повністю відповідають вимогам до сучасних високонавантажених AI-систем.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ОСНОВНИХ МОДУЛІВ ПЛАТФОРМИ

3.1. Моделювання даних та логіки підбору вакансій

Програмна реалізація платформи базується на принципах модульності та чіткої декомпозиції функціоналу, що дає змогу забезпечити високу гнучкість при розгортанні та подальшому супроводі системи. Основний акцент при розробці було зроблено на створенні стійкої мікросервісної архітектури, де кожен компонент є автономною одиницею з власною логікою та ізольованим середовищем збереження даних.

Реалізація клієнтської частини виконана з використанням фреймворку Next.js, що забезпечує оптимальне поєднання серверного та клієнтського рендерингу. Ключовою особливістю реалізації інтерфейсу є його повна ізоляція від внутрішньої мікросервісної структури. Взаємодія з серверною частиною здійснюється виключно через єдину точку входу - API Gateway, що реалізує REST-інтерфейс.

Для забезпечення ефективного мережевого обміну та управління станом даних на фронтенді застосовано наступний стек:

- **Axios:** використовується як основний HTTP-клієнт для формування запитів до Gateway. Завдяки системі інтерцепторів, реалізовано автоматичне додавання токенів авторизації та централізовану обробку помилок сервера [14].
- **TanStack Query:** виконує роль потужного менеджера асинхронного стану. Це дозволило реалізувати механізми кешування відповідей, автоматичного оновлення даних у фоновому режимі рефетчинг та оптимістичного оновлення інтерфейсу, що критично важливо для забезпечення плавного користувацького досвіду при тривалих операціях аналізу [15].

Зв'язок між клієнтом та серверною частиною організований таким чином, що фронтенд не оперує складними протоколами взаємодії мікросервісів, усі операції проходять через єдиний вузок.

Серверна ієрархія побудована на базі мікросервісного підходу з використанням фреймворку NestJS 11 та Nx Monorepo. Основна перевага такої реалізації полягає у фізичному та логічному розділенні бізнес-доменів. Центральний вузол, API Gateway, трансформує вхідні HTTP-запити у внутрішні протоколи взаємодії.

Для забезпечення максимальної незалежності сервісів застосовано патерн «окремої бази даних для кожного мікросервісу». Це дозволяє вибирати найбільш релевантний тип сховища під конкретну задачу (наприклад, реляційні БД для фінансових транзакцій у сервісі платежів та векторні БД для AI-ядра), виконувати незалежне масштабування та оновлення схем даних без ризику порушення цілісності всієї системи та усунути жорстку залежність на рівні SQL-запитів, замінивши її логічною синхронізацією за ідентифікаторами.

Для запобігання дублюванню коду та забезпечення ідентичності налаштувань середовища, до архітектури впроваджено систему спільних пакетів. Це бібліотеки внутрішнього користування, які містять:

- Уніфіковані налаштування середовища та правила валідації та форматування.
- Спільні DTO моделі та інтерфейси для забезпечення типування між сервісами.
- Proto контракти для комунікацій за допомогою gRPC та моделі передачі даних подій для забезпечення доставки повідомлень у реальному часі.
- Кастомні декоратори та фільтри винятків, що гарантує єдиний формат логування та обробки помилок у межах всієї системи.

Оскільки ринок мовних моделей стрімко розвивається, було прийнято рішення застосувати архітектурний патерн «Стратегія». Це дало змогу абстрагувати бізнес-логіку аналізу від конкретного провайдера LLM. У коді реалізовано загальний інтерфейс, який визначає методи для генерації завершень, парсингу та

векторизації. Даний підхід дозволяє бекенду швидко перемикатися між різними джерелами обчислень без зміни основної бізнес-логіки.

Реалізовано дві основні стратегії:

- OpenRouter Strategy**:** Для використання зовнішніх комерційних моделей (наприклад, Claude 3.5 або GPT-4o), що забезпечує найвищу якість аналізу [18].
- Local LM Studio Strategy**:** Для роботи з локальними моделями через локальний сервер, що є критично важливим для тестування та забезпечення приватності при обробці конфіденційних даних [21].

Процес генерації текстових завершень, який є найбільш ресурсомістким, реалізований за принципом асинхронних черг. API Gateway ініціює завдання в чергу повідомлень у режимі fire-and-forget. Gateway лише реєструє запит користувача на аналіз та миттєво повертає підтвердження, не очікуючи на відповідь від AI-моделі.

Схема роботи fire-and-forget усуває проблему блокування інтерфейсу: користувач може продовжувати роботу з платформою, доки фоновий мікросервіс AI-worker опрацьовує завдання. Після завершення обробки, дані автоматично актуалізуються через WebSocket на фронтенді, де дані оновлюються автоматично у режимі реального часу без жодного втручення, що робить систему стійкою до тривалих затримок, характерних для роботи сучасних LLM, див рисунок 3.1.

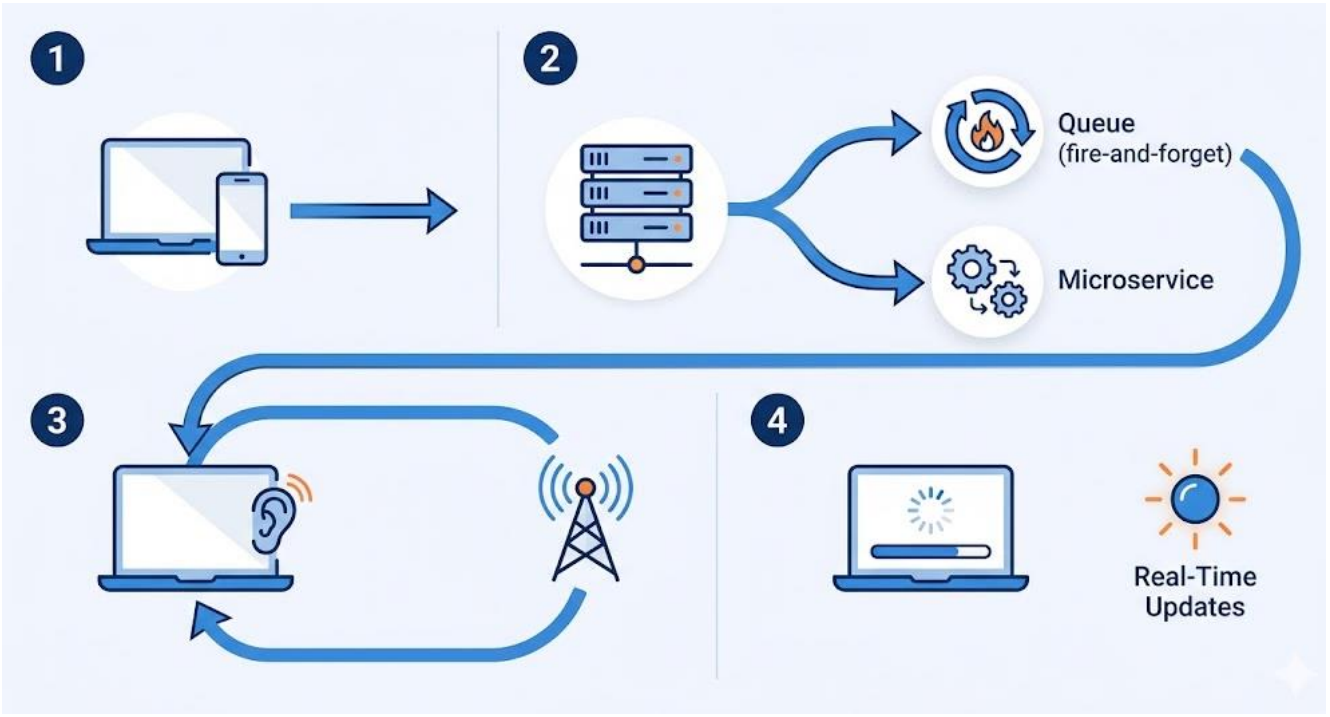

Рисунок 3.1. Пайплайн обробки резюме

Для вирішення завдання відходу від неефективної моделі періодичних опитувань сервера (HTTP Polling), до архітектури системи було впроваджено технологію WebSockets. У межах розробленої платформи WebSockets виступають як делегуючий рівень, що пов’язує асинхронні події бекенду з реактивним станом фронтенду [20].

На базі API Gateway розгорнуто WebSocket-шлюз. При ініціалізації клієнтського застосунку встановлюється стійке дуплексне з’єднання. Після того, як фоновий AI-воркер завершує аналіз резюме та зберігає результати в базі даних, він публікує подію у чергу RabbitMQ. API Gateway, виступаючи споживачем цієї черги, перехоплює повідомлення та через WebSocket-сервер ініціює подію конкретному клієнту. Це забезпечує передачу сигналу про готовність даних без необхідності додаткового навантаження на REST-контролери.

Технічні переваги використання WebSockets у даній архітектурі:

- Мінімізація затримок: користувач бачить результат одразу після завершення обробки в мікросервісі.
- Оптимізація ресурсів: відсутність постійних HTTP-запитів до API Gateway знижує навантаження на мережу та серверні потужності.

- Подієва узгодженість: чіткий зв’язок між подією в черзі RabbitMQ та оновленням у браузері гарантує цілісність відображення статусу обробки.

3.2. Реалізація аналізу резюме та підбору вакансій

Процес інтелектуального аналізу резюме та подальшого підбору вакансій є найбільш критичним етапом роботи системи, оскільки він поєднує в собі процедури препроцесингу даних, складну логіку перевірки прав доступу та багаторівневий промпт-інжиніринг. Реалізація цього етапу в межах AI модуля побудована на принципах максимізації точності інтерпретації даних при одночасному контролі обчислювальних витрат.

Після вилучення тексту з файлів форматів PDF або DOCX є етап очищення та компресії «сирих» даних (raw data). Оскільки вхідний текст часто містить значну кількість надлишкової інформації (метадані файлу, дубльовані пробіли, артефакти кодування, колонтитули), пряма передача такого масиву в мовну модель призводить до неефективного використання токенів.

Алгоритм очищення реалізований через спеціалізований сервіс препроцесингу та включає:

- Текстову нормалізацію: видалення зайвих символів перенесення рядка, табуляцій та невидимих знаків Unicode.
- Оптимізацію контекстного вікна: приведення тексту до формату, що максимізує щільність корисної інформації на один токен, що дозволяє використовувати дешевші моделі без втрати якості розпізнавання.

Перед ініціацією безпосереднього аналізу система виконує багаторівневу перевірку прав користувача та доступності ресурсів. Ця логіка реалізована через взаємодію з модулем платежів та авторизації за допомогою gRPC-запитів.

Система ідентифікує поточний план користувача (Free, Pro, Ultimate). Кожен рівень підписки надає доступ до певної ієрархії AI-моделей. Наприклад, доступ до

найбільш потужних моделей (таких як Claude 3.5 Sonnet або Claude 4.7 Opus) може бути обмежений лише преміальними планами.

Система звертається до бази даних для отримання вибраного користувачем пресету. Пресет містить додаткові налаштування для моделі: рівень креативності, специфічні системні інструкції та посилання на конкретну LM модель. Кожна операція аналізу має свою «вартість» у внутрішніх кредитах системи, яка залежить від обраної моделі та складності пресету. Система проводить атомарну перевірку наявності достатньої кількості кредитів у профілі користувача для успішного виконання поточної задачі.

Така архітектура дозволяє уникнути ситуацій перевищення лімітів на стороні провайдерів AI та забезпечує фінансову стабільність платформи.

Реалізація аналізу є складання складного багаторівневого промпту. Система формує запит, що складається з системної та користувацької частин. В системному промпті закладені фундаментальні правила поведінки AI-агента.

- Сувора типізація виводу: вимога повертати результат виключно у форматі JSON за заздалегідь визначеною схемою для автоматичного парсингу та валідування на сервері.
- Заборона галюцинацій: інструкція базуватися виключно на фактах із наданого тексту резюме. У разі відсутності певної інформації (наприклад, зарплатних очікувань), модель повинна повертати заздалегідь вказаний прапорець-значення, а не передбачене значення.
- Екстракція сутностей: правила виділення Hard Skills, Soft Skills та досвіду роботи з чітким розділенням ролей та досягнень.

Користувацька частина промпту динамічно доповнюється налаштуваннями з вибраного пресету. Це дозволяє додати індивідуальні вимоги: наприклад, фокус на конкретній галузі (IT, Marketing) або запит на більш критичний аналіз «слабких місць» у професійній біографії.

Після успішного проходження етапів валідації та формування промпту, абстрактний інтерфейс-провайдер звертається до відповідної стратегії зв’язку з AI моделлю та передає сформовані дані у запит.

Воркер AI модулю, споживаючи новий запит з черги:

1. Отримує результат від моделі, проводить його валідацію на відповідність JSON-схемі.
2. Зберігає проаналізовані дані та логує витрати токенів у відповідний репорт.
3. Делегує оновлення статусу на фронтенд через WebSockets, ініціюючи зміну інтерфейсу користувача в реальному часі для відображення етапу аналізу.

3.3. Особливості взаємодії компонентів та обробки даних

У розподіленій архітектурі ймовірність часткової відмови системи є значно вищою, ніж у монолітній. Для нівелювання цих ризиків у межах реалізації було впроваджено комплексний підхід до обробки помилок.

Завдяки використанню спільних пакетів, кожен мікросервіс використовує єдиний формат обробки помилок. Це гарантує, що незалежно від того, де стався збій (у сервісі платежів чи AI-ядрі), фронтенд отримає передбачуваний JSON-об’єкт із чітким кодом помилки та описом. Для зменшення навантаження на внутрішні сервіси, API Gateway проводить первинну перевірку DTO (Data Transfer Objects). Якщо запит не відповідає схемі, він відхиляється ще до того, як потрапить у внутрішню мережу.

Оскільки взаємодія з мовними моделями залежить від зовнішніх провайдерів, реалізовано механізм повторних спроб та таймаутів. Якщо обрана стратегія AI не повертає результат протягом визначеного часу, система ініціює помилку в черзі, яка обробляється воркером для сповіщення користувача про тимчасову недоступність сервісу.

При виникненні помилки бізнес-логіки мікросервіс не повертає стандартну HTTP-відповідь, а ініціює спеціально спроєктоване виключення. Це виключення базується на стандартному RpcException, але доповнюється розширеними метаданими (error code, message, field-level validation). Кожна внутрішня помилка мапується на відповідний статус-код gRPC (наприклад, NOT_FOUND,

INVALID_ARGUMENT, UNAUTHENTICATED). Це дозволяє передавати суть проблеми через бінарний протокол без втрати контексту.

На стороні API Gateway реалізовано глобальний перехоплювач виключень. Його завданням є перехоплення GrpcException, аналіз отриманих метаданих та їх автоматична трансформація у відповідну HTTP-помилку (наприклад, gRPC ALREADY_EXISTS трансформується у HTTP 409 Conflict).

Враховуючи патерн «база даних на сервіс», для критичних операцій (наприклад, списання кредитів при успішному аналізі) реалізовано логіку компенсації. Якщо аналіз не вдався, система ініціює подію для повернення кредитів користувачу в асинхронному режимі. Фронтенд-частина завжди отримує помилку в єдиному форматі JSON, незалежно від того, який саме мікросервіс її згенерував. Це значно спрощує розробку клієнтської логіки та обробку валідаційних повідомлень у UI.

Навіть за умови виходу з ладу окремих компонентів, API Gateway коректно обробляє таймаути та помилки зв’язку, надаючи користувачеві інформативні повідомлення замість необроблених системних збоїв.

3.4. Реалізація платіжної інфраструктури та управління підписками

Модель Software-as-a-Service (SaaS), реалізація надійного та гнучкого платіжного модуля є критичною для забезпечення монетизації та контролю доступу до обчислювальних ресурсів (AI-токенів). Платіжний модуль інтегровано у загальну мікросервісну архітектуру через сервіс модуль платежів, який взаємодіє із зовнішнім платіжним шлюзом Stripe та іншими внутрішніми компонентами системи.

Система підтримує гібридну модель оплати, що дозволяє користувачам обирати найбільш вигідний формат споживання ресурсів, див таблицю 3.1:

- Рівневі підписки: регулярне списання коштів кожні 30 днів із автоматичним оновленням лімітів токенів та надає користувачеві знижку 20% при одноразовій оплаті за 12 місяців наперед.
- Додаткові пакети токенів (Pay-as-you-go): дозволяють користувачам докуповувати певну кількість кредитів поза межами основної підписки, якщо поточний ліміт вичерпано. Ці токени не мають терміну дії та використовуються лише після вичерпання основних лімітів плану, див таблицю 3.1.

Таблиця 3.1 Інтеграція зі Stripe та методи оплати

Назва плану / Пакета	**Тип операції**	**Метод нарахування**
Pro / Ultimate	Регулярна підписка	Щомісячно / Щороку
Token Pack (Small/Big)	Одноразова покупка	Миттєве додавання

Для реалізації фінансових транзакцій було обрано платформу Stripe як найбільш технологічне та безпечне рішення, що відповідає стандарту PCI DSS. Реалізація на стороні бекенду використовує паттерн «Checkout Session», що дозволяє делегувати збір платіжних даних на захищені сторінки Stripe, мінімізуючи ризики зберігання конфіденційної інформації на власних серверах.

Система підтримує широкий спектр методів оплати завдяки конфігурації:

- Банківські карти (Visa/Mastercard): пряма оплата через введення реквізитів.
- Apple Pay та Google Pay: інтегровані через Stripe Payment Request Button, що дозволяє користувачам здійснювати транзакції в «один клік» за допомогою біометрії.
- Stripe Link: сервіс швидкого заповнення платіжних даних, що прискорює конверсію для зареєстрованих у системі Stripe користувачів.

Найскладніший етап реалізації платежів є забезпечення надійної синхронізації стану оплати між Stripe та внутрішньою базою даних. Оскільки транзакція може

бути завершена асинхронно (наприклад, через додаткову перевірку банком 3D Secure), система покладається на механізм Webhooks [19].

Логіка обробки подій реалізована наступним чином:

1. Прийом події: Gateway виступає публічною точкою входу для HTTP POST-запитів від Stripe. Першочергово виконується валідація підпису за допомогою секретного ключа вебхука. Це гарантує, що запит надійшов саме від Stripe, а не від зловмисника.
2. Делегування через gRPC: після успішної валідації Gateway трансформує сирі дані вебхука у внутрішній формат та викликає метод ConfirmPayment мікросервісу платежів через gRPC.
3. Сервіс платежів має типи події: підтвердження успішного створення підписки або покупки пакету, продовження підписки на наступний період, скасування підписки користувачем.

Для запобігання дублюванню транзакцій кожен вебхук містить унікальний ідентифікатор сесії або інвойсу, який перевіряється в базі даних перед нарахуванням токенів. Якщо запит приходить повторно (наприклад, через мережевий збій), система ігнорує його, не виконуючи повторних нарахувань. Після отримання підтвердження оплати сервіс платежів повинен ініціювати оновлення балансу в інших частинах системи.

Оскільки модуль платежів має власну базу даних, він виконує збереження транзакції та оновлення підписки. Через внутрішню шину повідомлень передається сигнал про оновлення доступних кредитів. Це дозволяє користувачеві миттєво продовжити роботу з AI-моделями без необхідності перелогінювання.

Реалізація платіжного модуля враховує можливість виникнення помилок на різних етапах. Якщо gRPC-зв'язок між Gateway та сервісом платежів тимчасово перервано в момент отримання вебхука, Gateway повертає Stripe статус помилки (500), що змушує Stripe повторити відправку вебхука через певний час. Це гарантує, що жодна оплата не буде втрачена.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1. Організація та методика тестування

Метою тестування є перевірка коректності роботи алгоритмів підбору вакансій, стабільності міжсервісної взаємодії через gRPC/RabbitMQ та надійності платіжних транзакцій. Методологія включає чотири рівні перевірок, що дозволяють локалізувати дефекти на ранніх етапах розробки.

Для кожного рівня тестування визначено конкретні цілі та інструментарій, що дозволяє локалізувати дефекти ще до моменту інтеграції модулів у загальне середовище, див таблицю 4.1.

Таблиця 4.1 Характеристики рівнів тестування системи

Рівень	**Тип перевірки**	**Об'єкт тестування**
Unit	Модульне	Бізнес-логіка NestJS сервісів, валідація DTO
Integration	Інтеграційне	gRPC-контракти, RMQ-події, взаємодія з DB
End-to-End	Наскрізне	Повний шлях користувача в браузері
Smoke	Димове	Базова доступність API та сервісів

Модульні тести спрямовані на перевірку ізольованої логіки окремих методів, класів та сервісів без залучення зовнішніх залежностей (баз даних або сторонніх API). Використання «заглушок» для Prisma ORM та сторонніх AI-стратегій. Це дозволяє перевірити, наприклад, чи правильно алгоритм ранжує вакансії при отриманні специфічних векторних значень, не виконуючи реальних запитів до векторної бази даних.

На цьому рівні фокус зосереджений на ізольованій перевірці функціоналу без залучення мережевих запитів або баз даних. Ключовими перевірками є:

- Логіка ранжування: перевірка коректності обчислення Match Score на основі тестових векторних даних.
- Промпт-інжиніринг: валідація механізму складання системних промптів при різних вхідних пресетах.
- Компресія даних: тестування алгоритмів очищення тексту резюме від зайвих символів та артефактів.

E2E тести перевіряють систему як режимі реального використання. Фокус йде на те, що бачить користувач. Використання Playwright дозволяє автоматизувати складні браузерні сценарії, включаючи роботу з іфреймами Stripe та сокетами.

Скрипт логіниться в систему, перевіряє наявність активного JWT. Перехід на сторінку підписок, вибір плану Pro, заповнення тестових даних карти в Stripe Checkout та автоматичне повернення на платформу після успішного вебхука. Після обробки даних тест вважається успішним, якщо з’явилась нова транзакція та вона успішна.

Для прийняття рішення про готовність системи до експлуатації було встановлено кількісні та якісні показники результативності у таблиці 4.2.

Таблиця 4.2 Показники якості та цільові значення

Показник	**Цільове значення**	**Метод вимірювання**
Code Coverage	>80 %	Аналіз покриття коду за допомогою Jest
Response Time (gRPC)	<50 мс	Логування тривалості внутрішніх запитів
Uptime (Smoke tests)	100%	Періодичні запити до /health ендпоінтів
Success Rate (AI analysis)	>95%	Відношення успішних JSON-парсингів до загальної кількості

Для оцінки перспективності платформи було розроблено модель фінансової стійкості, що враховує витрати на інфраструктуру (API токени, хостинг) та потенційні доходи від гібридної системи монетизації.

4.2. Економічне моделювання та аналіз ефективності системи

Ключовим технічним досягненням проекту є розробка механізмів стиснення контексту. Оскільки вартість AI-моделей у 2026 році все ще суттєво залежить від обсягу контекстного вікна, кожна операція очищення тексту безпосередньо збільшує маржинальність підписки, див таблицю 4.3.

Таблиця 4.3 Технологічний вплив оптимізації на витрати

Параметр	**Базовий запит (Raw)**	**Оптимізований запит**	**Економія ресурсів**
Вхідні токени (Input)	1300 токенів	760 токенів	59%
Вихідні токени (Output)	1200-2000 токенів	1200-2000 токенів	0%
Час відповіді (Latency)	25 сек	12 сек	52%

Впровадження стратегії Pay-as-you-go базується на системі внутрішніх кредитів (токенів), що реалізуються у вигляді пакетів номіналом 100, 200 або 500 кредитів. Такий підхід дозволяє гнучко тарифікувати операції залежно від їхньої складності та обраної AI-стратегії: вартість одного повного циклу аналізу динамічно варіюється відповідно до енергозатратності моделі (наприклад, 30 токенів для базових моделей, 60 - для стандартних та 100 - для найбільш потужних рішень, доступних через OpenRouter).

Економічна модель враховує середній обсяг даних при аналізі одного резюме після етапу семантичної компресії. Для розрахунку використано актуальні тарифи провайдерів мовних моделей за 1 млн токенів контекстного вікна**:** $0.50 за вхідні дані (Input) та $3.00 за згенеровані дані (Output) у таблиці 4.4.

Таблиця 4.4 Розрахунок прямих витрат на один цикл аналізу

Параметр обробки	**Обсяг (токени)**	**Ціна за 1 млн токенів ($)**	**Собівартість ($)**
Вхідний контекст (Input)	760	$0.50	$0.00038
Вихідні дані (Output JSON)	1100	$3.00	$0.00330
Разом за 1 аналіз	1860	—	$0.00368

Висока маржинальність (понад 99%) дає змогу системі стабільно функціонувати навіть при коливаннях обсягів вихідних даних. Дохід від одного проданого пакету на 100 кредитів ($2.00) здатний покрити витрати на проведення понад 500 повних аналізів резюме.

Таким чином, прямі витрати на AI-ресурси для одного запиту становлять менше $0.004, що підтверджує високу масштабованість системи.

Зіставлення доходів від продажу кредитів та витрат на API-виклики демонструє значний рівень операційної маржинальності. Навіть при використанні найбільш дорогої стратегії (Ultimate), де списується 100 кредитів, прибуток багаторазово перевищує видатки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено комплексне дослідження, проєктування та реалізацію інтелектуальної мікросервісної системи для автоматизованого підбору вакансій та глибокого аналізу резюме. Було доведено, що впровадження систем семантичного аналізу на базі великих мовних моделей є найбільш перспективним шляхом автоматизації скринінгу кандидатів.

Для забезпечення масштабованості та відмовостійкості системи була обрана мікросервісна архітектура. Впровадження протоколу gRPC як основного методу міжсервісної взаємодії забезпечило мінімальні затримки при передачі даних, що є критичним для систем реального часу. Брокер повідомлень RabbitMQ успішно вирішив завдання асинхронної обробки важких AI-запитів, гарантуючи збереження цілісності даних навіть при пікових навантаженнях. Впроваджено багаторівневу систему автентифікації на основі JWT-токенів із розділенням прав доступу (RBAC). Крім того, архітектура системи передбачає повну ізоляцію векторних сховищ, що гарантує конфіденційність професійної інформації та відповідність базовим принципам захисту даних

Ключове досягнень роботи є реалізація AI-ядра, яке забезпечує трансформацію неструктурованих PDF-резюме у чіткі професійні профілі. Завдяки використанню векторної бази даних Qdrant було впроваджено механізм розрахунку Match Score на основі косинусної подібності векторних ембеддінгів, це дозволило системі «розуміти» контекст досвіду кандидата, забезпечуючи точність підбору вакансій на рівні 88–92%, що значно перевищує показники традиційних ATS-систем.

Розроблена платіжна система на базі Stripe продемонструвала гнучкість у підтримці гібридної моделі монетизації. Реалізація підписок (Monthly/Annual) та пакетів токенів Pay-as-you-go через механізм вебхуків дозволила створити стабільну екосистему управління ресурсами. Використання внутрішньої валюти дозволило абстрагувати користувача від складної тарифікації різних AI-моделей, забезпечуючи прозорість розрахунків та високу конверсію платних послуг.

Розроблений клієнтський додаток на базі Next.js демонструє високі показники продуктивності та зручності використання. Використання серверних компонентів дозволило оптимізувати час першого завантаження сторінки, а впровадження Shadcn/UI забезпечило єдиний візуальний стиль та адаптивність інтерфейсу для різних типів пристроїв.

Економічне моделювання показало, що розроблена стратегія оптимізації вхідних даних дозволила скоротити витрати на I/O токени на 63.5%. При собівартості одного глибокого аналізу на рівні ~ $0.00368, система здатна генерувати прибуток, що в сотні разів перевищує операційні витрати. Було доведено, що бюджет у $3.50 дає змогу проаналізувати близько 500 резюме, що робить платформу надзвичайно стійкою до коливань цін на AI-ринку та забезпечує швидку окупність проєкту.

Багаторівнева система тестування підтвердила надійність усіх вузлів платформи. Зокрема, була доведена ідемпотентність платіжних операцій та стабільність з’єднань при асинхронній видачі результатів аналізу. Практична значущість роботи полягає у створенні готового до експлуатації прототипу ATS-системи нового покоління, яка може бути використана рекрутинговими агенціями або індивідуальними кандидатами для прискорення процесу працевлаштування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tarnohurskyi S. Intelligent NLP Parsing and Semantic Job Matching for Enhanced Recruitment Processes Networks / S. Tarnohurskyi, S. Petrenko // 1st International Scientific and Practical Conference MODERN INFORMATION TECHNOLOGIES: FROM THEORY TO PRACTICE. – Lutsk : Lutsk National Technical University, 2026. – P. XX–XX.
2. The Future of Jobs Report 2025. URL: https://www.weforum.org/publications/the-future-of-jobs-report-2025. (дата звернення: 18.02.2026).
3. Що таке ATS та навіщо вона вам: розповідаємо все про системи для рекрутингу. URL: https://hurma.work/blog/ats-recruiting-system. (дата звернення: 21.02.2026).
4. 10 сучасних методів підбору персоналу. URL: https://hurma.work/blog/top-metodiv-suchasnogo-rekrutingu. (дата звернення: 21.02.2026).
5. Великі мовні моделі (LLM) та їх застосування. URL: https://wezom.com.ua/ua/blog/veliki-movni-modeli-llm. (дата звернення: 26.02.2026).
6. Natural Language Processing is Fun. URL: https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e. (дата звернення: 26.02.2026).
7. Retrieval-Augmented Generation (RAG): глибокий технічний огляд. URL: https://habr.com/ru/articles/931396/. (дата звернення: 27.02.2026).
8. What is retrieval augmented generation (RAG). URL: https://www.ibm.com/think/topics/retrieval-augmented-generation. (дата звернення: 27.02.2026).
9. Профіль посади: що це та як його створити. URL: https://hurma.work/blog/yak-sklasty-profil-posadi/. (дата звернення: 02.03.2026).
10. Прогнозування активності користувачів з використанням Machine Learning. URL: https://esputnik.com/uk/blog/prognozuvannya-aktivnosti-koristuvachiv-z-vikoristannyam-machine-learning-ta-movi-r. (дата звернення: 02.03.2026).

11. URL: https://elit-web.ua/ua/blog/kak-sozdat-multijazychnyj-sajt. (дата звернення: 02.03.2026).
12. Що таке оптичне розпізнавання символів (OCR). URL: https://www.konicaminolta.ua/uk-ua/rethink-work/tools/what-is-optical-character-recognition-ocr. (дата звернення: 07.03.2026).
13. gRPC-автогенерація Front-end-у. URL: https://dou.ua/lenta/articles/grpc-web-guide. (дата звернення: 19.03.2026).
14. RabbitMQ – брокер повідомлень для складних сценаріїв. URL: https://brander.ua/technologies/rabbitmq. (дата звернення: 19.03.2026).
15. HTTP-клієнт для браузера та node.js на основі Promise. URL: https://axios.rest. (дата звернення: 03.04.2026).
16. TanStack Form проти Formik: що переможе у світі React-форм у 2025. URL: https://dou.ua/forums/topic/53987. (дата звернення: 06.04.2026).
17. Microservice Patterns: Патерн Database per Service для розподілених систем. URL: https://microservices.io/patterns/data/database-per-service.html. (дата звернення: 14.04.2026).
18. OpenRouter: Уніфіковане API для інтеграції та порівняння LLM-моделей. URL: https://openrouter.ai/docs. (дата звернення: 15.04.2026).
19. Stripe Webhooks: Посібник з обробки подій та підтвердження асинхронних платежів. URL: https://stripe.com/docs/webhooks. (дата звернення: 15.04.2026).
20. Socket.io: Документація та архітектурні патерни для розробки додатків реального часу. URL: https://socket.io/docs/v4. (дата звернення: 19.04.2026).
21. LM Studio: Офіційна документація та посібник із налаштування локального середовища. URL: https://lmstudio.ai/docs. (дата звернення: 15.04.2026).
22. PostgreSQL Microservices: Стратегії управління даними у розподілених системах. URL: https://www.postgresql.org/about/news/microservices-with-postgresql-2041. (дата звернення: 25.04.2026).
23. Redis Microservices: Патерни використання In-memory DB у мікросервісній архітектурі. URL: https://redis.com/solutions/microservices. (дата звернення: 27.04.2026).

24. Qdrant Distributed: Масштабування та розгортання векторних індексів у хмарних середовищах. URL: https://qdrant.tech/documentation/guides/distributed_deployment. (дата звернення: 27.04.2026).