Міністерство освіти і науки України
Національний університет “Києво-Могилянська академія”
Факультет інформатики
Кафедра інформатики

Магістерська робота

освітній ступінь – магістр

на тему: “Розробка AI-асистента для автоматизованого аналізу кандидатів на базі low-code підходів”

Виконав: студент 2-го року навчання,
спеціальності 121 Інженерія програмного забезпечення

Казістов Владислав Юрійович

Керівник Нагірна Алла Миколаївна,
завідувач кафедри, доцент,
кандидат фізико-математичних наук

Рецензент _____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою _____

Секретар ЕК _____
“___” _____ 20___ р.

Київ – 2026

ГРАФІК ПІДГОТОВКИ МАГІСТЕРСЬКОЇ РОБОТИ ДО ЗАХИСТУ

Додаток А

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника. Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	жовтень			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	жовтень – листопад			
3.	Складання плану кваліф. роботи та узгодження з науковим керівником	листопад			
4.	Написання розділів роботи	листопад – березень			
5.	Проміжний контроль виконання роботи	31 січня			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	січень – березень			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)				
	Розділ 2 (аналітично-дослідницька частина)				
	Розділ 3 (проєктно-рекомендаційна частина)				
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	квітень – початок травня			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності	середина травня			
9.	Подання на зовнішню рецензію	до 6 квітня			
10.	Підготовка до захисту кваліфікаційної роботи на засіданні	до 27 квітня			

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
	кафедри: написання доповіді та виготовлення ілюстративного матеріалу				
11.	Попередній захист кваліфікаційної роботи на засіданні кафедри	27 квітня			
12.	Подання кваліфікаційної роботи на кафедру з усіма супровідними документами	до 20 травня			
13.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	4–5 червня			

Графік узгоджено “___” _____ 2026 р.

Науковий керівник Нагірна Алла Миколаївна (ПІБ)

Виконавець кваліфікаційної роботи Казістов Владислав Юрійович (ПІБ)

Національний університет “Києво-Могилянська академія”

Факультет Інформатики (ФІ)

Кафедра Інформатики

Освітній ступінь Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітня/освітньо-наукова програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри _____

“___” _____ 20___ року

З А В Д А Н Н Я

ДЛЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Казістов Владислав Юрійович

(прізвище, ім'я, по батькові)

1. Тема роботи Розробка AI-асистента для автоматизованого аналізу кандидатів на базі low-code підходів

керівник роботи Нагірна Алла Миколаївна, завідувач кафедри, доцент, кандидат фізико-математичних наук

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “___” _____ 20___ року № ___

2. Строк подання студентом роботи 20 травня 2026 року

3. План роботи

ЗМІСТ

АНОТАЦІЯ

Казістов В. Ю. Розробка AI-асистента для автоматизованого аналізу кандидатів на базі low-code підходів. Магістерська робота. Національний університет “Києво-Могилянська академія”, Київ, 2026.

Роботу присвячено розробці та оцінці AI-асистента HireAI для автоматизованого скорингу кандидатів засобами low-code платформ. Реалізовано повнофункціональний веб-застосунок на стеку WeWeb (фронтенд) та Xano (бекенд) з інтеграцією Gemini API для аналізу резюме. Проведено порівняльний аналіз платформ фронтенду (FlutterFlow, WeWeb) та бекенду (Firebase, Xano), обґрунтовано вибір технологічного стеку з урахуванням функціональності, безпеки та швидкості розробки. Описано архітектуру системи з розмежованою фронтенд/бекенд-логікою, реалізацію захищених ендпоінтів на основі JWT-автентифікації та розмежування доступу до сторінок. Проведено функціональне тестування системи та оцінку ефективності low-code підходу для побудови AI-орієнтованих програмних систем.

Ключові слова: low-code, no-code, AI-асистент, рекрутинг, WeWeb, Xano, Gemini API, автоматизований аналіз кандидатів, HireAI.

ВСТУП

Рекрутинг сьогодні - це не просто пошук людей, це управління потоками інформації, що зростають швидше, ніж команди HR-відділів фізично здатні їх опрацьовувати. Конкуренція за кваліфікованих фахівців загострюється щороку, а кількість заявок на одну вакансію давно перестала бути керованою вручну величиною. За даними дослідження Ladders Inc., проведеного із застосуванням eye-tracking технології, рекрутер витрачає в середньому близько 7 секунд на первинний перегляд одного резюме. У технологічному секторі кількість заявок на одну позицію нерідко сягає кількох сотень. Ручна обробка такого обсягу не просто неефективна, вона системно генерує помилки відбору, зумовлені втомою та суб'єктивністю оцінювача. [11]

Паралельно з цим у сфері розробки програмного забезпечення відбувається зрушення, яке ще кілька років тому важко було передбачити. Low-code та no-code платформи поступово витісняють уявлення про те, що складний застосунок обов'язково потребує великої команди і місяців розробки. Аналітична компанія Gartner прогнозує, що до кінця 2026-го року понад 75% нових застосунків будуть створені з використанням low-code інструментів. Це не лише про швидкість, це про принципово інший поріг входу у створення програмних продуктів. Водночас активний розвиток великих мовних моделей і відкритий доступ до AI API зробили можливим те, що раніше вимагало окремої команди ML-інженерів: інтеграцію штучного інтелекту безпосередньо в low-code середовищах без написання серверного коду.

Актуальність теми. Інтенсивне зростання конкуренції за кваліфікованих фахівців і збільшення кількості заявок на вакансії роблять ручний відбір кандидатів неефективним. Водночас у сфері розробки активно утверджується low-code, що суттєво скорочує час і витрати на створення застосунків. Дослідження практичного поєднання цих двох тенденцій методом розробки AI-асистента для автоматизованого аналізу кандидатів засобами low-code платформ - є актуальним як з наукової, так і з прикладної точки зору.

Саме на перетині цих двох тенденцій і знаходиться актуальність цього дослідження. З одного боку очевидна потреба рекрутингової галузі в інструментах автоматизованого аналізу кандидатів. З іншого брак систематизованих досліджень того, чи здатні low-code підходи забезпечити побудову повноцінних AI-орієнтованих систем із розмежованою фронтенд/бекенд архітектурою в умовах реального проєктного циклу. Наявні наукові роботи, як правило, розглядають або AI у рекрутингу як окрему тему, або можливості low-code платформ у загальному контексті, але не їхнє практичне поєднання з конкретним результатом у вигляді працюючої системи.

Зв'язок роботи з науковими програмами. Магістерська робота виконана на кафедрі інформатики Національного університету “Києво-Могилянська академія” в рамках освітньої програми “Інженерія програмного забезпечення” та відповідає напрямам досліджень у галузі розробки інтелектуальних інформаційних систем і сучасних парадигм програмної інженерії.

Мета дослідження - розробити AI-асистента для автоматизованого аналізу кандидатів засобами low-code платформ та оцінити практичну ефективність цього підходу для побудови повноцінних програмних систем.

Завдання дослідження:

- проаналізувати концептуальні засади low-code/no-code розробки та визначити актуальний стан ринку відповідних платформ;
- дослідити існуючі підходи до застосування штучного інтелекту в HR-процесах і здійснити огляд наявних рішень для аналізу кандидатів;
- сформувати систему критеріїв порівняльного оцінювання low-code платформ з урахуванням функціональності, швидкості розробки та безпеки;
- провести порівняльний аналіз платформ фронтенду (FlutterFlow та WeWeb) і бекенду (Firebase та Xano) та обґрунтувати вибір технологічного стеку;
- спроєктувати архітектуру системи HireAI з чітким розмежуванням фронтенд і бекенд-логіки;

- реалізувати AI-асистента з інтеграцією Gemini API на платформах WeWeb і Xano та оцінити ефективність розробленої системи.

Об'єкт дослідження - low-code платформи для розробки AI-орієнтованих програмних систем.

Предмет дослідження - методи та інструменти проєктування і реалізації AI-асистента для автоматизованого аналізу кандидатів засобами low-code підходів.

Методи дослідження. У роботі застосовано: порівняльний аналіз - для оцінювання low-code платформ за визначеними критеріями; метод прототипування - для ітеративної розробки системи HireAI; системний аналіз - для проєктування архітектури з розмежованою фронтенд/бекенд-логікою; метод тестування - для оцінки якості та ефективності розробленої системи.

Практична цінність роботи. Розроблена система HireAI є повнофункціональним AI-асистентом для автоматизованого скорингу кандидатів, реалізованим виключно засобами low-code платформ WeWeb і Xano. Показовим є те, що на розробку знадобилось 1-2 місяці, включаючи освоєння обох платформ з нуля. Результати порівняльного аналізу, обґрунтування вибору стеку та описана архітектура можуть бути практично корисними для розробників і команд, що розглядають low-code підхід для побудови AI-систем у HR та суміжних галузях.

Результати дослідження мають практичне значення для розробників, продуктових команд та дослідників що розглядають low-code підхід для побудови AI-орієнтованих систем. Сформована система критеріїв, обґрунтований стек WeWeb + Xano + Gemini API, описана архітектура розмежованої фронтенд/бекенд системи та задокументований підхід до інтеграції LLM через low-code бекенд можуть слугувати методологічною основою для реалізації подібних проєктів у HR та суміжних галузях.

Перспективами подальшого розвитку HireAI є розширення AI-функціональності: аналіз відповідності культурних цінностей кандидата і компанії, предиктивна оцінка потенціалу до зростання, автоматизація комунікації з кандидатами через персоналізовані AI-повідомлення. З технічної точки зору

перспективним є дослідження масштабування через перехід на вищі тарифні плани або гібридну архітектуру що поєднує low-code компоненти з традиційною інфраструктурою для критично навантажених модулів.

Структура роботи. Магістерська робота складається з анотації, вступу, трьох розділів, висновків, та списку використаних джерел. Перший розділ охоплює теоретичні основи low-code розробки і застосування AI у рекрутингу. У другому розділі проведено порівняльний аналіз платформ і обґрунтовано вибір технологічного стеку. Третій розділ присвячено проєктуванню та реалізації системи HireAI, а також оцінці її ефективності.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ LOW-CODE РОЗРОБКИ ТА AI-АСИСТЕНТІВ У HR

1.1. Концепція low-code/no-code розробки: визначення, класифікація, ринкові тенденції

Традиційна розробка програмного забезпечення десятиліттями залишалась справою вузького кола фахівців. Глибокі знання мов програмування, архітектурних патернів, інструментів розгортання, все це створювало бар'єр між ідеєю і її реалізацією. Але попит на цифрові рішення зростав значно швидше, ніж ринок встигав готувати розробників. Цей розрив і став головною причиною появи парадигми low-code та no-code розробки - підходу, що принципово знижує технічний поріг входу у створення програмних застосунків.

Термін “low-code” вперше з'явився у широкому вжитку завдяки аналітичній компанії Forrester Research у 2014 році. Ним позначали платформи, що дозволяють створювати застосунки переважно через візуальний інтерфейс з мінімальним написанням коду вручну. Сьогодні це ціла категорія інструментів з графічними конструкторами інтерфейсів, готовими компонентами та вбудованими інтеграціями, де при цьому залишається можливість дописати власний код там де стандартних засобів не вистачає. [3]

No-code платформи розвивають цю ідею ще далі, прибираючи написання коду повністю. Якщо low-code здебільшого адресований так званим “citizen developers”, тобто фахівцям з базовими технічними знаннями, то no-code орієнтований на аналітиків, менеджерів і підприємців, яким потрібна автоматизація без залучення IT-відділу.

Ключова відмінність між двома підходами полягає у рівні гнучкості. Low-code платформи підтримують роботу з API, складну логіку обробки даних, кастомні компоненти та інтеграцію з корпоративними системами. No-code рішення пропонують більш обмежений, але значно доступніший набір інструментів, якого цілком достатньо для автоматизації типових бізнес-процесів.

Ринок low-code та no-code рішень сьогодні надзвичайно різноманітний, і єдиної усталеної класифікації не існує. За функціональним призначенням платформи можна розподілити на чотири основні категорії.

Перша - платформи для розробки веб та мобільних застосунків. Вони забезпечують повний цикл створення клієнтського інтерфейсу від дизайну до публікації. Серед найбільш відомих: WeWeb, FlutterFlow, Bubble, Webflow та Adalo. Кожна займає власну нішу: WeWeb спеціалізується на веб-застосунках з акцентом на гнучкість та інтеграцію із зовнішніми бекенд-сервісами, FlutterFlow орієнтований на кросплатформенну мобільну розробку на базі Flutter від Google.

Друга категорія - платформи для розробки бекенду та управління даними. Xano, Supabase, Firebase, Airtable: всі вони надають інструменти для створення баз даних, API-ендпоінтів і бізнес-логіки без необхідності налаштовувати серверну інфраструктуру вручну. Xano зокрема позиціонує себе як “no-code backend” і забезпечує візуальне конструювання RESTful API з вбудованою логікою та автентифікацією.

Третя - платформи автоматизації та оркестрації бізнес-процесів. Make, Zapier, n8n дозволяють будувати складні сценарії автоматизації, з'єднуючи сотні різних сервісів без написання коду.

Четверта - корпоративні low-code платформи для великого бізнесу та enterprise-середовища: Microsoft Power Apps, OutSystems, Mendix, SAP Build Apps. Вони пропонують підвищений рівень безпеки і відповідність регуляторним вимогам, але відрізняються вищою вартістю та складнішою кривою навчання (рис. 1.1).

Рисунок 1.1 - Класифікація low-code/no-code платформ за функціональним призначенням

Ринок low-code та no-code платформ зростає помітно швидше за традиційний сектор розробки програмного забезпечення. За оцінками Gartner, у 2023 році його обсяг склав близько 10 мільярдів доларів, а до кінця 2026 року очікується зростання до понад 26 мільярдів із середньорічним темпом близько 25%. [2]

Forrester Research фіксує: організації що впровадили low-code підходи скорочують час розробки в середньому у 5-10 разів. Це підтверджується і на практиці. Система HireAI, описана в цій роботі, була розроблена на WeWeb і Xano приблизно за 1-2 місяці, тоді як порівнянний за функціональністю традиційний застосунок міг би займати 6-12 місяців роботи досвідченої команди. [1]

Серед ключових тенденцій варто виділити кілька. По-перше, активна інтеграція AI безпосередньо у low-code платформи: провідні гравці вже додають асистентів для генерації компонентів, автоматизації логіки та побудови запитів. По-друге, рух у бік так званої “composable architecture”, коли застосунки збираються з незалежних взаємозамінних блоків, що спілкуються через API. По-третє, зростаючий попит з боку корпоративного сектору підштовхує платформи до підвищення стандартів безпеки та відповідності вимогам зокрема GDPR.

Водночас low-code підхід має і свої обмеження. Дослідники вказують на залежність від конкретного вендора, обмежену гнучкість при нестандартних

вимогах, потенційні складнощі з масштабуванням при великих навантаженнях. Ці фактори необхідно враховувати при виборі платформи, що детально розглядається у другому розділі роботи.

Таким чином, low-code та no-code розробка це не просто черговий технологічний тренд. Це фундаментальне зрушення у підходах до створення програмного забезпечення, що відкриває принципово нові можливості для швидкої реалізації складних цифрових рішень, зокрема AI-орієнтованих систем.

1.2. Застосування штучного інтелекту в рекрутингу: огляд підходів та інструментів

Рекрутинг як бізнес-процес завжди був одним із найбільш ресурсоємних напрямів роботи HR-відділів. Пошук, первинний відбір, оцінювання та ранжування кандидатів потребують значних витрат часу і людського ресурсу, при цьому залишаючись вразливими до суб'єктивних помилок та упереджень. Саме тому рекрутинг став однією з перших HR-функцій, куди почали активно впроваджувати інструменти штучного інтелекту.

Сучасне застосування AI в рекрутингових процесах охоплює кілька напрямів, кожен з яких закриває конкретні операційні проблеми. Перший і найпоширеніший - автоматизований аналіз резюме та скоринг кандидатів. Системи на основі алгоритмів машинного навчання та обробки природної мови здатні за частки секунди проаналізувати резюме, витягти ключові параметри (досвід, навички, освіту, мовні компетенції) та зіставити їх із вимогами вакансії, формуючи числовий показник відповідності. Рекрутер при цьому отримує вже відфільтрований пул і може зосередитись на якісній взаємодії з найрелевантнішими кандидатами.

Другий напрям - автоматизація комунікації. AI чат-боти перебирають на себе функції першого контакту: відповідають на типові запитання, збирають додаткову інформацію, призначають співбесіди, надсилають сповіщення про

статус заявки. Це помітно покращує досвід кандидата і знижує рутинне навантаження на команду.

Третій - предиктивна аналітика та оцінювання потенціалу. Аналізуючи великі масиви історичних даних про найм, AI-системи будують прогностичні моделі що оцінюють не лише поточну відповідність кандидата вимогам, а й його довгостроковий потенціал у конкретній ролі.

Четвертий напрям - аналіз відеоспівбесід. Частина сучасних рішень використовує комп'ютерний зір та аналіз мовлення для оцінювання кандидатів під час записаних або живих відеоспівбесід: аналізуються мімічні реакції, темп і тональність мовлення, структура відповідей. Втім, саме цей напрям залишається найбільш дискусійним з точки зору етики та достовірності результатів.

Ринок AI-рішень для рекрутингу сьогодні достатньо зрілий і різноманітний. HireVue - один із піонерів AI-рекрутингу, що спеціалізується на відеооцінюванні кандидатів. Платформа аналізує мовлення та мімічні реакції для формування оцінки за результатами відеоспівбесіди. Попри широке корпоративне впровадження, HireVue неодноразово отримувала критику від дослідників і регуляторів щодо потенційних упереджень в алгоритмах. [22]

Workday Recruiting та SAP SuccessFactors - приклади великих корпоративних HRM-систем з вбудованими AI-модулями. Вони інтегровані у ширші екосистеми управління персоналом і орієнтовані переважно на enterprise-сегмент.

Greenhouse та Lever представляють клас сучасних ATS-систем з елементами AI-автоматизації. Пропонують інструменти для структурованого управління кандидатським потоком і базової аналітики, але глибина AI-аналізу в них суттєво скромніша порівняно з вузькоспеціалізованими рішеннями.

Окремої уваги заслуговує новий клас рішень на базі великих мовних моделей. На відміну від систем, що спирались на ключові слова і шаблонні алгоритми, LLM-базовані рішення здатні до глибокого семантичного аналізу тексту резюме, розуміння контексту і нюансів досвіду кандидата, генерації

розгорнутих аналітичних висновків. Саме цей підхід реалізований у системі HireAI.

Впровадження AI в рекрутингові процеси дає кілька очевидних переваг. По-перше, суттєве скорочення часу на первинний відбір: автоматизована система опрацьовує сотні резюме за той час, що людині знадобився б на перегляд десяти. По-друге, підвищення консистентності оцінювання, адже алгоритм застосовує однакові критерії до кожного кандидата незалежно від часу доби чи настрою. По-третє, можливість масштабування рекрутингу без пропорційного зростання команди HR (рис. 1.2).

Рисунок 1.2 - Напрями застосування штучного інтелекту в рекрутингу

Але є і серйозні обмеження. Проблема алгоритмічних упереджень залишається однією з найбільш обговорюваних: якщо навчальні дані містять упередження щодо певних демографічних груп, алгоритм їх не просто відтворює, а посилює. Показовий приклад - Amazon, яка у 2018 році відмовилась від власного AI-інструменту для відбору резюме через виявлене упередження проти жінок-кандидатів.

Є і ризик надмірної формалізації: алгоритм може відсіяти нестандартного, але потенційно сильного кандидата, чий досвід просто не вписується у шаблонні критерії. Тому більшість дослідників рекомендують розглядати AI не як замінник рекрутера, а як інструмент підтримки рішень.

Впровадження AI у рекрутинг відбувається в умовах дедалі пильнішої уваги з боку регуляторів. У Євросоюзі GDPR встановлює жорсткі вимоги щодо обробки персональних даних кандидатів, зокрема обмеження на автоматизоване прийняття рішень без участі людини. EU AI Act, прийнятий у 2024 році, відносить системи AI для відбору кандидатів до категорії “високоризикових”, що передбачає підвищені вимоги до прозорості, аудиту та документування алгоритмів. [13]

Ці правові реалії безпосередньо вплинули на архітектуру HireAI: система свідомо позиціонується як інструмент підтримки рішень рекрутера, а не автономна система відбору. Всі оцінки та рекомендації AI-модуля є допоміжними і передбачають обов'язкову участь людини у фінальному рішенні щодо кандидата.

1.3. Огляд існуючих рішень для автоматизованого аналізу кандидатів

Ринок програмних рішень для автоматизованого аналізу кандидатів є одним із найдинамічніших сегментів HR-tech індустрії. Перш ніж обґрунтовувати доцільність розробки власного рішення, варто зрозуміти що вже існує на ринку і де саме є прогалини. Тому у цьому підрозділі розглядаються основні класи систем, їхні можливості та обмеження.

Applicant Tracking System (ATS) - це базова інфраструктура сучасного рекрутингу. Такі системи централізують управління кандидатськими потоками: збирають заявки з різних джерел, зберігають резюме, відстежують статуси та підтримують комунікацію з кандидатами. Серед найвідоміших представників: Workday Recruiting, Greenhouse, Lever, iCIMS та SAP SuccessFactors.

Попри широке розповсюдження, класичні ATS мають суттєві обмеження в частині інтелектуального аналізу. Більшість із них здійснює первинний відбір на основі ключових слів, тобто простого пошуку збігів між текстом резюме та вимогами вакансії. Підхід грубий і контексту не враховує: кандидат з релевантним досвідом, але іншою термінологією, може бути відфільтрований, тоді як той, хто просто “нафарширував” резюме потрібними словами, пройде далі.

Крім того, традиційні ATS є переважно системами зберігання і управління, а не аналізу. Вони фіксують факт подачі заявки, але не дають рекрутеру реальної аналітичної підтримки. Інтеграція сучасних AI-можливостей у такі системи здебільшого реалізована як стороннє доповнення, а не органічна частина продукту.

Окремий клас рішень - спеціалізовані платформи орієнтовані безпосередньо на інтелектуальний аналіз та скоринг. Eightfold AI є однією з найбільш технологічно розвинених платформ у цьому сегменті. Система застосовує глибоке машинне навчання для аналізу резюме, прогнозування потенціалу та виявлення прихованих талантів у базі даних. Eightfold будує так звані “кар'єрні траєкторії” на основі мільйонів анонімізованих профілів, що дозволяє оцінювати кандидата не лише за поточними навичками, а й за потенціалом зростання. Але платформа орієнтована на великі корпорації і для малого та середнього бізнесу просто недоступна за ціною. [15]

Pymetrics використовує нейронаукові ігри та алгоритми машинного навчання для оцінювання когнітивних та емоційних характеристик. Платформа порівнює профіль кандидата з профілями успішних співробітників компанії і формує рекомендації щодо відповідності. Сильна сторона - акцент на зниженні алгоритмічних упереджень через регулярний аудит моделей. Слабка - підхід вузькоспеціалізований і не охоплює традиційного аналізу резюме та досвіду. [16]

HireEZ (раніше Hiretual) автоматизує пошук і залучення кандидатів через аналіз публічних профілів у LinkedIn, GitHub та інших джерелах. Система збирає і структурує інформацію, формуючи розширені профілі. Головне обмеження - залежність від публічно доступних даних і питання відповідності вимогам GDPR.

Поява великих мовних моделей відкрила принципово новий клас рішень для аналізу кандидатів. На відміну від систем на основі ключових слів або класичного машинного навчання, LLM розуміють семантику тексту резюме, враховують контекст, виявляють неочевидні зв'язки між досвідом кандидата і вимогами вакансії, формулюють розгорнуті аналітичні висновки у природній мові.

Серед перших комерційних рішень цього класу - ChatGPT-based плагіни та інтеграції що з'явились у 2023 році, а також продукти на кшталт Fetcher, Beamery та оновленого SmartRecruiters з GPT-модулем. Але більшість із них або є надбудовами над існуючими ATS, або залишаються дорогими корпоративними продуктами, недоступними для малого та середнього бізнесу.

Якщо систематизувати огляд, можна виділити кілька спільних обмежень що характерні для переважної більшості розглянутих рішень. По-перше, висока вартість: enterprise-рішення типу Eightfold AI або Workday Recruiting передбачають річні підписки на десятки і сотні тисяч доларів. По-друге, складність кастомізації - більшість платформ пропонують стандартизовані рішення що погано адаптуються під специфіку конкретної компанії або галузі. По-третє, непрозорість алгоритмів: рекрутер отримує числовий скоринг без жодного пояснення чому саме такий результат, що ускладнює і довіру до системи, і аргументацію рішень перед кандидатами.

Саме ці обмеження і визначили концепцію HireAI. На відміну від описаних рішень, система реалізує LLM-базований аналіз засобами low-code платформ, що дає суттєво нижчу вартість і час розробки, можливість кастомізації під конкретні потреби та прозорість аналізу: система не просто виставляє оцінку, вона генерує розгорнутий аналітичний звіт по кожному кандидату у зрозумілій для рекрутера формі (рис. 1.3).

Порівняльний огляд класів рішень для аналізу кандидатів

Клас рішень	Представники	Переваги	Обмеження	
Класичні ATS	Workday, SAP SuccessFactors	Централізація процесу	Відсутність AI-аналізу	
Спеціалізовані AI-платформи	HireVue, Pymetrics, Eightfold AI	Глибокий аналіз кандидатів	Висока вартість, вузька спеціалізація	
ATS з AI-модулями	Greenhouse, Lever	Зручність інтеграції	Обмежена гнучкість AI-аналізу	
LLM-орієнтовані рішення (HireAI)	GPT, Gemini, Claude API	Контекстний аналіз, гнучкість	Потребує настройки промптів	← HireAI

Рисунок 1.3 - Порівняльний огляд класів рішень для автоматизованого аналізу кандидатів

Висновки до розділу 1

У першому розділі розглянуто теоретичні основи двох ключових технологічних напрямів, на перетині яких і знаходиться предмет цього дослідження. Встановлено що low-code та no-code розробка є швидкозростаючою парадигмою програмної інженерії, що суттєво знижує поріг входу у створення складних програмних рішень і скорочує час розробки у 5-10 разів порівняно з традиційними підходами. Визначено основні класи low-code платформ та їхні характеристики, що є підґрунтям для порівняльного аналізу у наступному розділі.

Досліджено основні напрями та інструменти застосування AI у рекрутингу, встановлено правовий і етичний контекст впровадження таких систем. Огляд існуючих рішень виявив суттєву незайняту нішу: відсутність доступних, кастомізованих та прозорих LLM-базованих інструментів аналізу кандидатів орієнтованих на малий та середній бізнес. Розробка системи HireAI спрямована саме на заповнення цієї ніші із застосуванням low-code підходу як ключового методологічного інструменту.

РОЗДІЛ 2. АНАЛІЗ ПЛАТФОРМ ТА ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ

2.1. Критерії оцінки low-code платформ: функціональність, швидкість розробки та безпека

Вибір технологічного стеку у будь-якому проєкті розробки програмного забезпечення це рішення, яке визначає все інше. У контексті low-code підходу воно набуває ще більшої ваги: платформа визначає не лише технічні можливості кінцевого продукту, а й швидкість розробки, вартість підтримки, рівень безпеки та ступінь залежності від конкретного вендора. Щоб порівняльний аналіз у наступних підрозділах був методологічно обґрунтованим, спочатку потрібно визначити систему критеріїв оцінювання.

У цьому дослідженні сформовано три укрупнені групи критеріїв: функціональність, швидкість розробки та безпека. Разом вони охоплюють як технічні, так і операційні аспекти вибору платформи для розробки AI-орієнтованого застосунку.

Перша група стосується технічних можливостей платформи, тобто того що вона взагалі здатна реалізувати в рамках конкретного проєкту.

Підтримка зовнішніх API та інтеграцій є критично важливим критерієм для HireAI, оскільки система передбачає інтеграцію з Gemini API. Платформа повинна забезпечувати гнучке налаштування HTTP-запитів, передачу заголовків авторизації, обробку JSON-відповідей та управління помилками. Якщо ця можливість суттєво обмежена, платформа одразу відпадає.

Гнучкість реалізації бізнес-логіки визначає наскільки складні алгоритмічні сценарії можна реалізувати не виходячи за межі платформи. Для системи аналізу кандидатів це включає логіку скорингу, умовні розгалуження залежно від параметрів вакансії, трансформацію та збагачення даних. Платформи що обмежують бізнес-логіку простими умовами “якщо-то” для повноцінного AI-асистента не підходять.

Управління базою даних є важливим з огляду на те, що HireAI оперує складними взаємопов'язаними сутностями: вакансіями, кандидатами, резюме,

результатами аналізу. Потрібна підтримка реляційних зв'язків між таблицями, гнучкість у визначенні схеми даних та зручні інструменти для роботи з даними.

Можливість кастомізації інтерфейсу визначає наскільки унікальним і зручним для рекрутера може бути фронтенд. Це включає підтримку кастомних компонентів, гнучку стилізацію, адаптивний дизайн та можливість підключення сторонніх UI-бібліотек.

Друга група критеріїв безпосередньо пов'язана з центральною гіпотезою дослідження: low-code підхід суттєво скорочує час від ідеї до робочого продукту.

Крива навчання та документація визначають наскільки швидко розробник без попереднього досвіду може вийти на продуктивний рівень роботи з платформою. Це особливо важливий критерій у контексті даного дослідження, адже і WeWeb і Xano освоювались з нуля безпосередньо у процесі розробки HireAI. Якість документації, наявність відеоуроків, активність спільноти та доступність реальних прикладів суттєво впливають на швидкість входження.

Наявність готових компонентів та шаблонів прискорює розробку інтерфейсу та типових функціональних блоків. Багата бібліотека компонентів дозволяє зосередитись на унікальній логіці проєкту, а не витрачати час на реалізацію базових елементів з нуля.

Зручність налагодження та тестування впливає на ефективність ітераційного циклу. Хороші інструменти для відстеження помилок і логування запитів скорочують час на виявлення і виправлення проблем.

Можливість ітеративного розгортання визначає наскільки легко вносити зміни у вже розгорнуту систему. У low-code середовищі ідеально коли правки у логіці або інтерфейсі одразу відображаються у живій системі без тривалих циклів збірки.

Безпека - критерій який при виборі low-code платформ часто недооцінюють. Але для системи що обробляє персональні дані кандидатів він є принципово важливим.

Управління автентифікацією та авторизацією є базовою вимогою. HireAI повинна забезпечувати доступ до даних кандидатів виключно авторизованим

рекрутерам. Ключове питання: наскільки платформа спрощує реалізацію цієї вимоги, чи потребує захист сторінок і ендпоінтів складного налаштування або реалізується з мінімальними зусиллями. [20]

Захист API-ендпоінтів на рівні бекенду визначає чи можна обмежити доступ до бізнес-логіки та даних виключно для авторизованих запитів. Ідеально коли платформа перевіряє авторизацію автоматично на рівні кожного ендпоінту без необхідності прописувати цю логіку вручну у кожному запиті.

Контроль доступу до даних на рівні запитів визначає можливість обмеження видимості даних залежно від ролі користувача. Рекрутер має бачити лише своїх кандидатів, а не всі записи у базі даних.

Відповідність вимогам конфіденційності стосується того як платформа забезпечує зберігання та обробку персональних даних відповідно до GDPR. Це включає питання географічного розташування серверів, шифрування даних та наявність інструментів для обробки запитів на видалення даних.

Сформована система охоплює дванадцять конкретних показників у трьох групах. Саме за ними у наступних підрозділах буде проведено порівняльний аналіз платформ фронтенду (FlutterFlow та WeWeb) і бекенду (Firebase та Xano).

Важливо підкреслити: ці критерії не є абстрактними теоретичними показниками. Вони випливають безпосередньо зі специфіки проєкту HireAI, що забезпечує релевантність аналізу та обґрунтованість кінцевого вибору стеку (рис. 2.1).

Рисунок 2.1 - Система критеріїв оцінки low-code платформ для проекту HireAI

2.2. Порівняльний аналіз платформ фронтенду: FlutterFlow vs WeWeb

Вибір платформи для розробки фронтенду є першим із двох ключових технологічних рішень у проєкті HireAI. На початковому етапі розглядались дві платформи: FlutterFlow та WeWeb, що представляють принципово різні підходи до low-code розробки клієнтської частини застосунків.

FlutterFlow - це low-code платформа для візуальної розробки кросплатформенних застосунків на базі фреймворку Flutter від Google. Вона дозволяє будувати застосунки що компілюються у нативний код для iOS, Android та веб, зберігаючи єдину кодову базу. FlutterFlow орієнтований переважно на мобільну розробку і позиціонується як рішення для команд що хочуть охопити кілька платформ одночасно. [26]

З точки зору функціональності платформа є достатньо потужною. Вона підтримує інтеграцію із зовнішніми API, має вбудовану підтримку Firebase та надає можливість кастомізації через власний Dart-код. Бібліотека готових компонентів охоплює більшість типових UI-елементів.

Але для проєкту HireAI FlutterFlow виявив кілька критичних обмежень. По-перше, платформа спроєктована з акцентом на мобільний досвід і розробка повноцінного веб-застосунку з адаптивним дизайном для десктопу значно складніша ніж для мобільного. HireAI орієнтована на рекрутерів що працюють за комп'ютером, тому якісний веб-інтерфейс є пріоритетом. По-друге, крива навчання FlutterFlow суттєво крутіша: для ефективної роботи потрібне базове розуміння концепцій Flutter та Dart, що уповільнює освоєння для розробника без відповідного досвіду. По-третє, налагодження і тестування веб-версій у FlutterFlow менш зручне порівняно з нативними веб-інструментами.

WeWeb - це low-code платформа що спеціалізується виключно на веб і позиціонує себе як “frontend builder for any backend”. На відміну від FlutterFlow, WeWeb від початку створювався для веб і не намагається охопити мобільні платформи, що дозволяє сконцентрувати всі можливості на якісному веб-досвіді. [5]

WeWeb забезпечує гнучке підключення до будь-якого бекенду через REST API або GraphQL, підтримує динамічне зв'язування даних з компонентами інтерфейсу, дозволяє розширювати функціональність через власний JavaScript-код та підключати кастомні Vue.js компоненти. Для HireAI особливо важливою є здатність платформи працювати з будь-яким бекендом: це дозволяє використовувати Xano як незалежний сервіс з чітким розмежуванням відповідальностей між фронтендом і бекендом.

Обидві платформи підтримують інтеграцію із зовнішніми API, але WeWeb робить це значно природніше для веб-застосунків. Налаштування HTTP-запитів реалізоване через менеджер колекцій, що дозволяє централізовано керувати всіма з'єднаннями з бекендом, включаючи автоматичну передачу заголовків авторизації. FlutterFlow також підтримує API-інтеграції, але налаштування менш інтуїтивне і вимагає глибшого розуміння внутрішньої архітектури платформи.

Щодо бізнес-логіки на фронтенді, WeWeb дозволяє використовувати повноцінний JavaScript будь-де в застосунку: в обробниках подій, обчислюваних властивостях, кастомних функціях. Це дає практично необмежену гнучкість. FlutterFlow надає аналогічну можливість через Dart-код, але для розробника з веб-бекґраундом JavaScript є значно природнішим вибором.

З точки зору швидкості розробки WeWeb має суттєву перевагу. Розробник з базовим розумінням HTML/CSS та JavaScript може вийти на продуктивний рівень за кілька днів. Документація структурована і актуальна, офіційний YouTube-канал містить детальні відеоуроки по всіх ключових функціях. FlutterFlow натомість вимагає значно більше часу на освоєння через стейт-менеджмент, віджетне дерево і провайдери даних. У контексті цього проєкту де обидві платформи освоювались з нуля, ця різниця виявилась вирішальною: WeWeb дозволив перейти від вивчення до продуктивної розробки помітно швидше.

З точки зору безпеки WeWeb має принципову перевагу що безпосередньо вплинула на вибір платформи. Система управління доступом реалізована на рівні сторінок через механізм Page Access: однією дією можна обмежити доступ до будь-якої сторінки виключно для авторизованих користувачів. При спробі

неавторизованого доступу система автоматично перенаправляє на сторінку входу. Цей механізм інтегрується з автентифікацією Xano де токени перевіряються на кожному захищеному запиті. У FlutterFlow аналогічний функціонал потребує значно більше ручного налаштування і власного коду, що підвищує ризик помилок у логіці доступу.

За результатами аналізу WeWeb є оптимальним вибором для реалізації фронтенду HireAI. Платформа перевершує FlutterFlow за всіма трьома групами критеріїв: забезпечує природнішу інтеграцію з зовнішнім бекендом, має значно пологішу криву навчання для розробника з веб-бекґраундом і надає вбудований захист сторінок без додаткового коду. Орієнтація WeWeb виключно на веб, яка в іншому контексті могла б здатися обмеженням, тут є перевагою: HireAI орієнтована на рекрутерів що працюють у браузері і не потребує мобільних нативних застосунків (рис. 2.2).

Порівняльний аналіз платформ фронтенду: FlutterFlow vs WeWeb

Критерій	FlutterFlow	WeWeb
Підтримка веб-орієнтованого UI	– Часткова (мобільний акцент)	✓ Повна (веб-first)
Інтеграція зовнішніх REST API	✓ Присутня	✓ Нативна, гнучка
Гнучкість JavaScript логіки	– Обмежена (Custom Code)	✓ Повна (будь-де)
Механізм захисту сторінок	✗ Відсутній нативний	✓ Page Access (вбудов.)
Крива навчання для веб-розробника	– Вища (Flutter/Dart)	✓ Нижча (HTML/CSS/JS)
Хостинг та розгортання	– Власна інфра потрібна	✓ Вбудований хостинг
Висновок для HireAI	✗ Не оптимальний	✓ Обрано

Рисунок 2.2 - Порівняльний аналіз платформ фронтенду: FlutterFlow та WeWeb

2.3. Порівняльний аналіз платформ бекенду: Firebase vs Xano

Вибір платформи бекенду є другим ключовим технологічним рішенням у проєкті HireAI. Бекенд відповідає за зберігання даних, реалізацію бізнес-логіки, інтеграцію з Gemini API та безпеку ендпоінтів. На етапі проєктування розглядались дві платформи: Firebase та Xano, що представляють суттєво різні підходи до організації бекенду у low-code середовищі.

Firebase - це платформа від Google що надає широкий набір хмарних сервісів: базу даних реального часу (Realtime Database та Firestore), систему автентифікації, хмарні функції (Cloud Functions), хостинг та аналітику. Firebase є одним із найвідоміших бекенд-сервісів як у low-code, так і у традиційній розробці, тому він розглядався як природний кандидат для проєкту. [27]

Ключова перевага Firebase - глибока інтеграція з екосистемою Google та нативна підтримка у FlutterFlow, що робить цю пару органічним вибором для певного типу проєктів. Firestore забезпечує масштабоване хмарне сховище документів із підтримкою запитів у реальному часі. Система автентифікації підтримує широкий спектр провайдерів: email/пароль, Google, GitHub та інші. [27]

Але при детальнішому аналізі в контексті HireAI Firebase виявив кілька суттєвих обмежень. По-перше, він використовує документно-орієнтовану модель даних (NoSQL), що менш природна для системи з чіткими реляційними зв'язками між вакансіями, кандидатами та результатами аналізу. Складні реляційні запити у Firestore вимагають або денормалізації даних, або кількох послідовних запитів на клієнті. По-друге, реалізація складної бізнес-логіки потребує Cloud Functions - серверних функцій на Node.js, що фактично означає написання традиційного коду і налаштування серверного середовища. Це прямо суперечить концепції low-code розробки. По-третє, Firebase не має вбудованого конструктора REST API: для захищених ендпоінтів з власною логікою потрібно писати Cloud Functions вручну.

Xano позиціонує себе як “no-code backend” і надає повний набір інструментів для створення масштабованого бекенду без серверного коду: реляційну базу даних PostgreSQL, візуальний конструктор API-ендпоінтів, вбудовану систему автентифікації та інструменти для побудови бізнес-логіки через візуальний редактор функцій. [6]

На відміну від Firebase, Xano будує бекенд навколо реляційної моделі даних. PostgreSQL - одна з найпотужніших реляційних СУБД з відкритим кодом, що дозволяє повноцінно працювати з таблицями, зовнішніми ключами, складними JOIN-запитами та транзакціями. Для HireAI з її чіткою реляційною структурою це принципова перевага. [19]

Центральний елемент Xano - візуальний конструктор API-ендпоінтів. Кожен ендпоінт налаштовується через послідовність кроків у так званому “function stack”: розробник візуально будує логіку обробки запиту, валідацію вхідних даних, запити до бази, виклики зовнішніх API, трансформацію даних та формування відповіді. Все це без написання коду.

З точки зору функціональності Xano має суттєву перевагу. Реляційна модель PostgreSQL дозволяє природно описати всі сутності системи та їхні зв'язки без обхідних рішень. Візуальний конструктор бізнес-логіки підтримує умовні розгалуження, цикли, роботу зі змінними, агрегацію даних та виклики зовнішніх сервісів, і все це без коду. Firebase для аналогічної складності логіки вимагає Cloud Functions, що фактично виводить розробку за межі low-code парадигми.

З точки зору швидкості розробки Xano також виграє. Візуальний конструктор дозволяє розробнику без серверного досвіду створити повноцінний REST API з автентифікацією і складною логікою за години, а не дні. Документація детальна і добре структурована, спільнота активно підтримується командою платформи. Firebase при всій ширині документації і великій спільноті вимагає значно глибшого технічного занурення для реалізації складної логіки.

З точки зору безпеки Xano надає вирішальну перевагу, що стала одним із ключових факторів вибору. Система автентифікації побудована навколо JWT-токенів і інтегрована безпосередньо у кожен ендпоінт. Розробник одним перемикачем активує вимогу автентифікації - і платформа автоматично відхиляє всі запити без валідного токена зі стандартизованою помилкою 401. Захист ендпоінтів це не результат ручного коду в кожній функції, а вбудована можливість що активується декларативно. Це знижує ризик помилок і забезпечує консистентний захист усіх ендпоінтів системи.

Firebase натомість надає Authentication як окремий сервіс, але перевірка у Cloud Functions вимагає ручної реалізації в кожній функції: розробник самостійно отримує та верифікує ID-токен у кожному обробнику. Це збільшує обсяг коду і підвищує ризик помилок якщо перевірку випадково пропустити в одній із функцій.

Щодо управління доступом до даних: Xano дозволяє вбудовувати логіку фільтрації безпосередньо у function stack ендпоінту. Наприклад, автоматично обмежувати вибірку кандидатів лише тими записами що належать поточному авторизованому рекрутеру, зчитуючи його ідентифікатор з JWT-токена. Firebase реалізує це через Security Rules - декларативну мову правил доступу до Firestore, достатньо потужну але з нетривіальним синтаксисом що потребує окремого вивчення.

За результатами аналізу Xano є оптимальним вибором бекенду для HireAI. Платформа перевершує Firebase за всіма трьома групами критеріїв: реляційна модель PostgreSQL природно відповідає структурі сутностей системи, візуальний конструктор забезпечує швидку реалізацію складної логіки без серверного коду, а вбудований захист ендпоінтів одним перемикачем є принциповою перевагою з точки зору безпеки і швидкості розробки (рис. 2.3).

Порівняльний аналіз платформ бекенду: Firebase vs Xano

Критерій	Firebase	Xano
Модель даних	– NoSQL (документна)	✓ PostgreSQL (реляційна)
Складні реляційні запити	– Обмежені JOIN	✓ Повна підтримка JOIN
Візуальний API-конструктор	✗ Відсутній	✓ Вбудований
JWT-автентифікація	– Auth як окремий сервіс	✓ Вбудована автоматично
Захист ендпоінтів	– Вручну (Cloud Func.)	✓ Один тумблер
Фільтрація даних за роллю	– Security Rules	✓ Вбудований фільтр
Висновок для HireAI	✗ Не обрано	✓ Обрано

Рисунок 2.3 - Порівняльний аналіз платформ бекенду: Firebase та Xano

Поєднання WeWeb як фронтенду і Xano як бекенду формує стек що забезпечує чітке розмежування відповідальностей між клієнтською і серверною частинами, природну інтеграцію через REST API та узгоджену модель безпеки на обох рівнях застосунку. Детальне обґрунтування цього архітектурного рішення розглядається у підрозділі 2.5.

2.4. Вибір AI API та обґрунтування використання Gemini API

Інтеграція штучного інтелекту є центральним функціональним елементом системи HireAI. Від вибору конкретного AI API залежить якість аналізу кандидатів, вартість експлуатації, швидкість відповіді та зручність інтеграції з бекендом на Xano. У цьому підрозділі розглядаються основні доступні AI API для роботи з текстом, визначаються критерії вибору та обґрунтовується рішення на користь Gemini API.

Станом на момент розробки HireAI ринок комерційних AI API для роботи з текстом представлений кількома ключовими гравцями.

OpenAI API надає доступ до сімейства моделей GPT, зокрема GPT-4 та GPT-4o, що є одними з найпотужніших і найвідоміших мовних моделей на ринку. Платформа має детальну документацію, широку спільноту розробників та велику кількість готових інтеграцій. Підтримуються різні режими роботи: чат-комплешени, функціональні виклики, структуровані виводи. Ціноутворення базується на кількості токенів і є відносно конкурентним, хоча найпотужніші моделі при великих обсягах запитів можуть коштувати відчутно. [18]

Anthropic Claude API вирізняється великим контекстним вікном до 200 тисяч токенів у Claude 3, підвищеним акцентом на безпеку та сильними можливостями аналізу довгих документів. Claude потенційно привабливий для аналізу резюме, але на момент початку розробки HireAI доступ до API потребував окремої реєстрації і затвердження заявки, а активного ключа не було. [17]

Google Gemini API надає доступ до мультимодальних моделей Gemini від Google. Gemini Pro та Gemini 1.5 Pro підтримують обробку тексту, зображень, аудіо та відео в єдиному API, мають велике контекстне вікно і демонструють конкурентну якість генерації. Важливою практичною перевагою є наявність безкоштовного рівня використання через Google AI Studio, що дозволяє розпочати розробку без жодних фінансових витрат. [7]

Cohere API та Mistral AI є альтернативними постачальниками LLM з акцентом на корпоративне використання. Втім обидва мають меншу спільноту,

менш зрілу документацію та обмеженішу екосистему інтеграцій порівняно з провідними гравцями.

Для обґрунтованого вибору визначено чотири ключові критерії що відповідають вимогам проєкту.

Якість аналізу текстового контенту є першочерговим. Основна функція системи - генерація змістовного аналізу резюме у контексті вимог конкретної вакансії. API повинен розуміти семантику тексту, виявляти релевантні навички навіть за різної термінології та генерувати структуровані висновки у зрозумілій для рекрутера формі.

Розмір контекстного вікна є важливим технічним параметром: аналіз кандидата передбачає одночасну обробку кількох документів в одному запиті, опису вакансії, резюме та системного промпту. Обмежене вікно призводить або до втрати частини інформації, або до складного розбиття даних на частини що ускладнює логіку бекенду.

Доступність та простота інтеграції визначає наскільки швидко можна підключити API у процесі розробки. Активний ключ, детальна документація, підтримка стандартних HTTP-запитів без специфічних SDK - все це безпосередньо впливає на швидкість розробки і зручність інтеграції з Xano.

Вартість використання є суттєвим критерієм з точки зору практичної застосовності системи. Важливо щоб модель ціноутворення дозволяла розробляти і тестувати без значних витрат, а промислове використання було прийнятним для малого та середнього бізнесу.

За результатами оцінювання для HireAI обрано Google Gemini API. Це рішення обумовлено сукупністю практичних та технічних факторів.

З точки зору доступності Gemini API виявився найзручнішим вибором. На момент початку розробки активний ключ був доступний через Google AI Studio без очікування затвердження, що дозволило одразу розпочати інтеграцію. Gemini API підтримує стандартні HTTP POST запити з JSON-тілом, що ідеально підходить для інтеграції через візуальний конструктор ендпоінтів Xano без будь-яких додаткових бібліотек.

З точки зору якості аналізу Gemini Pro та Gemini 1.5 Pro демонструють високу якість роботи з текстом, зокрема у завданнях структурованого аналізу та генерації звітів. Модель приймає детальний системний промпт з інструкціями щодо формату виводу і критеріїв оцінювання, що дозволяє точно налаштувати логіку аналізу під вимоги HireAI. У процесі тестування Gemini стабільно генерував структуровані звіти що включали оцінку відповідності навичок, аналіз досвіду, сильні сторони та потенційні ризики у формі зрозумілій для рекрутера. [7]

З точки зору контекстного вікна Gemini 1.5 Pro підтримує до одного мільйона токенів, що з великим запасом покриває потребу одночасної обробки опису вакансії і резюме в одному запиті. Навіть базова версія Gemini Pro має вікно достатнє для завдань HireAI.

З точки зору вартості Gemini API пропонує конкурентне ціноутворення з безкоштовним рівнем що є суттєвою перевагою на етапі розробки і тестування. Безкоштовний рівень Google AI Studio дозволяє виконувати достатньо запитів для повноцінного тестування, а перехід на комерційне використання передбачає прозору модель оплати за токени.

Інтеграція Gemini API реалізована на рівні бекенду Xano, і це принципове архітектурне рішення. Виклик AI API здійснюється виключно з серверної частини, а не з фронтенду WeWeb. Це дає кілька важливих переваг.

По-перше, безпека API-ключа: він зберігається у захищених змінних середовища Xano і ніколи не передається на клієнт. По-друге, централізація логіки промптингу: системний промпт що визначає формат і критерії аналізу зберігається на рівні бекенду і може оновлюватись без змін у фронтенді. По-третє, контроль витрат: всі запити до Gemini проходять через захищені ендпоінти Xano, доступні лише авторизованим користувачам, що унеможливлює несанкціоноване використання API-квоти.

Технічно інтеграція реалізована через стандартний HTTP-крок у function stack Xano: при отриманні запиту на аналіз ендпоінт формує промпт що включає опис вакансії, резюме кандидата та системні інструкції, і передає його до Gemini

API. Отримана відповідь парситься та зберігається у базі даних Xano як результат аналізу, доступний рекрутеру через інтерфейс WeWeb (рис. 2.4).

Порівняльна оцінка AI API за критеріями проекту HireAI

Критерій	OpenAI GPT-4	Anthropic Claude	Google Gemini	Cohere / Mistral
Якість аналізу тексту	Висока	Висока	Висока	Середня
Контекстне вікно	128K токенів	200K токенів	1M токенів	128K
Доступність ключа	Реєстрація+оплата	Список очікування	Миттєво безкоштовно	Реєстрація
Вартість	Платна	Платна	Безкоштовний рівень	Платна
Підсумок	**Не обрано**	**Не обрано**	**ОБРАНО ✓**	**Не обрано**

Рисунок 2.4 - Порівняльна оцінка AI API для інтеграції в систему HireAI

2.5. Обґрунтування фінального технологічного стеку та архітектурного підходу

Попередні підрозділи забезпечили детальний порівняльний аналіз альтернативних платформ фронтенду, бекенду та AI API. Тепер ці результати варто синтезувати у цілісне архітектурне рішення і пояснити логіку поєднання обраних технологій. HireAI - це не випадковий набір інструментів, а свідомий проєктний вибір де кожен компонент підсилює інші.

За результатами аналізу для реалізації системи HireAI обрано такий стек: WeWeb як платформа фронтенду, Xano як платформа бекенду та Gemini API як AI-сервіс для аналізу кандидатів. Разом вони формують узгоджену архітектуру, де сильні сторони кожного компоненту взаємно доповнюють одна одну.

Принциповою архітектурною рисою обраного стеку є чітке розмежування відповідальностей між фронтендом і бекендом. WeWeb відповідає виключно за відображення даних та взаємодію з користувачем, тоді як уся бізнес-логіка, обробка даних, інтеграція з Gemini API та безпека зосереджені на рівні Xano. Комунікація між шарами здійснюється через REST API, стандартизований протокол незалежний від конкретних платформ.

Рішення винести всю бізнес-логіку та AI-інтеграцію на рівень бекенду є одним із ключових принципів системи, і важливим з кількох точок зору.

З точки зору безпеки концентрація логіки на бекенді захищає чутливі дані і ключі доступу. API-ключ Gemini, логіка авторизації та прямий доступ до бази даних залишаються на серверній стороні і ніколи не передаються клієнту. WeWeb взаємодіє лише із захищеними ендпоінтами Xano, передаючи токен авторизації у заголовку кожного запиту. Xano перевіряє валідність токена і відхиляє неавторизовані запити ще до виконання будь-якої логіки. Тобто навіть якщо зловмисник отримає доступ до клієнтського коду WeWeb, він не зможе дістатись до даних або AI-функціональності.

З точки зору масштабованості і підтримки розмежований підхід забезпечує незалежність еволюції фронтенду і бекенду. Зміни у логіці аналізу кандидатів, оновлення промптів для Gemini або модифікація структури даних виконуються на рівні Xano без змін у WeWeb. І навпаки, оновлення інтерфейсу рекрутера не зачіпає бекенд. Це суттєва перевага порівняно з монолітними підходами де зміна одного компоненту може непередбачувано вплинути на всю систему.

З точки зору відповідності low-code парадигмі розмежований підхід дозволяє повністю реалізувати кожен шар засобами відповідної платформи без виходу за її межі. WeWeb оптимізований для веб-інтерфейсів, Xano для бекенд-логіки та API. Використання кожного інструменту за його прямим призначенням забезпечує максимальну ефективність розробки і мінімум обхідних рішень.

Одна з найважливіших переваг обраного стеку - наскрізна модель безпеки що охоплює обидва рівні системи.

На рівні фронтенду WeWeb захищає сторінки через механізм Page Access: кожна сторінка окрім сторінки входу налаштована як приватна і доступна лише авторизованим користувачам. При спробі неавторизованого доступу система автоматично перенаправляє на сторінку входу. Цей захист активується декларативно через налаштування сторінки без жодного рядка коду.

На рівні бекенду Xano захищає кожен ендпоінт через вбудований механізм автентифікації. Всі ендпоінти HireAI окрім входу та реєстрації налаштовані як захищені: Xano автоматично перевіряє наявність і валідність JWT-токена у заголовку запиту та відхиляє неавторизовані запити з помилкою 401. Токен генерується при успішному вході, зберігається у WeWeb і автоматично додається до кожного наступного запиту.

Таким чином система реалізує двошаровий захист: навіть якщо користувач якось обійде фронтенд, всі запити до бекенду будуть відхилені без валідного токена. Ця модель проста у реалізації завдяки вбудованим можливостям обох платформ, і водночас достатньо надійна для системи що обробляє персональні дані кандидатів. [20]

FlutterFlow + Firebase є природним і добре інтегрованим стеком для мобільних застосунків, особливо тих що потребують синхронізації даних у реальному часі. Але для HireAI з веб-інтерфейсом для рекрутерів, складною серверною логікою та безпечною інтеграцією з зовнішнім AI API цей стек має принципові обмеження. Бізнес-логіка вимагає Cloud Functions з ручним кодом, захист ендпоінтів потребує ручної верифікації токенів у кожній функції, а реляційна структура даних є неприродною для документно-орієнтованої Firestore. У сукупності це означає суттєво більше ручного кодування що суперечить самій концепції low-code розробки.

Найпереконливішим аргументом на користь обраного стеку є практичний результат: повнофункціональна система HireAI розроблена з нуля за 1-2 місяці одним розробником без попереднього досвіду роботи з WeWeb та Xano. Це безпосереднє підтвердження центральної гіпотези дослідження про ефективність low-code підходу для розробки AI-орієнтованих систем.

Упродовж розробки обидві платформи дозволяли зосередитись на логіці та функціональності, а не на інфраструктурних задачах. Налаштування бази даних, створення ендпоінтів, побудова інтерфейсу та інтеграція з Gemini API - все вирішувалось переважно через візуальні інструменти. Можливість бачити

результат змін у реальному часі без тривалих циклів збірки є однією з ключових практичних переваг обраного стеку.

Висновки до розділу 2

У другому розділі проведено системний порівняльний аналіз альтернативних платформ та технологій для реалізації HireAI. На основі сформованої системи критеріїв - функціональності, швидкості розробки та безпеки - обґрунтовано вибір WeWeb як фронтенду, Xano як бекенду та Gemini API як AI-сервісу.

Обраний стек забезпечує чітке розмежування відповідальностей між фронтендом і бекендом, узгоджену наскрізну модель безпеки та максимальну відповідність принципам low-code розробки без ручного серверного коду. Практична валідація підтверджується розробкою повнофункціональної системи за 1-2 місяці одним розробником без попереднього досвіду з обраними платформами, що є переконливим свідченням ефективності low-code підходу для AI-орієнтованих систем.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ AI-АСИСТЕНТА HIREAI

3.1. Архітектура системи та модель даних

Проєктування архітектури HireAI базується на принципі чіткого розмежування відповідальностей між фронтендом і бекендом. Система складається з трьох архітектурних рівнів: клієнтського на WeWeb, серверного на Xano та рівня зовнішніх сервісів у вигляді Gemini API. Взаємодія між рівнями відбувається виключно через стандартизовані REST API запити з JWT-автентифікацією.

Клієнтський рівень (WeWeb) відповідає за відображення інтерфейсу рекрутера, обробку користувацьких дій та передачу запитів до бекенду. WeWeb не містить жодної бізнес-логіки: всі операції з даними виконуються виключно через захищені ендпоінти Xano. Клієнтська частина зберігає лише JWT-токен авторизованого користувача, який автоматично додається до заголовка кожного запиту.

Серверний рівень (Xano) є центральним компонентом системи. Саме тут реалізована вся бізнес-логіка: валідація вхідних даних, взаємодія з базою даних PostgreSQL, інтеграція з Gemini API та формування відповідей для клієнта. Xano також забезпечує автентифікацію та авторизацію: кожен захищений ендпоінт автоматично перевіряє валідність JWT-токена до виконання будь-якої логіки.

Рівень зовнішніх сервісів представлений Gemini API від Google, що використовується для двох ключових операцій: парсингу PDF-файлів вакансій і резюме та аналізу відповідності кандидата вимогам вакансії. Gemini API ніколи не викликається з фронтенду, всі звернення проходять виключно через відповідні ендпоінти Xano.

Бекенд організований у чотири API-групи в Xano, кожна з яких відповідає за окремий функціональний домен системи. Така структура забезпечує логічне розмежування ендпоінтів і спрощує підтримку.

Група Authentication містить три ендпоінти: POST auth/login для входу та отримання токена, POST auth/signup для реєстрації, GET auth/me для отримання

даних авторизованого користувача. Це єдині публічні ендпоінти системи, вони не потребують JWT-токена, оскільки призначені саме для його отримання. При успішній автентифікації Xano генерує та повертає JWT-токен, який WeWeb зберігає і використовує для всіх наступних запитів.

Група google-oauth містить чотири ендпоінти що реалізують повний цикл OAuth-автентифікації через Google: oauth/google/init для ініціалізації потоку, oauth/google/continue для обробки і входу і реєстрації залежно від стану акаунту, oauth/google/login та oauth/google/signup для відповідних окремих сценаріїв. Цю групу згенерував сам Xano через вбудований OAuth-шаблон, що значно прискорило реалізацію.

Група Vacancy містить шість ендпоінтів: GET vacancy для отримання списку вакансій, POST vacancy для створення запису, POST vacancy/parse-pdf для створення вакансії через парсинг PDF, GET vacancy/{vacancy_id} для отримання деталей, PATCH vacancy/{vacancy_id} для оновлення та DELETE vacancy/{vacancy_id} для видалення. Всі ендпоінти захищені і автоматично фільтрують дані за ідентифікатором авторизованого користувача.

Група Candidate також містить шість ендпоінтів: GET candidate для отримання списку, POST candidate для створення запису, POST candidate/analyze для додавання кандидата через парсинг резюме з одночасним AI-аналізом, GET candidate/{candidate_id} для отримання деталей, PATCH candidate/{candidate_id} для редагування та DELETE candidate/{candidate_id} для видалення.

База даних HireAI побудована на PostgreSQL через Xano і містить три основні таблиці: user, Vacancy та Candidate (рис. 3.1). Реляційна модель природно відображає предметну область і забезпечує цілісність даних через зовнішні ключі.

Таблиця user зберігає дані авторизованих рекрутерів і є центральною сутністю моделі. Поля таблиці: id, created_at, name, email, password (хешований), google_oauth (JSON-об'єкт з id, name та email від Google), magic_link (JSON з token, expiration та used для відновлення паролю) та vacancy_id[] (масив зовнішніх ключів до вакансій). Наявність поля google_oauth поруч із традиційним паролем забезпечує підтримку обох способів входу в межах єдиної таблиці.

Таблиця Vacancy зберігає дані про вакансії і пов'язана з таблицею user через зовнішній ключ user_id. Поля: id, created_at, Title, Department, user_id, Company, Location, Salary, Responsibilities[] (масив обов'язків), Requirements[] (масив вимог), Skills[] (масив навичок), Status, Archived, Employment_type, file_base64 (base64-вміст завантаженого PDF) та candidates[] (масив пов'язаних кандидатів). Використання масивів для Responsibilities, Requirements та Skills дозволяє зберігати структуровані списки без необхідності окремих таблиць, спрощуючи і модель даних, і логіку API.

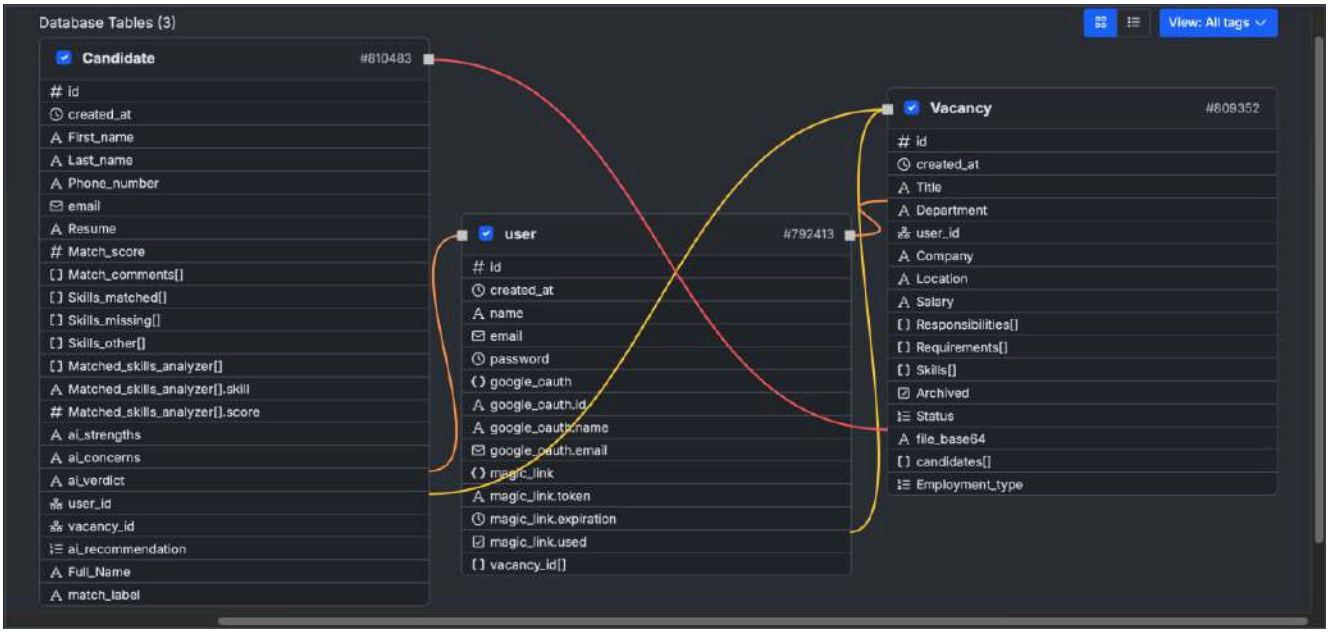

Рисунок 3.1 - Таблиці бази даних та реляційні зв’язки системи HireAI у Xano

Таблиця Candidate зберігає дані кандидатів та результати AI-аналізу і пов'язана з Vacancy через зовнішній ключ vacancy_id. Поля: id, created_at, First_name, Last_name, Full_Name, Phone_number, email, Resume, Match_score (числова оцінка 0-100), Match_comments[], Skills_matched[], Skills_missing[], Skills_other[] (навички що не входять до вимог але присутні у резюме), Matched_skills_analyzer[] (масив об'єктів зі skill та score), ai_strengths, ai_concerns, ai_verdict, ai_recommendation, match_label, user_id та vacancy_id. Така структура дозволяє зберігати повний результат AI-аналізу безпосередньо у записі кандидата, що забезпечує миттєвий доступ до аналітики без повторних звернень до Gemini API.

Система реалізує наскрізну модель безпеки на обох архітектурних рівнях. На рівні WeWeb всі сторінки окрім входу та реєстрації налаштовані як приватні через механізм Page Access. При спробі неавторизованого доступу WeWeb автоматично перенаправляє на сторінку входу.

На рівні Xano всі ендпоінти окрім Authentication та google-oauth груп налаштовані як захищені. Xano верифікує JWT-токен у заголовку Authorization кожного вхідного запиту і відхиляє неавторизовані зі статусом 401 до виконання будь-яких кроків function stack. Ідентифікатор авторизованого користувача витягується з верифікованого токена і використовується для фільтрації даних: кожен рекрутер має доступ виключно до своїх вакансій і кандидатів. [23]

Додатковим рівнем захисту є Precondition-перевірки у критичних ендпоінтах. Наприклад, в ендпоінті candidate/analyze перевіряється що вакансія до якої додається кандидат належить поточному авторизованому користувачу (Vacancy1.user_id == id), що унеможливлює маніпуляції з чужими вакансіями навіть за наявності валідного токена. [25]

Клієнтська частина реалізована у WeWeb з темною темою як основною і світлою як альтернативною, між якими рекрутер перемикається одним тоглом у правому верхньому куті. Колірна схема побудована на поєднанні темного фону (#181818) з акцентним зеленим (#1DCF94) що використовується для активних елементів навігації, кнопок підтвердження та числових показників відповідності кандидатів.

Навігація реалізована через постійну ліву бокову панель з чотирма розділами: Dashboard, Vacancies, Candidates та Analytics. У нижній частині відображається ім'я та email поточного рекрутера з кнопкою виходу. Такий дизайн є стандартним для SaaS-застосунків і знайомий користувачам без необхідності навчання.

3.2. Реалізація бекенду на Xano: бізнес-логіка, безпека ендпоінтів та інтеграція Gemini API

Бекенд системи HireAI реалізований повністю засобами Xano без написання серверного коду. У цьому підрозділі детально розглядається реалізація ключових ендпоінтів, логіка їхнього function stack та механізми безпеки на рівні кожного запиту. [6]

Як зазначено у підрозділі 3.1, бекенд організований у чотири API-групи: Authentication, google-oauth, Vacancy та Candidate. Разом вони формують 19 ендпоінтів. Всі ендпоінти груп Vacancy та Candidate є захищеними: Xano автоматично перевіряє JWT-токен до виконання будь-якого кроку function stack.

Розробка кожного ендпоінту відбувається через візуальний редактор function stack де розробник послідовно додає кроки обробки запиту: отримання даних з бази, виклики зовнішніх API, трансформацію даних, умовні перевірки та формування відповіді. Такий підхід принципово відрізняється від традиційного написання серверного коду і дозволяє будувати складну логіку через інтуїтивний візуальний інтерфейс.

Один із ключових ендпоінтів системи - POST /vacancy/parse-pdf, що реалізує автоматичне створення вакансії на основі завантаженого PDF. Саме він демонструє центральну концепцію: рекрутер не заповнює форму вручну, а просто завантажує готовий PDF з описом вакансії, після чого Gemini автоматично витягує всі структуровані дані.

Ендпоінт приймає єдиний вхідний параметр - file_base64 (текстове поле з base64-кодованим вмістом PDF). Function stack складається з п'яти послідовних кроків (рис. 3.2).

Крок 1 - API Request To Gemini 2.5 Flash. У тілі запиту передається base64-вміст PDF і детальний системний промпт з інструкцією витягти структуровані дані вакансії у JSON: назву посади, компанію, локацію, зарплату, тип зайнятості, вимоги, обов'язки та навички.

Крок 2 - Create Variable: відповідь Gemini парситься з JSON-рядка у структурований об'єкт через функцію json_decode і зберігається у змінній resultParsed.

Крок 3 - Add Record In Vacancy: витягнуті дані записуються у таблицю Vacancy, при цьому поле user_id автоматично заповнюється ідентифікатором авторизованого користувача з JWT-токена.

Крок 4 - Get Record From user: завантажується поточний запис користувача для оновлення поля vacancy_id[].

Крок 5 - Edit Record In user: до списку vacancy_id[] додається новостворена вакансія.

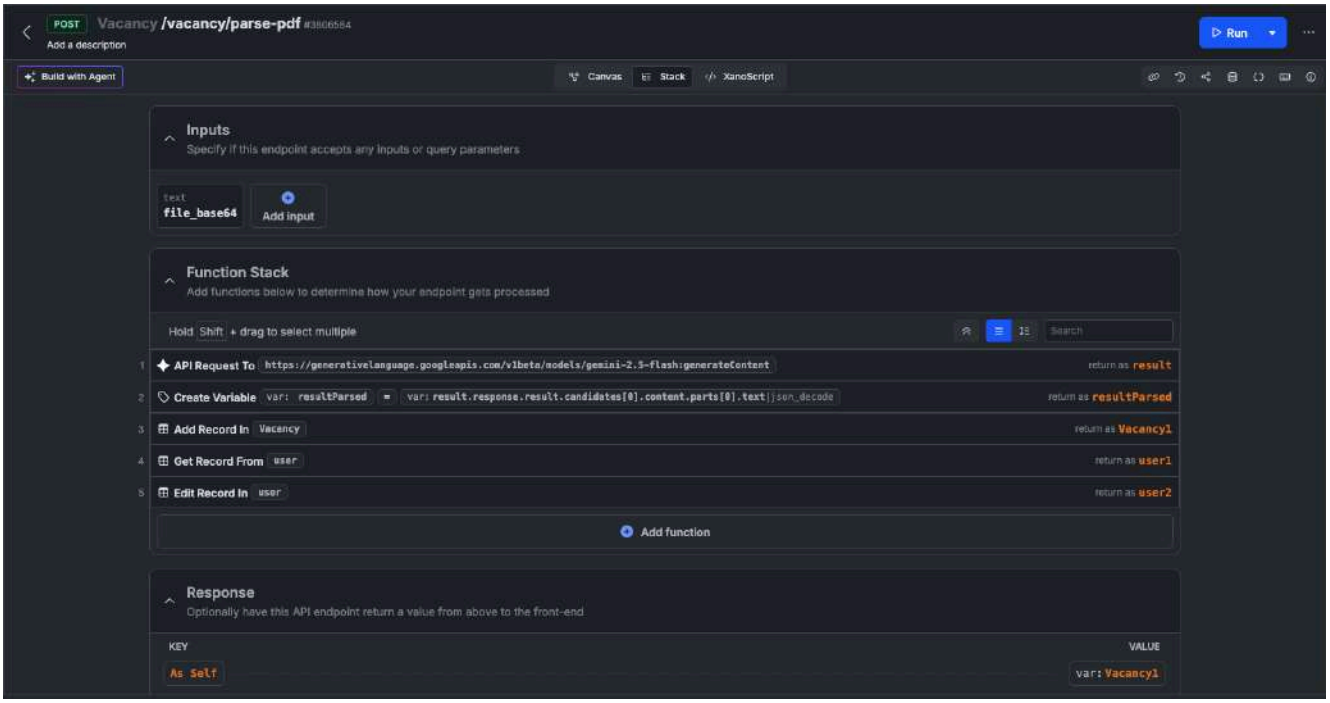

Рисунок 3.2 - Function stack ендпоінту POST /vacancy/parse-pdf у Xano

Такий підхід має принципову перевагу з точки зору досвіду користувача: рекрутер завантажує PDF що вже існує у компанії, і система сама створює структурований запис. Ніякого ручного введення даних.

Найважливіший і найскладніший ендпоінт системи - POST /candidate/analyze. Він реалізує повний цикл обробки нового кандидата: парсинг резюме, AI-аналіз відповідності вакансії та збереження результатів. Саме цей ендпоінт є серцем HireAI.

Ендпоінт приймає два вхідні параметри: file_base64 (base64-кодоване резюме у форматі PDF) та vacancy_id (ідентифікатор вакансії). Function stack складається з шести кроків (рис. 3.3).

Крок 1 - Get Record From Vacancy: завантажується повний запис вакансії за переданим vacancy_id, включаючи всі вимоги, обов'язки та навички.

Крок 2 - Precondition: перевіряється що вакансія належить поточному авторизованому рекрутеру (Vacancy1.user_id == id). Якщо умова не виконується, ендпоінт негайно повертає помилку авторизації без виконання жодних подальших кроків. Це унеможливлює ситуацію коли зловмисник з валідним токеном намагається додати кандидата до чужої вакансії.

Крок 3 - API Request To Gemini: це центральний крок ендпоінту. Gemini отримує одночасно base64-вміст PDF резюме і витягнуті поля вакансії з першого кроку та виконує комплексний аналіз: витягує персональні дані кандидата (ім'я, прізвище, email, телефон, навички) і одразу оцінює їх відповідність вимогам вакансії. Системний промпт інструктує модель повернути у JSON: Match_score (0-100), Match_comments, Skills_matched, Skills_missing, рекомендацію HIRE/MAYBE/REJECT, а також ai_strengths та ai_concerns.

Крок 4 - Create Variable: відповідь Gemini декодується з JSON-рядка у структурований об'єкт resultParsed через функцію json_decode.

Крок 5 - Add Record In Candidate: у таблицю Candidate записується повний профіль кандидата що включає персональні дані і результати AI-аналізу. Поле vacancy_id автоматично зв'язує кандидата з відповідною вакансією.

Крок 6 - Edit Record In Vacancy: оновлюється лічильник кандидатів та середній показник відповідності у записі вакансії що відображається на сторінці списку вакансій та у дашборді.

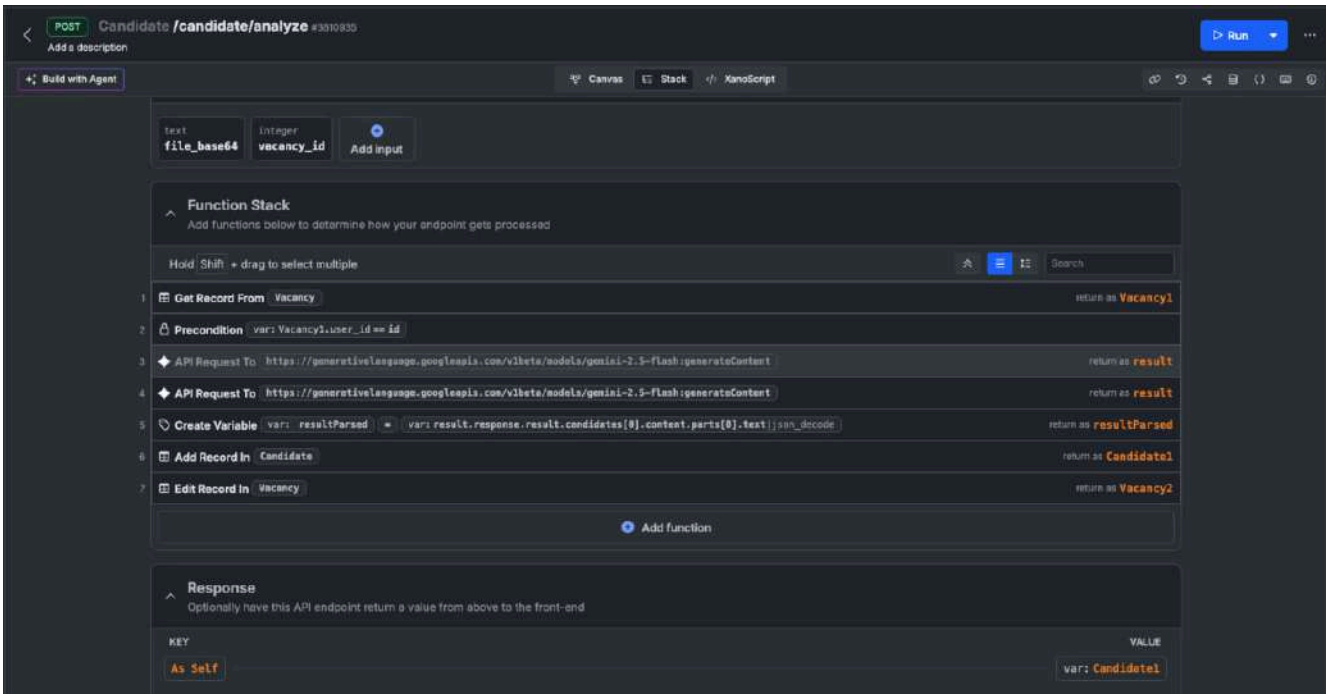

Рисунок 3.3 - Function stack ендпоінту POST /candidate/analyze у Xano

Єдиний виклик Gemini API що поєднує одночасно парсинг резюме і аналіз відповідності є свідомим рішенням: передача в одному запиті і файлу резюме, і даних вакансії дозволяє моделі одразу виконати контекстний аналіз без проміжних кроків. Це спрощує логіку ендпоінту і скорочує кількість API-викликів.

Безпека бекенду HireAI реалізована на трьох рівнях, що доповнюють один одного.

Перший - автоматична JWT-автентифікація. Всі ендпоінти груп Vacancy та Candidate налаштовані як захищені одним перемикачем у налаштуваннях. Xano перевіряє наявність і валідність Bearer-токена у заголовку Authorization кожного запиту. Термін дії токена обмежений, що унеможливлює використання перехоплених токенів тривалий час.

Другий - фільтрація даних за user_id. Запити до бази у всіх ендпоінтах автоматично обмежуються записами поточного авторизованого користувача. Ендпоінт GET vacancy повертає лише записи де user_id збігається з ідентифікатором з токена. Це архітектурно виключає витік даних між рекрутерами навіть при наявності валідного токена.

Третій - Precondition-перевірки для критичних операцій. В ендпоінтах що виконують запис або аналіз у контексті конкретної вакансії додатково перевіряється що вакансія належить поточному користувачу. Це захист від IDOR-атак (Insecure Direct Object Reference) коли зловмисник намагається маніпулювати чужими об'єктами передаючи їхні ідентифікатори у запиті.

API-ключ Gemini зберігається у розділі Keys & Variables платформи Xano як захищена змінна середовища. У кроках API Request використовується посилання на змінну, а не сам ключ у відкритому вигляді. Це забезпечує безпечне зберігання і спрощує ротацію ключа без будь-яких змін у логіці ендпоінтів.

Реалізація всієї описаної логіки без серверного коду є практичним підтвердженням ефективності low-code підходу. Візуальний function stack дає змогу одразу бачити послідовність виконання, легко переставляти і модифікувати кроки, тестувати кожен ендпоінт через вбудований Run-режим і читати логіку як документацію навіть без глибоких технічних знань. При цьому гнучкості цілком достатньо для складних сценаріїв: послідовні виклики зовнішніх API, Precondition-перевірки безпеки, трансформації даних.

3.3. Реалізація фронтенду на WeWeb: розмежування доступу та інтерфейс рекрутера

Фронтенд системи HireAI реалізований на WeWeb як повноцінний односторінковий веб-застосунок з п'ятьма основними сторінками та кількома модальними вікнами. У цьому підрозділі описується структура клієнтської частини, захист доступу, ключові сторінки та підхід до підключення даних з бекенду. [5]

Інтерфейс WeWeb будується на основі ієрархії компонентів що відображається у панелі Layout. Кожна сторінка складається з вкладених контейнерів і елементів: секцій, карток, форм, таблиць, графіків та окремих UI-елементів. Будь-який елемент інтерфейсу можна прив'язати до динамічних

даних через систему колекцій та змінних що оновлюються при отриманні відповіді від Xano.

Підключення до бекенду реалізоване через менеджер колекцій WeWeb у розділі Data & API. Для кожного ендпоінту Xano створена окрема колекція що налаштовується через URL, метод (GET або POST), параметри та заголовки. Заголовок авторизації Bearer передається автоматично для всіх захищених колекцій через глобальну змінну що зберігає JWT-токен поточної сесії. Розробнику не потрібно вручну додавати токен до кожного запиту - достатньо одноразово налаштувати глобальний заголовок на рівні колекції.

Захист доступу реалізований через механізм Page Access у налаштуваннях кожної сторінки. Кожна сторінка окрім Login та SignUp налаштована як приватна: при спробі неавторизованого доступу система автоматично перенаправляє на Login. Це налаштування активується декларативно через інтерфейс WeWeb без написання жодного рядка коду.

При успішному вході через ендпоінт Xano /auth/login або /auth/signup WeWeb отримує JWT-токен і зберігає його у глобальній змінній сесії. Цей токен автоматично використовується всіма захищеними колекціями для формування заголовка Authorization у кожному запиті. При виході токен очищається і WeWeb перенаправляє на Login. Таким чином двошаровий захист - Page Access на рівні WeWeb і JWT-верифікація на рівні Xano - забезпечує надійний контроль доступу без зайвої складності.

Система надає два екрани авторизації з єдиним візуальним стилем: темний фон, логотип HireAI з іконкою мозку та підпис “Smart recruiting powered by AI” у верхній частині, форма авторизації по центру.

Сторінка Login містить поля Email Address та Password, кнопку Sign In і кнопку входу через Google. Посилання “Forgot password?” розташоване поруч з полем паролю, у нижній частині - посилання на реєстрацію “Don't have an account yet? Create Account”.

Сторінка SignUp аналогічна за структурою але містить додаткове поле Name та кнопку Sign Up. Обидві сторінки підтримують автентифікацію через Google

OAuth що реалізована через відповідну групу ендпоінтів Xano та стандартний OAuth 2.0 flow з перенаправленням.

У WeWeb обидві сторінки є єдиними що налаштовані як публічні. Всі інші автоматично перенаправляють неавторизованих користувачів на Login (рис. 3.4).

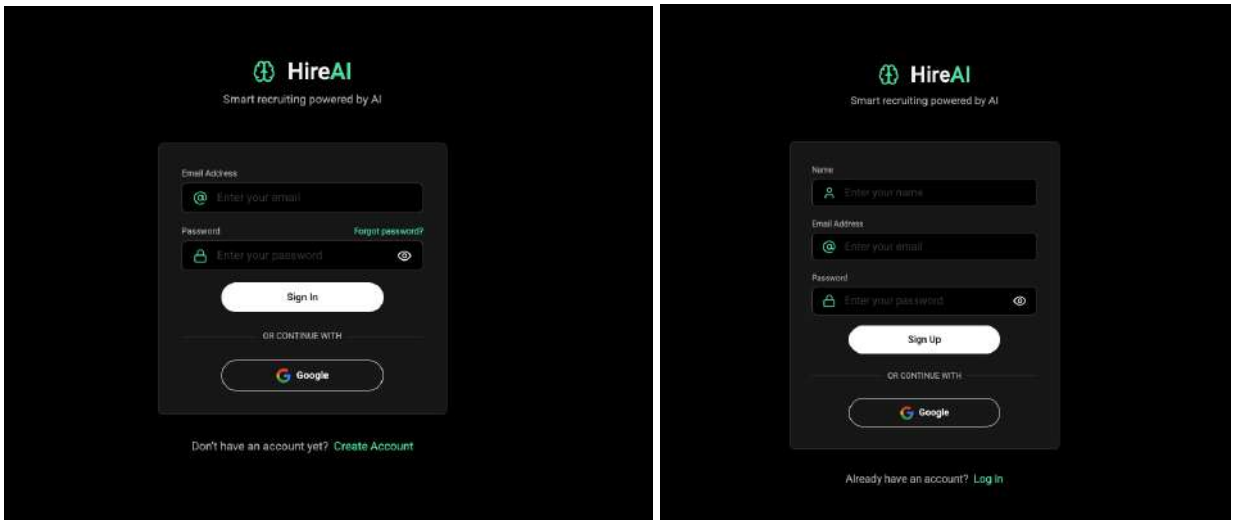

Рисунок 3.4 - Сторінки авторизації HireAI (Login та SignUp)

Dashboard - перша сторінка що бачить рекрутер після входу. Вона дає оперативний огляд поточного стану рекрутингових процесів і підтримує темну та світлу теми між якими перемикає тогл у правому верхньому куті.

У верхній частині розташовані чотири статистичні картки: Active Vacancies (кількість активних вакансій з динамікою за тиждень), Total Candidates (загальна кількість кандидатів), Analyzed Today (проаналізовано за поточний день) та Avg Match Score (середній показник відповідності у відсотках). Кожна картка показує не лише поточне значення а й динаміку: зростання зеленим зі стрілкою вгору, падіння червоним зі стрілкою вниз.

У середній частині два блоки. Ліворуч - “Recent Vacancies” з трьома останніми вакансіями та кнопкою Review для швидкого переходу. Праворуч - “Top AI Matches” з трьома кандидатами що мають найвищий Match Score, з аватаром на основі ініціалів та відсотком відповідності у зеленому бейджі.

У нижній частині блок “AI Analysis Timeline” з лінійним графіком активності аналізу у часі та стрічка останніх подій системи.

Всі дані Dashboard отримуються через колекції WeWeb що звертаються до захищених ендпоінтів Xano при завантаженні сторінки. Агрегація даних для статистичних карток виконується на рівні бекенду що відповідає архітектурному принципу розмежування відповідальностей (рис. 3.5).

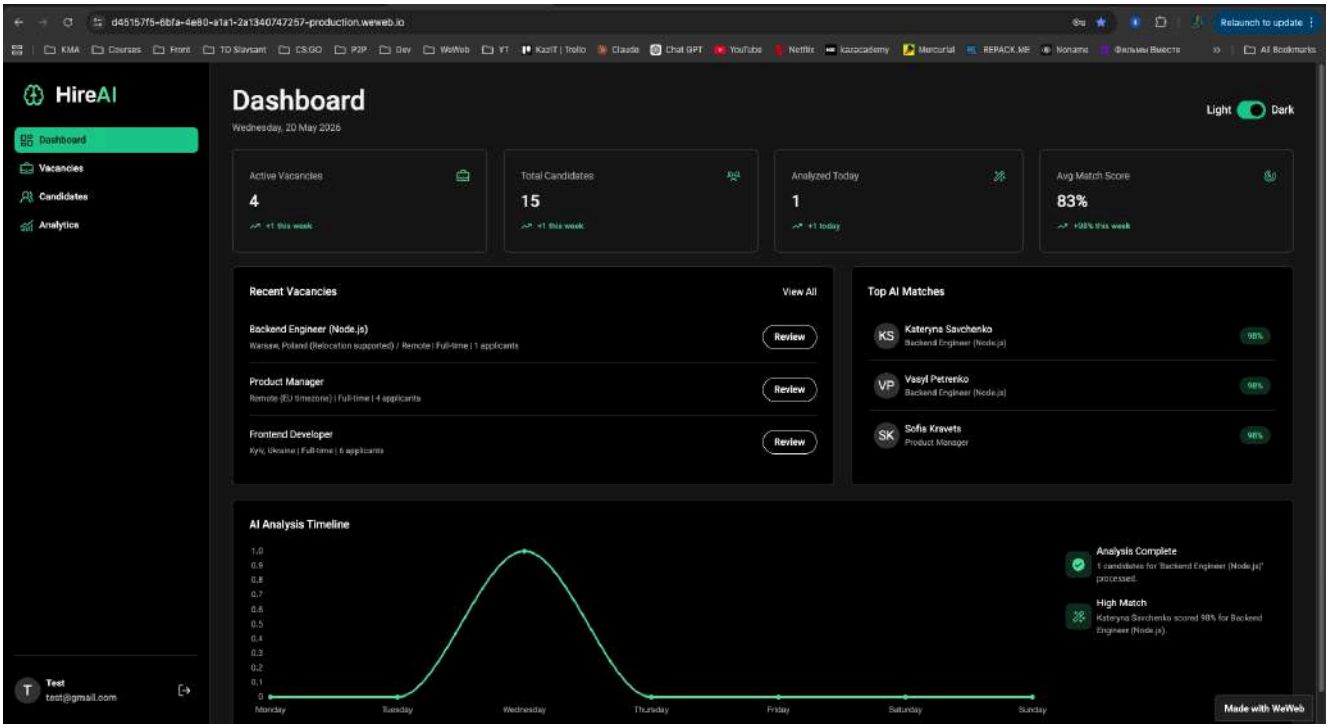

Рисунок 3.5 - Головна сторінка Dashboard системи HireAI (темна тема)

Сторінка Vacancies надає повний огляд вакансій рекрутера та інструменти для управління ними. У верхній частині - поле пошуку за назвою, фільтри за статусом і типом зайнятості та кнопка “+ Add Vacancy”.

Вакансії відображаються картками у сітці з двох колонок. Кожна картка містить назву вакансії, теги типу зайнятості та локації, бейдж статусу, кількість кандидатів, середній Match Score та дату публікації. Кнопки у нижній частині дозволяють редагувати або видалити вакансію (рис. 3.6).

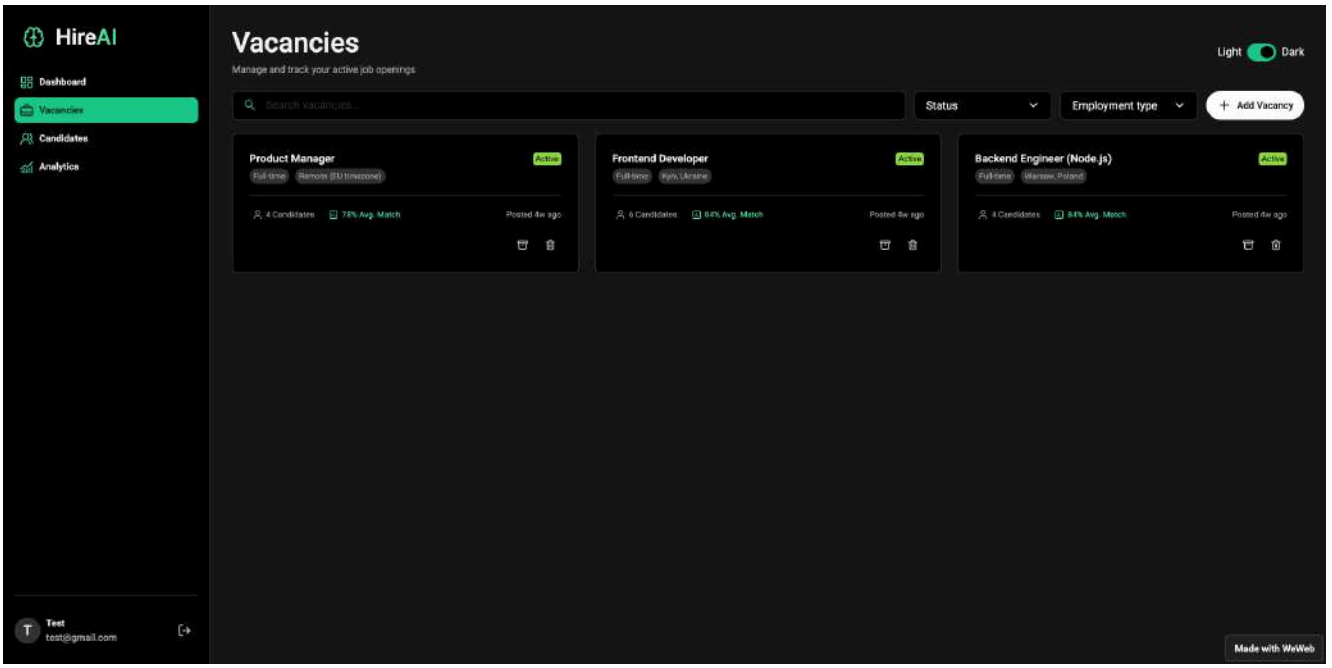

Рисунок 3.6 - Сторінка управління вакансіями системи HireAI

При натисканні “+ Add Vacancy” відкривається модальне вікно з зоною завантаження PDF. Рекрутер завантажує файл, натискає “Add”, WeWeb передає base64-вміст до ендпоінту /vacancy/parse-pdf, Gemini витягує всі структуровані дані і нова вакансія одразу з'являється у списку без жодного ручного введення.

При переході на деталі вакансії відображається повна інформація у редагованих полях: назва посади, компанія, локація, зарплата, тип зайнятості та статус. Нижче - секції Job Details зі списками вимог, обов'язків та навичок у вигляді тегів що можна видаляти. Праворуч бокова панель “Candidates” з переліком прив'язаних кандидатів та їхніми показниками відповідності. Кнопки “Delete” та “Update” у верхньому куті дозволяють видалити вакансію або зберегти зміни (рис. 3.7).

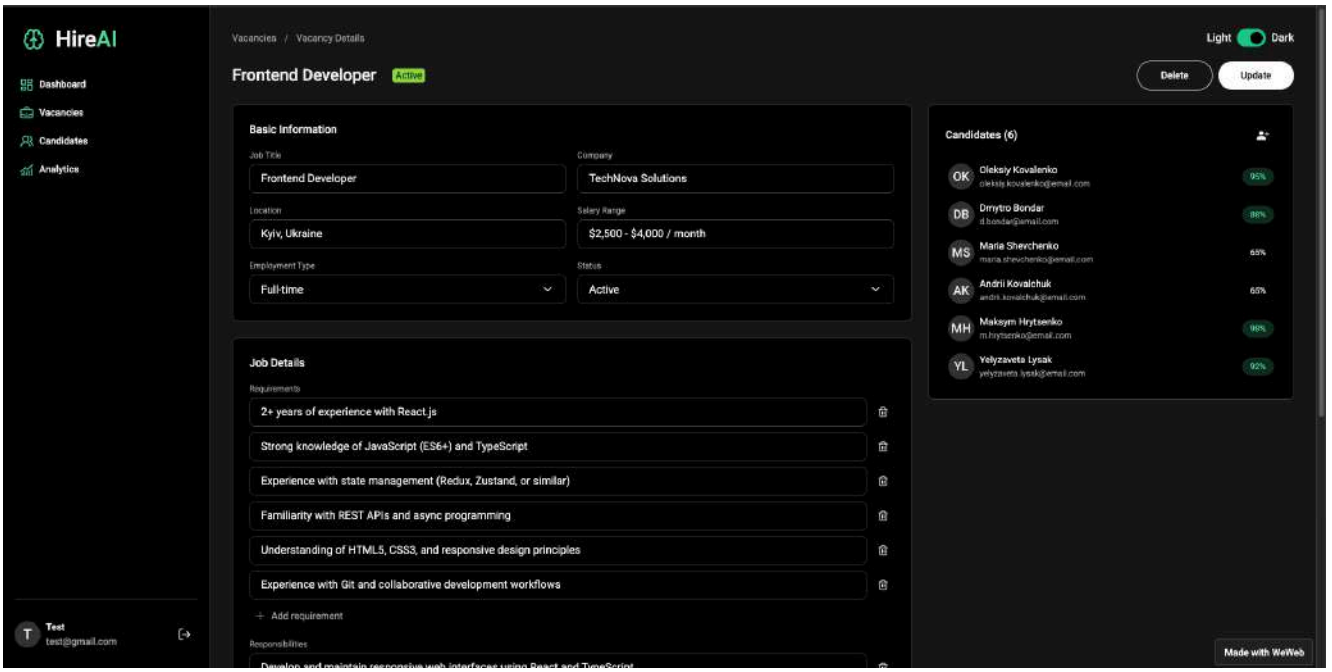

Рисунок 3.7 - Сторінка деталей вакансії з переліком кандидатів

Сторінка Candidates дає зведений огляд всіх кандидатів системи незалежно від вакансії. Дані відображаються таблицею з колонками: Name, Phone, Email, Score, Vacancy та Result. Колонки підтримують сортування. Score відображається числом від 0 до 100, Result - статусом HIRE, MAYBE або REJECT що генерує Gemini. У нижній частині пагінація із загальною кількістю записів (рис. 3.8).

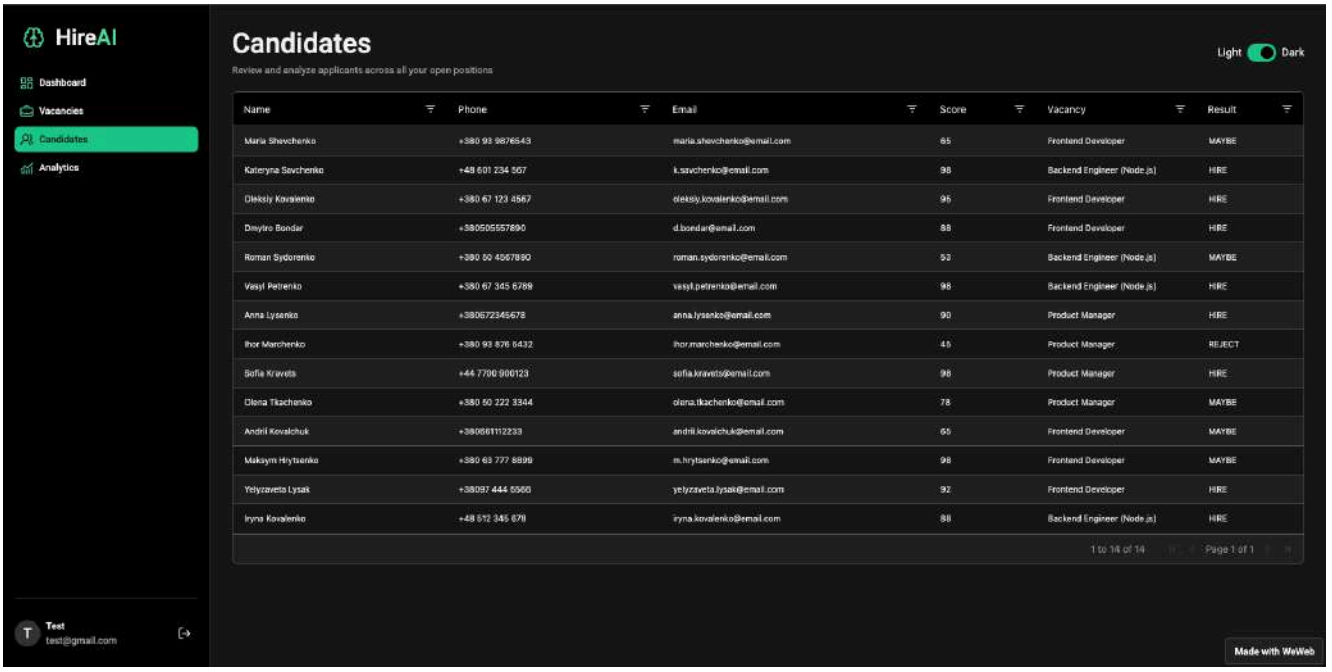

Рисунок 3.8 - Сторінка списку кандидатів системи HireAI

При натисканні на рядок відкривається сторінка Candidate Details. Ліворуч - персональні дані кандидата: ім'я, вакансія, дата завантаження резюме, email, телефон та кнопка завантаження PDF. Нижче блок “Extracted Skills” з трьома категоріями: MATCHED (збігаються з вимогами, виділені зеленим), MISSING (яких бракує, помаранчевим) та OTHER (додаткові навички кандидата).

Праворуч кругова діаграма з числовим Match Score та кольоровим кодуванням, три ключові тези від Gemini щодо сильних сторін, блок “Skill Proficiency vs Requirement” з bar-чартом що порівнює рівень навичок кандидата з вимогами по кожній технології. У нижній правій частині блок “AI Hiring Recommendation” з бейджем рекомендації, розгорнутими секціями Strength та Concerns і фінальним Verdict від Gemini (рис. 3.9).

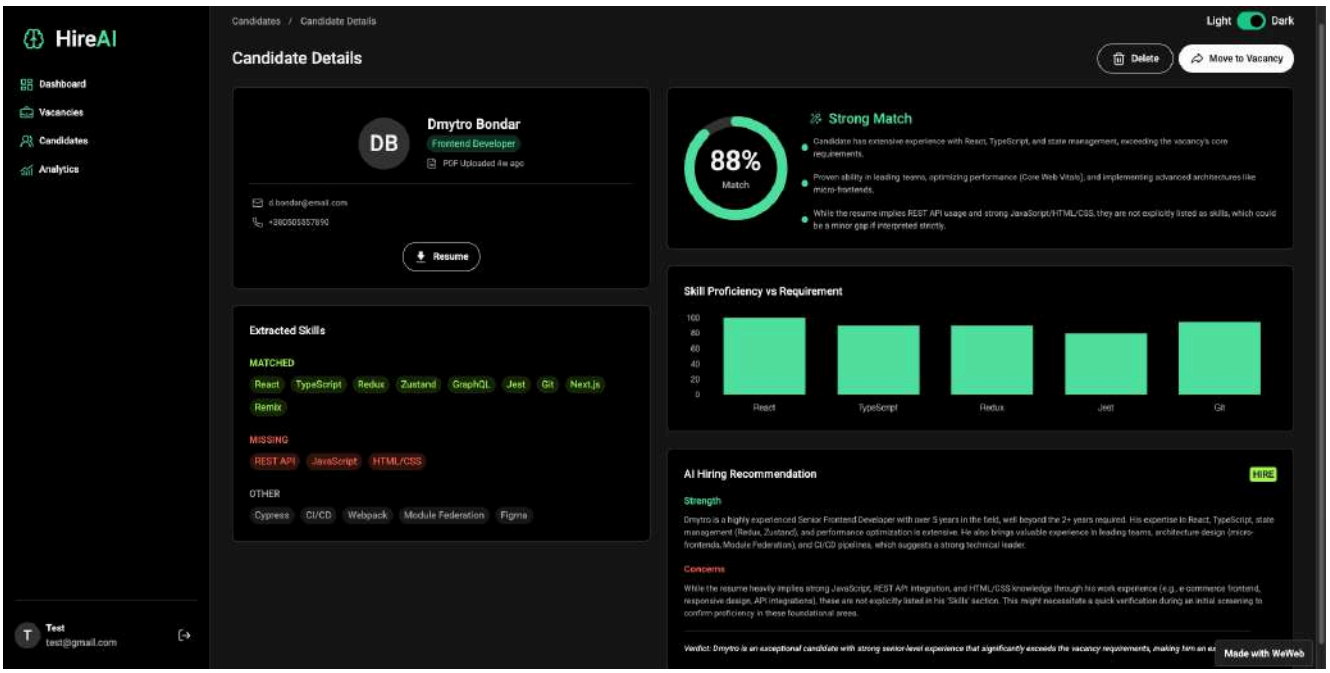

Рисунок 3.9 - Профіль кандидата з AI-аналізом відповідності

Всі динамічні дані прив'язані до компонентів через систему колекцій. Колекція - це іменоване з'єднання з конкретним ендпоінтом Xano що може бути прив'язане до будь-якого елемента через механізм data binding. Список вакансій наприклад прив'язаний до колекції що звертається до GET /vacancy при завантаженні сторінки. Кожна картка відображає дані конкретного елемента колекції.

Для операцій запису - створення, оновлення, видалення - використовуються обробники подій що прив'язуються до кнопок та інтерактивних елементів. При натисканні “Add” у модальному вікні WeWeb виконує JavaScript-функцію конвертації файлу у base64 і викликає POST-запит до /vacancy/parse-pdf. Після успішної відповіді колекція Vacancies оновлюється автоматично і нова вакансія з'являється у списку без перезавантаження сторінки.

Така архітектура взаємодії з даними є характерною для SPA-застосунків і забезпечує плавний користувацький досвід без переходів між сторінками та повних перезавантажень.

3.4. Тестування та оцінка ефективності системи

Завершальний етап розробки HireAI - оцінка функціональної коректності системи, якості AI-аналізу та практичної ефективності low-code підходу. У цьому підрозділі описуються підходи до тестування, результати оцінювання якості аналізу кандидатів та узагальнені висновки щодо обраного стеку. [24]

Тестування проводилось на чотирьох рівнях що відповідають архітектурним шарам застосунку. [24]

На рівні бекенду кожен ендпоінт Xano тестувався через вбудований Run-режим платформи. Він дозволяє виконати запит безпосередньо у редакторі function stack із заданими вхідними параметрами і переглянути покроковий результат кожного кроку. Це одна з ключових практичних переваг Xano: розробник бачить проміжні результати після кожного кроку що суттєво спрощує пошук і виправлення помилок. Для ендпоінтів що потребують JWT-токена тестування виконувалось з попередньо отриманим токеном авторизації. Окрема увага приділялась Precondition-перевіркам: верифікувалось що спроба передати vacancy_id чужої вакансії коректно відхиляється зі статусом помилки авторизації.

На рівні інтеграції з Gemini API тестувались різні типи PDF-документів: вакансії різної структури та обсягу, резюме різних форматів (хронологічні, функціональні, комбіновані), документи різними мовами. Перевірялась

коректність парсингу, якість витягування навичок та стабільність формату JSON-відповіді при варіюванні вхідних даних.

На рівні фронтенду тестувалась коректність відображення у всіх станах інтерфейсу: порожній стан (відсутність вакансій або кандидатів), завантаження даних, успішне отримання відповіді та обробка помилок. Окремо перевірялась логіка захисту сторінок: поведінка системи при прямому переході за URL захищених сторінок без авторизації та після закінчення сесії.

На рівні наскрізного тестування виконувався повний сценарій: реєстрація нового рекрутера, створення вакансії через завантаження PDF, додавання кандидатів через завантаження резюме, перегляд результатів аналізу та навігація між розділами. Сценарій відтворювався у темній і світлій темах інтерфейсу.

Якість генерованих Gemini висновків безпосередньо визначає практичну цінність системи. Для оцінювання використовувались реальні сценарії з трьома вакансіями різного профілю: Product Manager, Frontend Developer та Backend Engineer (Node.js). До кожної вакансії додавались від чотирьох до шести кандидатів з різним рівнем відповідності.

Результати підтверджують здатність системи коректно диференціювати кандидатів. Для вакансії Backend Engineer кандидат Kateryna Savchenko з шістьма роками досвіду у Node.js та TypeScript отримала оцінку 98% і рекомендацію HIRE. Водночас кандидат Ihor Marchenko для вакансії Product Manager отримав 45% і рекомендацію REJECT: система точно виявила суттєву невідповідність між його досвідом і вимогами позиції.

Блок “Extracted Skills” демонструє здатність Gemini не просто витягувати навички з резюме, а класифікувати їх за трьома категоріями відносно конкретної вакансії: MATCHED (прямо відповідають вимогам), MISSING (відсутні у резюме) та OTHER (додаткові навички що можуть бути корисними). Така класифікація дає рекрутеру структурований огляд профілю кандидата і суттєво спрощує прийняття рішення.

Блок “AI Hiring Recommendation” містить три компоненти: Strength (сильні сторони стосовно конкретної вакансії), Concerns (потенційні ризики або

прогалини) та підсумковий Verdict. Якість коментарів є достатньо високою для практичного використання: вони конкретні, посилаються на реальні деталі резюме та вимоги, а не є загальними шаблонними фразами.

Avg Match Score по всіх вакансіях склав 82%. Розподіл рекомендацій у тестовій вибірці: HIRE - 7 кандидатів, MAYBE - 5 кандидатів, REJECT - 2 кандидати. Це відповідає реалістичному розподілу у реальному рекрутинговому процесі.

Центральна гіпотеза дослідження полягає у тому що low-code підхід дозволяє суттєво скоротити час розробки AI-орієнтованих систем без критичного зниження якості. Практичні результати HireAI є переконливим підтвердженням цього.

Повнофункціональна система з автентифікацією, управлінням вакансіями та кандидатами, AI-аналізом, дашбордом та аналітикою розроблена одним розробником за 1-2 місяці. При цьому обидві платформи освоювались з нуля безпосередньо у процесі розробки. Для порівнянного за функціональністю застосунку на традиційному стеку - React на фронтенді та Node.js з Express на бекенді - аналогічний результат потребував би значно більшого часу навіть від досвідченого розробника. [1, 2]

З точки зору функціональності реалізована система є повноцінним робочим продуктом, а не прототипом. Вона підтримує автентифікацію через email/пароль та Google OAuth, надійний захист через JWT і Precondition-перевірки, реальну інтеграцію з Gemini API, повний CRUD для вакансій і кандидатів, адаптивний інтерфейс з темною та світлою темою.

Водночас low-code підхід має і свої обмеження що проявились у процесі розробки. По-перше, залежність від стабільності платформ: оновлення WeWeb або Xano можуть потенційно порушити частину логіки що потребує моніторингу. По-друге, обмеження при нестандартних сценаріях: деякі специфічні задачі потребують обхідних рішень або кастомного JavaScript у WeWeb чи XanoScript у Xano. По-третє, питання масштабованості при значному зростанні навантаження

потребує переходу на вищі тарифні плани або у крайньому випадку міграції частини логіки на традиційну інфраструктуру.

Попри ці обмеження для цільової аудиторії HireAI - малого та середнього бізнесу - обраний підхід є оптимальним балансом між швидкістю розробки, функціональністю та вартістю.

Висновки до розділу 3

У третьому розділі описано повний цикл проєктування та реалізації системи HireAI засобами low-code платформ WeWeb і Xano з інтеграцією Gemini API. Архітектура побудована на принципі чіткого розмежування фронтенду і бекенду з комунікацією через REST API та наскрізною моделлю безпеки на основі JWT-автентифікації і Precondition-перевірок.

Бекенд на Xano включає 19 захищених ендпоінтів у чотирьох функціональних групах, реалізованих без серверного коду. Ключовим технічним досягненням є ендпоінт /candidate/analyze що поєднує в одному API-виклику до Gemini парсинг резюме і глибокий аналіз відповідності кандидата вимогам вакансії з генерацією структурованого звіту.

Фронтенд на WeWeb реалізує повноцінний SPA з п'ятьма основними сторінками, захистом доступу через Page Access та динамічним зв'язуванням даних з бекендом. Тестування підтвердило функціональну коректність всіх компонентів, якість AI-аналізу та практичну ефективність low-code підходу для розробки AI-орієнтованих систем.

ВИСНОВКИ

Магістерська робота присвячена розробці AI-асистента для автоматизованого аналізу кандидатів на базі low-code підходів та оцінці практичної ефективності цього підходу для створення повноцінних AI-орієнтованих програмних систем. За результатами проведеного дослідження сформульовано такі висновки.

Аналіз концептуальних засад low-code та no-code розробки засвідчив що ця парадигма є фундаментальним зрушенням у підходах до створення програмного забезпечення, а не просто черговим технологічним трендом. Ринок low-code платформ зростає із середньорічним темпом близько 25% і за прогнозами Gartner до 2026 року перевищить 26 мільярдів доларів. Організації що впровадили low-code підходи скорочують час розробки у 5-10 разів порівняно з традиційними методами, що підтверджується практичними результатами цього дослідження. [4]

Дослідження застосування штучного інтелекту в рекрутингу виявило широкий спектр підходів: від ключових слів у класичних ATS-системах до предиктивної аналітики та LLM-базованих рішень нового покоління. Огляд існуючих рішень встановив суттєву незайняту нішу: відсутність доступних, кастомізованих та прозорих інструментів аналізу кандидатів на основі великих мовних моделей, орієнтованих на малий та середній бізнес. Систему HireAI розроблено саме для заповнення цієї ніші.

Сформована система критеріїв оцінювання low-code платформ охоплює три укрупнені групи: функціональність, швидкість розробки та безпеку. Вона забезпечила методологічно обґрунтований підхід до порівняльного аналізу альтернативних технологій і може бути застосована іншими дослідниками та практиками при виборі low-code стеку для реалізації AI-орієнтованих проєктів.

Порівняльний аналіз платформ фронтенду засвідчив перевагу WeWeb над FlutterFlow у контексті розробки веб-орієнтованого AI-асистента для рекрутерів. WeWeb забезпечує природнішу інтеграцію із зовнішнім бекендом через REST API, значно пологішу криву навчання для розробника з веб-бекґраундом та вбудований

механізм захисту сторінок що активується без написання коду. Порівняльний аналіз бекенд-платформ підтвердив перевагу Xano над Firebase: реляційна модель даних PostgreSQL природно відповідає структурі сутностей системи, а вбудований захист ендпоінтів одним перемикачем є принциповою перевагою з точки зору безпеки та швидкості розробки.

Вибір Gemini API обґрунтований сукупністю практичних і технічних факторів: негайна доступність ключа через Google AI Studio, підтримка стандартних HTTP-запитів без додаткових SDK, велике контекстне вікно та конкурентне ціноутворення з безкоштовним рівнем. Архітектурне рішення викликати Gemini API виключно з бекенду Xano забезпечує захист ключа, централізацію логіки промптингу та контроль витрат.

Спроєктована та реалізована система HireAI є повнофункціональним AI-асистентом для автоматизованого аналізу кандидатів побудованим на принципі чіткого розмежування фронтенду і бекенду. Бекенд на Xano включає 19 захищених ендпоінтів у чотирьох функціональних групах реалізованих без серверного коду. Фронтенд на WeWeb являє собою повноцінний SPA з п'ятьма основними сторінками, захистом через Page Access та динамічним зв'язуванням даних. Наскрізна модель безпеки на основі JWT-автентифікації та Precondition-перевірок надійно захищає персональні дані кандидатів на обох архітектурних рівнях.

Ключовим технічним досягненням є ендпоінт /candidate/analyze що в одному API-виклику до Gemini поєднує парсинг PDF-резюме, витягування персональних даних і навичок кандидата та глибокий аналіз відповідності вимогам вакансії. Результатом є структурований звіт що включає числовий Match Score, класифікацію навичок за категоріями MATCHED, MISSING та OTHER, а також розгорнуті блоки Strength, Concerns та підсумковий Verdict з рекомендацією HIRE, MAYBE або REJECT.

Тестування на реальних сценаріях рекрутингу підтвердило функціональну коректність всіх компонентів та достатній рівень якості AI-аналізу для практичного використання. Gemini коректно диференціює кандидатів: релевантні отримують оцінки 90-98% і рекомендацію HIRE з конкретним обґрунтуванням,

невідповідні - низькі оцінки і REJECT. Середній Match Score по тестовій вибірці склав 82%.

Практична валідація центральної гіпотези дослідження підтверджена головним результатом: повнофункціональна система HireAI розроблена одним розробником за 1-2 місяці з нуля, включаючи освоєння обох платформ. Це переконливо свідчить що low-code підхід із WeWeb та Xano забезпечує якісно вищу швидкість розробки AI-орієнтованих систем порівняно з традиційними методами, зберігаючи при цьому достатній рівень функціональності, безпеки та масштабованості для малого та середнього бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Forrester Research. Low-Code Development Platforms Forecast, 2021–2026. Cambridge: Forrester Research, 2021. 34 p.
2. Gartner. Magic Quadrant for Enterprise Low-Code Application Platforms. Stamford: Gartner Inc., 2023. 48 p.
3. Richardson C., Rymer J. New Development Platforms Emerge For Customer-Facing Applications. Cambridge: Forrester Research, 2014. 17 p.
4. Gartner. Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 25% in 2024. URL: https://www.gartner.com/en/newsroom/press-releases/2024-02-29-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-25-percent-in-2024 (дата звернення: 15.03.2026).
5. WeWeb Documentation. WeWeb Official Docs. URL: https://docs.weweb.io (дата звернення: 20.02.2026).
6. Xano Documentation. Xano Official Docs. URL: https://docs.xano.com (дата звернення: 20.02.2026).
7. Google AI. Gemini API Documentation. URL: https://ai.google.dev/gemini-api/docs (дата звернення: 10.03.2026).
8. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008.
9. Brown T. et al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877–1901.
10. LinkedIn. Global Talent Trends 2023: The New World of Work. LinkedIn Talent Solutions. 2023. 52 p. [10]
11. Ladders Inc. Eye-Tracking Study: Recruiters Look at Resumes for 7.4 Seconds. New York: TheLadders, 2018. 12 p.
12. European Parliament. Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence

(Artificial Intelligence Act). Official Journal of the European Union. 2024. L 1689. 144 p.

13. European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data (GDPR). Official Journal of the European Union. 2016. L 119. P. 1–88.
14. Dastin J. Amazon scraps secret AI recruiting tool that showed bias against women. Reuters. 2018. URL: [14] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G (дата звернення: 05.03.2026).
15. Eightfold AI. The State of Talent Intelligence 2023. Santa Clara: Eightfold AI, 2023. 38 p.
16. Pymetrics. Science Behind Pymetrics: Neuroscience Games and AI. URL: https://www.pymetrics.ai/science (дата звернення: 12.03.2026).
17. Anthropic. The Claude Model Family. URL: https://www.anthropic.com/claude (дата звернення: 01.04.2026).
18. OpenAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774. 2023. 100 p.
19. Google DeepMind. Gemini: A Family of Highly Capable Multimodal Models. arXiv preprint arXiv:2312.11805. 2023. 62 p.
20. Watt A., Eng A. Database Design. 2nd ed. Victoria: BCcampus, 2014. 297 p.
21. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: PhD dissertation. University of California, Irvine, 2000. 162 p.
22. OWASP Foundation. OWASP Top Ten 2021. URL: https://owasp.org/www-project-top-ten (дата звернення: 18.03.2026).
23. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. 30 p.
24. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2015. 816 p.
25. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Boston: Addison-Wesley, 2012. 624 p.

26. FlutterFlow Documentation. FlutterFlow Official Docs. URL: https://docs.flutterflow.io (дата звернення: 15.02.2026).
27. Firebase Documentation. Firebase Official Docs. URL: https://firebase.google.com/docs (дата звернення: 15.02.2026).
28. Supabase Documentation. Supabase Official Docs. URL: https://supabase.com/docs (дата звернення: 10.02.2026).
29. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16 (дата звернення: 05.03.2026).
30. Greenhouse Software. The Definitive Guide to Structured Hiring. New York: Greenhouse Software, 2022. 44 p.
31. McKinsey Global Institute. The Economic Potential of Generative AI: The Next Productivity Frontier. McKinsey & Company, 2023. 68 p.
32. HireVue. AI in Hiring: What You Need to Know. URL: https://www.hirevue.com/blog/hiring/ai-in-hiring (дата звернення: 20.03.2026).
33. ResumeGo. How Long Do Recruiters Spend on a Resume? 2024 Recruiter Survey. URL: https://www.resumego.net/research/recruiter-time (дата звернення: 25.03.2026).
34. Vue.js Documentation. Vue.js Official Guide. URL: https://vuejs.org/guide/introduction (дата звернення: 01.03.2026).
35. Дорошенко А. Ю., Жежерун О. П. Технології розробки програмного забезпечення. Київ: Наукова думка, 2018. 312 с.