Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«__»_____20__р

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня «бакалавр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»

на тему**: «Інтелектуальна система підбору венчурних інвесторів для стартапів на основі семантичного пошуку»**

Виконав:
здобувач освіти 4 курсу,
групи ІСТ-41
Чупринський Максим Вікторович
(прізвище, ім'я, по – батькові)

Керівник:
д.е.н., професор Грицюк П. М.
(науковий ступінь, вчене звання прізвище та ініціали)

Рецензент:
к.т.н., доцент Джоші О. І.
(науковий ступінь, вчене звання прізвище та ініціали)

Рівне – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики
Освітньо-кваліфікаційний рівень – бакалавр
Освітньо-професійна програма
«Інформаційні системи і технології»
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«__»_____20__р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Чупринський Максим Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи: *Інтелектуальна система підбору венчурних інвесторів для стартапів на основі семантичного пошуку*

керівник роботи: д. е. н., проф. Петро Миколайович Грицюк
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від *25 квітня 2026 р.*

2.Термін здачі студентом закінченої роботи: *08 червня 2026 р.*

3. Вихідні дані до роботи: *дані про венчурні фонди, портфельні компанії та програмну реалізацію системи Fund Matcher*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *аналіз предметної області, проєктування, реалізація, тестування та визначення напрямів розвитку системи Fund Matcher*

5. Перелік графічного матеріалу:
1. *Схема стадій венчурного фінансування*
2. *Схема взаємодії стартапу, системи Fund Matcher та фонду*
3. *Порівняння конкурентних платформ*
4. *ER-діаграма бази даних Fund Matcher*

5. *Загальна архітектурна схема системи*
6. *Скріншоти клієнтського інтерфейсу*
7. *Фрагменти програмної реалізації*
8. *Блок-схема алгоритму пошуку*

6. Консультація розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Завдання видав (підпис, дата)	Завдання прийняв (підпис, дата)
Всі розділи	*Грицюк П.М.*		

7. Дата видачі завдання: 16 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Примітка
1.	*Постановка мети та визначення актуальності теми*	*16.11.25–23.11.25*	
2.	*Аналіз предметної області венчурного фінансування*	*24.11.25–07.12.25*	
3.	*Аналіз існуючих рішень для пошуку інвесторів*	*08.12.25–21.12.25*	
4.	*Формування вимог до інформаційної системи Fund Matcher*	*22.12.25–11.01.26*	
5.	*Проєктування структури бази даних*	*12.01.26–25.01.26*	
6.	*Проєктування архітектури серверної та клієнтської частин*	*26.01.26–08.02.26*	
7.	*Реалізація бази даних, моделей та міграцій*	*09.02.26–22.02.26*	
8.	*Реалізація серверної частини та API*	*23.02.26–08.03.26*	
9.	*Реалізація клієнтського інтерфейсу та адміністративної частини*	*09.03.26–15.03.26*	
10.	*Реалізація алгоритму семантичного пошуку та ранжування фондів*	*16.03.26–20.03.26*	
11.	*Тестування, виправлення помилок та оптимізація системи*	*21.03.26–31.03.26*	
12.	*Написання пояснювальної записки*	*01.04.26–05.05.26*	
13.	*Підготовка презентації та демонстраційних матеріалів*	*06.05.26–25.05.26*	
14.	*Фінальне редагування роботи та підготовка документів до захисту*	*26.05.26–07.06.26*	
15.	*Представлення роботи та інших документів до захисту*	*09.06.26*	

Студент _____
(підпис) (прізвище, та ініціали)

Керівник роботи _____
(підпис) (прізвище, та ініціали)

ЗАЯВА

щодо самостійності виконання випускної кваліфікаційної роботи

Я, _____(ПІБ), студент(ка) _____ курсу ___ групи ___ ННІ _____

ЗАЯВЛЯЮ:

моя випускна кваліфікаційна робота на тему «_____ _____», яка надається в екзаменаційну комісію із захисту кваліфікаційних робіт зі спеціальності «_____» для захисту, виконана самостійно і не містить плагіату.

Всі запозичення з друкованих та електронних джерел, у тому числі із захищених раніше випускових кваліфікаційних робіт мають відповідні посилання.

Я не використовував(ла) шахрайські методи маніпуляції з текстом (заміна букв, інтервали, мікропробіли, білі знаки, парафрази та ін.).

Інструменти штучного інтелекту використовував(ла) без порушення академічної доброчесності.

Я ознайомлений(а) з чинним Порядком перевірки навчальних, кваліфікаційних, навчально-методичних та наукових робіт на наявність ознак академічного плагіату в НУВГП та Положенням про академічну доброчесність в НУВГП, за яким виявлення плагіату є підставою для відмови в допуску моєї роботи до захисту та застосування відповідних санкцій (академічної відповідальності).

Метадані

ДОКУМЕНТ

Заголовок

2026 Чупринський М В Інтелектуальна система підбору венчурних інвесторів для стартапів на основі семантичного пошуку.docx

Автор	Науковий керівник / Експерт	ІД документу
Чупринський Максим Вікторович	**Чупринський Максим Вікторович**	**334263699**

ОРГАНІЗАЦІЯ

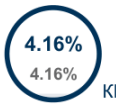
Назва організації	підрозділ
National University of Water and Environmental Engineering	**National University of Water and Environmental Engineering**

ЗВІТ

Дата звіту	Дата редагування
6/11/2026	**---**

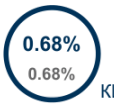
Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

7160
Кількість слів



КЦ

55951
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	9

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ep3.nuwm.edu.ua/30214/1/04-05-84%D0%9C.pdf	67 (0.94 %)

АКТ

перевірки випускної кваліфікаційної роботи

автора _____

на рівень запозичень та можливих маніпуляцій з текстом

Відповідно до даних системи StrikePlagiarism файл «_____» містить: ____ % коефіцієнта подібності; ____% коефіцієнта цитованості; ____ замінених букв; ____ інтервалів; ____ мікропробілів; ____ білих знаків; ____ % ймовірність використання ШІ.

За результатами перевірки засвідчую, що:

автор кваліфікаційної роботи _____:

- самостійно виконав(ла) кваліфікаційну роботу;
- коректно посилався(лась) на використані інформаційні джерела;
- в роботі відсутні ознаки академічної недоброчесності.

Керівник кваліфікаційної роботи *підпис* _____

(ПІП)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи на тему «Інтелектуальна система підбору венчурних інвесторів для стартапів на основі семантичного пошуку» для здобуття освітнього ступеня «Бакалавр», за освітньо-професійною програмою «Інформаційні системи і технології» за спеціальністю 126 «Інформаційні системи та технології» написана на 42 сторінках та містить 14 рисунків, 5 таблиць, 22 джерел та додатки.

Метою роботи є проєктування та розробка інформаційної системи для підбору венчурних інвесторів для стартапів на основі поєднання формальних критеріїв фільтрації та семантичного пошуку.

Об'єктом дослідження є процес пошуку, оцінювання та ранжування венчурних фондів для стартапів.

Предметом дослідження є методи побудови інформаційної системи, що поєднує реляційну базу даних, векторний пошук, API-сервіси та клієнтський інтерфейс.

У процесі виконання роботи було використано системний аналіз, методи проєктування баз даних, клієнт-серверну архітектуру, REST API, векторне представлення текстів, семантичний пошук і тестування програмного забезпечення. Програмна реалізація виконана з використанням Python, FastAPI, PostgreSQL, pgvector, SQLAlchemy, Alembic, Docker, OpenAI API, React, TypeScript, GitHub та Supabase.

Практичне значення одержаних результатів полягає у створенні прототипу системи Fund Matcher, яка може використовуватися для автоматизації первинного пошуку релевантних венчурних фондів. Система дозволяє враховувати не лише стадію фінансування, галузь, географію та розмір інвестиційного чека, а й фактичні інвестиційні інтереси фондів через аналіз їх портфельних компаній.

Ключові слова: венчурний капітал, стартап, венчурний фонд, семантичний пошук, векторний пошук, embeddings, pgvector, FastAPI, PostgreSQL, React, інформаційна система.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API	–	Інтерфейс прикладного програмування
REST	–	Передача репрезентативного стану
AI	–	Штучний інтелект
ML	–	Машинне навчання
NLP	–	Обробка природної мови
SQL	–	Мова структурованих запитів
ORM	–	Об’єктно-реляційне відображення
JWT	–	JSON Web Token, токен автентифікації
CRUD	–	Створення, читання, оновлення та видалення даних
UI	–	Користувацький інтерфейс
UX	–	Користувацький досвід
CI/CD	–	Безперервна інтеграція та безперервне розгортання
MVP	–	Мінімально життєздатний продукт
VC	–	Венчурний капітал
БД	–	База даних
ІС	–	Інформаційна система
ПЗ	–	Програмне забезпечення
СКБД	–	Система керування базами даних
API-ключ	–	Унікальний ключ доступу до зовнішнього API
similarity score	–	Показник семантичної схожості
rule score	–	Оцінка відповідності за формальними критеріями
combined score	–	Загальна оцінка релевантності

ЗМІСТ

ВСТУП

У сучасній цифровій економіці доступ до якісного фінансування є одним із ключових чинників розвитку технологічних стартапів. Молоді компанії часто мають обмежені ресурси, невеликі команди та високу залежність від зовнішнього капіталу, тому правильний вибір венчурного фонду безпосередньо впливає на швидкість масштабування продукту, вихід на нові ринки та здатність проходити наступні раунди інвестицій.

Курсова робота присвячена проєктуванню та розробці інформаційної системи Fund Matcher, яка забезпечує пошук і підбір венчурних фондів для стартапів на основі поєднання правил фільтрації та семантичного аналізу портфельних компаній. Система враховує стадію фінансування, галузь, географію, розмір чека та приховані інвестиційні інтереси фонду, що виявляються через аналіз його попередніх інвестицій.

Актуальність теми полягає у тому, що традиційні каталоги інвесторів зазвичай надають лише табличні фільтри: стадія, сектор, регіон, розмір фонду. Такий підхід не завжди дозволяє виявити фонди, які формально не вказують певний сектор, але фактично інвестують у схожі компанії. Тому використання векторних представлень текстів і семантичного пошуку дозволяє підвищити релевантність підбору.

Метою роботи є набуття практичних навичок проєктування та розробки інформаційних систем на прикладі веб-платформи для підбору венчурних фондів, реалізованої з використанням FastAPI, PostgreSQL, pgvector, SQLAlchemy, Alembic, OpenAI API та React з TypeScript.

Об’єктом дослідження є процес пошуку, оцінювання та ранжування венчурних фондів для стартапів. Предметом дослідження є методи побудови інформаційної системи, що поєднує реляційну базу даних, векторний пошук, API-сервіси та клієнтський інтерфейс.

Практична цінність системи полягає у створенні інструмента, який може використовуватися як прототип для автоматизації первинного пошуку інвесторів. У реальному стартап-процесі засновник витрачає значну кількість часу на перегляд таблиць, сайтів фондів, відкритих баз даних і профілів портфельних компаній. Fund Matcher переносить частину цієї роботи у програмну систему: користувач описує власний стартап, а система формує ранжований список потенційно релевантних фондів.

На відміну від простого каталогу, розроблений застосунок демонструє повний цикл побудови інформаційної системи: від моделювання даних і розробки API до семантичного пошуку, автентифікації користувачів, журналювання запитів та підготовки до розгортання. Це робить проєкт придатним не лише як навчальну роботу, а й як основу для подальшої практичної розробки.

Методологічна основа роботи

У процесі виконання роботи використано системний підхід до аналізу предметної області, методи реляційного моделювання даних, принципи клієнт-

серверної архітектури, REST API, векторного пошуку та об’єктно-орієнтованого проєктування. Окрему увагу приділено тому, щоб технічні рішення відповідали реальній логіці венчурного ринку: стадіям інвестування, галузевим мандатам, географічним обмеженням і портфельній історії фондів.

Інформаційною базою роботи є структура репозиторію Fund Matcher, вихідний код серверної частини, моделі бази даних, API-модулі, сценарії запуску середовища та загальні принципи побудови сучасних веб-застосунків. Усі пояснення в документі сформовані таким чином, щоб показати не окремі фрагменти коду, а загальну архітектурну логіку інформаційної системи.

Завдання роботи

- проаналізувати предметну область венчурного фінансування та структуру ринку інвесторів;
- визначити основні сутності системи: фонди, користувачі, портфельні компанії, джерела даних, пошукові запити;
- спроєктувати базу даних для зберігання фондів, портфельних компаній і векторних embeddings;
- реалізувати серверну частину на FastAPI з модульною структурою API;
- реалізувати алгоритм пошуку, який поєднує rule-based scoring і semantic search;
- описати підходи до тестування, розгортання та оптимізації системи.

Методи та інструменти дослідження

У роботі використано системний аналіз, методи проєктування баз даних, клієнт-серверну архітектуру, REST API, векторне представлення текстів, тестування API та порівняння результатів пошуку. Інструментарій включає Python 3.11, FastAPI, PostgreSQL 16, pgvector, SQLAlchemy, Alembic, Docker, OpenAI API, React, TypeScript, GitHub та Supabase як можливе хмарне середовище для PostgreSQL-сумісного розгортання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЕКОСИСТЕМИ ВЕНЧУРНОГО КАПІТАЛУ

1.1. Структура ринку венчурного капіталу

Венчурний капітал є формою фінансування інноваційних компаній із високим потенціалом зростання та підвищеним рівнем ризику. На відміну від банківського кредитування, венчурний фонд інвестує в частку компанії та очікує значного зростання вартості бізнесу в майбутньому. Для стартапів це особливо важливо, оскільки на ранніх етапах вони часто не мають стабільного прибутку або достатньої кредитної історії.

Ринок венчурних інвестицій структурований за стадіями розвитку компанії. Кожна стадія має власну логіку оцінювання ризику, вимоги до команди та продукту, очікуваний розмір раунду й тип інвестора. Fund Matcher враховує цю структуру через поля stages, sectors, geographies та check size у моделі фонду.

Таблиця 1.1 –

Характеристика основних стадій венчурного фінансування

Стадія	**Характеристика**	**Типові критерії фонду**
Pre-seed	Початкова перевірка ідеї, MVP або ранній прототип	сильна команда, чітка проблема, первинні користувачі
Seed	Перші продажі або доказ попиту	ринок, traction, масштабованість продукту
Series A	Підтверджена бізнес-модель	зростання виручки, повторювані продажі, unit economics
Series B	Масштабування компанії	розширення ринку, операційна ефективність, сильна команда

Окремим виміром є географічний мандат фонду. Частина інвесторів працює глобально, частина зосереджується на США, Європі, Великій Британії, Центральній та Східній Європі або конкретних локальних ринках. Для автоматизованого підбору важливо не лише зберігати географію, а й реалізувати гнучке порівняння, коли, наприклад, значення Global може відповідати стартапу з будь-якого регіону.

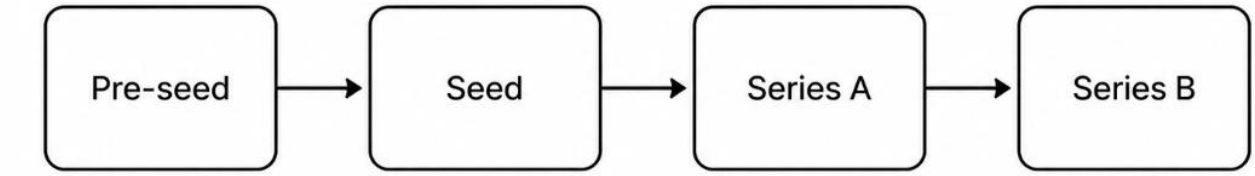


Рис. 1.1 – Схема стадій венчурного фінансування

Ринок венчурного капіталу відрізняється від класичного банківського фінансування тим, що інвестор бере на себе високий ризик в обмін на частку в компанії та потенційно значне зростання вартості цієї частки в майбутньому. Тому під час аналізу фондів важливо враховувати не лише загальний опис інвестора, а й реальну поведінку фонду: у які компанії він вже інвестував, на яких стадіях, у яких країнах і в яких секторах.

Для інформаційної системи це означає необхідність зберігати декілька рівнів даних. Перший рівень - формальні характеристики фонду: назва, сайт, штаб-квартира, стадії, сектори, географія, мінімальний і максимальний розмір чека. Другий рівень - історія портфельних компаній, яка показує фактичні інвестиційні інтереси. Третій рівень - векторні ознаки, які дають змогу порівнювати зміст описів стартапів і компаній за семантичною близькістю.

Стадія інвестування визначає не лише розмір фінансування, а й очікування фонду до зрілості бізнесу. Pre-seed зазвичай означає перевірку ідеї та ранній MVP, seed - перші ознаки попиту, Series A - підтверджену бізнес-модель, Series B - масштабування. Якщо система ігнорує цю різницю, вона може показувати фонди, які формально працюють у потрібній галузі, але не інвестують на відповідній стадії.

Географічний мандат має таке саме значення. Частина фондів інвестує глобально, але частина має юридичні, операційні або стратегічні обмеження щодо певних країн. Для стартапу з України або Європи це може бути вирішальним чинником, оскільки навіть високий семантичний збіг із портфелем фонду не гарантує релевантності, якщо фонд не працює в цьому регіоні.

1.2. Проблема підбору фондів і інформаційна асиметрія

Пошук релевантного фонду є складною інформаційною задачею. Стартап бачить публічні описи інвесторів, але не завжди має доступ до повної історії інвестицій, реальних пріоритетів партнерів фонду та неформальних критеріїв прийняття рішень. У результаті виникає інформаційна асиметрія: фонд має більше даних про власну стратегію, ніж засновник стартапу, який намагається оцінити його релевантність.

Традиційні фільтри працюють лише з явними ознаками: галузь, стадія, географія, розмір інвестиційного чека. Вони не виявляють приховані інтереси

фонду. Наприклад, фонд може не вказувати окремо “AI infrastructure”, але в його портфелі може бути кілька компаній із подібним технічним профілем. Саме тому Fund Matcher аналізує не тільки сам фонд, а й його портфельні компанії.

Семантичний пошук дозволяє перетворити опис стартапу та описи портфельних компаній у векторні представлення. Після цього система порівнює тексти не за точним збігом слів, а за змістовою близькістю. Такий підхід дає змогу знайти фонди, які мають релевантний інвестиційний досвід, навіть якщо їхні публічні категорії не повністю збігаються з описом стартапу.

- зменшується залежність від ручного пошуку у великих каталогах інвесторів;
- підвищується ймовірність знайти фонд із фактичним досвідом у потрібній ніші;
- результати ранжуються не лише за формальними фільтрами, а й за близькістю портфельних компаній;
- користувач отримує пояснення релевантності через список схожих компаній у портфелі фонду.

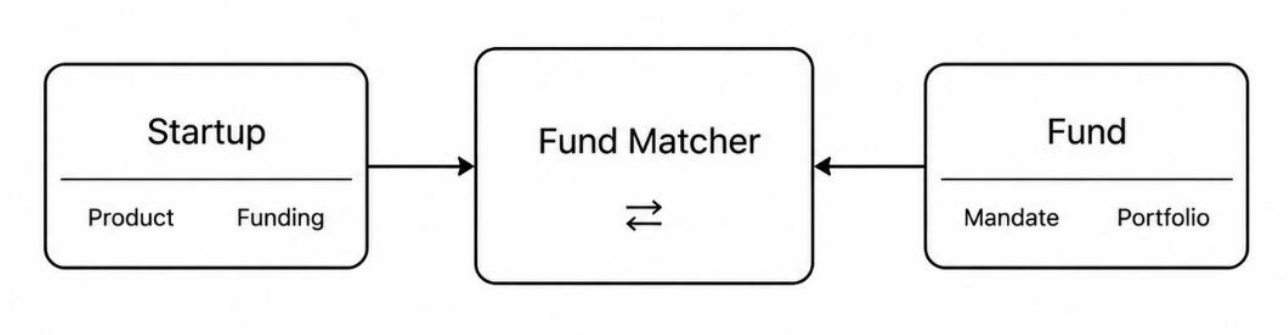


Рис. 1.2 – Схема взаємодії програми з двома сторонами ринку

Інформаційна нерівність проявляється також у різній якості відкритих даних. Деякі фонди детально описують інвестиційний фокус, публікують портфель, команду, інвестиційні тези та приклади угод. Інші обмежуються коротким описом або взагалі мають неповні профілі у відкритих джерелах. Через це система повинна бути стійкою до неповноти даних і використовувати кілька сигналів одночасно.

У Fund Matcher ця проблема вирішується комбінованим підходом. Rule-based filtering відсікає явно нерелевантні фонди за стадією, сектором, географією та розміром чека. Semantic search доповнює цей підхід, виявляючи близькість між описом стартапу та портфельними компаніями. У результаті система не покладається лише на один критерій, а будує узагальнену оцінку релевантності.

Прикладом прихованого інтересу може бути фонд, який формально зазначає сектор “Enterprise software”, але в портфелі має кілька компаній, що працюють із AI automation, developer tools або vertical SaaS. Звичайний фільтр може не знайти такий фонд за запитом “AI productivity platform”, тоді як векторний пошук здатний виявити змістовну близькість між описами.

Пояснюваність результатів є важливою вимогою до такої системи. Користувач повинен бачити не тільки фінальний список фондів, а й причину,

чому конкретний фонд потрапив у рекомендації. Саме тому в результатах варто показувати схожі портфельні компанії, similarity score, rule score і combined score.

1.3. Аналіз конкурентних рішень: Crunchbase, OpenVC, PitchBook

Для оцінки доцільності розробки Fund Matcher проведено огляд поширених платформ, які використовуються для пошуку інвесторів і збору ринкових даних. У цьому підрозділі конкуренти розглядаються не як повні аналоги, а як індустріальні стандарти, з якими можна порівняти функціональність курсового проєкту.

Таблиця 1.2 –

Порівняльний аналіз конкурентних платформ для підбору венчурних інвесторів

Критерій порівняння	**Crunchbase**	**OpenVC**	**PitchBook**	**Fund Matcher**
Основне призначення	Пошук інформації про компанії, фонди та інвестиційні раунди	Каталог венчурних фондів для засновників стартапів	Професійна аналітична платформа для інвесторів і фінансових компаній	Підбір релевантних венчурних фондів для стартапів
Цільова аудиторія	Стартапи, інвестори, аналітики	Засновники стартапів	Інвестиційні фонди, аналітики, корпорації	Засновники стартапів на ранніх етапах
Основні критерії підбору	Галузь, країна, раунд, інвестори, компанії	Стадія, сектор, географія, розмір чека	Ринкові дані, угоди, компанії, інвестори	Стадія, сектор, географія, check size, схожість портфельних компаній
Аналіз портфельних компаній	Наявний, але потребує ручного аналізу	Обмежений	Розширений, але орієнтований на професійну аналітику	Використовується як основа для matching-алгоритму
Семантичний пошук	Не є основною функцією	Не є основною функцією	Обмежено доступний аналітичних інструментів	Реалізований через OpenAI Embeddings і pgvector
Пояснення результатів	Переважно через відкриті профілі та дані	Через опис фонду та його критерії	Через аналітичні дані та звіти	Через similarity score, rule score і схожі портфельні компанії

Критерій порівняння	**Crunchbase**	**OpenVC**	**PitchBook**	**Fund Matcher**
Доступність для невеликого стартапу	Частково обмежена платними функціями	Доступна і зручна для стартапів	Висока вартість доступу	Може бути використана як власний прототип
Гнучкість адаптації	Обмежена можливостями платформи	Обмежена структурою каталогу	Обмежена закритою системою	Можна змінювати базу даних, алгоритм і критерії ранжування
Основна перевага	Велика база компаній та інвесторів	Простий доступ до списку фондів	Глибока фінансова та ринкова аналітика	Автоматизований підбір фондів на основі змістової схожості
Основне обмеження	Великий обсяг даних потребує ручного аналізу	Якість залежить від заповненості профілів фондів	Висока вартість і складність використання	Потребує розширення бази фондів і подальшого тестування

За результатами порівняння можна зробити висновок, що існуючі платформи переважно орієнтовані на зберігання та фільтрацію великого обсягу ринкових даних. Fund Matcher відрізняється тим, що робить акцент саме на автоматизованому підборі фондів для конкретного стартапу, використовуючи не лише формальні критерії, а й семантичну схожість із портфельними компаніями інвесторів.

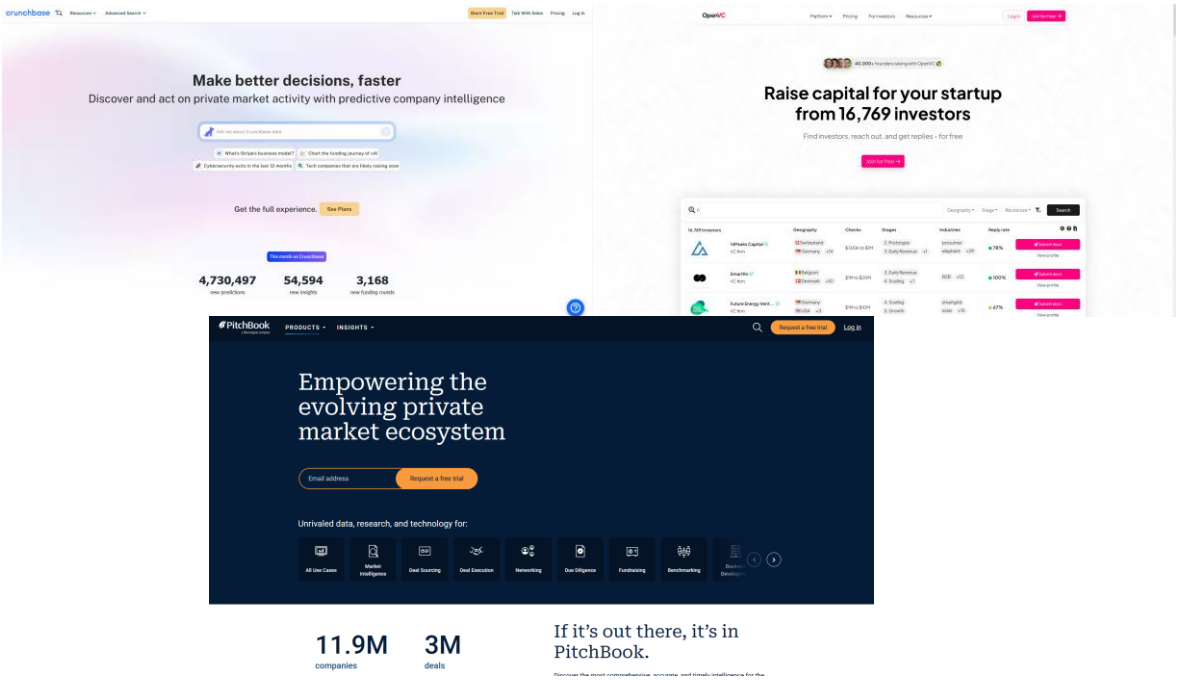


Рис. 1.3 – Головні сторінки конкурентних платформ

РОЗДІЛ 2
ПРОЄКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз даних системи Fund Matcher

Основою Fund Matcher є база даних, яка зберігає структуровану інформацію про венчурні фонди, портфельні компанії, користувачів і результати пошукових запитів. На відміну від простого довідника, система повинна підтримувати як традиційні SQL-запити, так і векторний пошук через pgvector.

Головними групами даних є: фондові профілі, інвестиційні критерії, портфельні компанії, embeddings, користувачі, історія пошуку та адміністративні зміни. Така структура дозволяє відокремити бізнес-дані від службових сутностей і підтримувати подальше розширення системи.

Таблиця 2.1 –

Основні групи даних системи Fund Matcher

Група даних	**Призначення**
Funds	зберігання основної інформації про фонд: назва, сайт, опис, стадії, сектори, географія, check size
Fund embeddings	векторне представлення канонічного тексту фонду для семантичного аналізу
Portfolio companies	опис компаній, у які інвестували фонди
Portfolio company embeddings	векторне представлення описів портфельних компаній
Users	автентифікація, ролі admin/founder, статус активності
Search queries	журнал запитів користувачів, фільтри, кількість результатів і час обробки

Окремою вимогою до даних є відокремлення тестових записів від реальних. У моделях фондів і портфельних компаній передбачено поле is_fake, яке дозволяє не враховувати seed-дані у реальному пошуку. Це важливо для тестування, оскільки розробник може наповнювати базу штучними прикладами, не змішуючи їх з продукційними даними.

Дані в системі можна поділити на операційні, аналітичні та службові. Операційні дані потрібні для основної роботи застосунку: фонди, користувачі, портфельні компанії. Аналітичні дані зберігають історію пошукових запитів, кількість результатів, top fund ids і час обробки. Службові дані містять технічні поля: timestamps, needs_refresh, embedding_model, import_source.

Важливою особливістю є наявність канонічного тексту для embeddings. Перед генерацією вектора система повинна привести дані до стабільної текстової

форми: назва, опис, сектори, стадії, географія та інші релевантні атрибути. Це зменшує випадковість у генерації векторів і робить результати пошуку більш стабільними.

2.2. Проєктування структури бази даних

Структура бази даних побудована за реляційним принципом. Основна сутність funds пов'язана з портфельними компаніями через проміжну таблицю fund_portfolio_companies. Така модель дозволяє зберігати зв'язок many-to-many, оскільки одна компанія може мати кількох інвесторів, а один фонд може інвестувати в багато компаній.

Для векторного пошуку створено окремі таблиці embeddings. Це рішення спрощує оновлення векторів, дозволяє позначати записи як needs_refresh і не змішує велику технічну структуру embedding_vector з основними бізнес-полями.

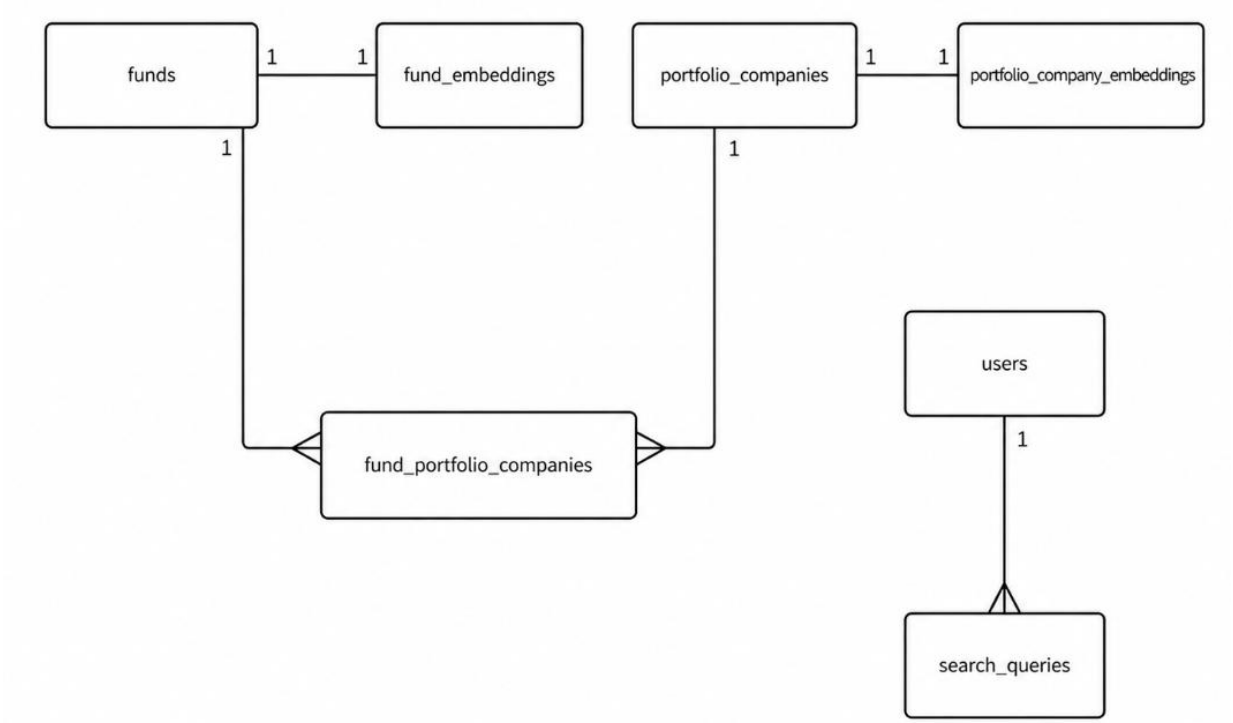


Рис. 2.1 – ER-діаграма бази даних Fund Matcher

У структурі бази даних використано принцип нормалізації: фондові дані, портфельні компанії, зв'язки інвестицій і векторні представлення винесені в окремі таблиці. Це зменшує дублювання, спрощує оновлення записів і дозволяє зберігати додаткові атрибути зв'язку між фондом і компанією, наприклад round, role, amount_invested, invested_at та exit_status.

Таблиця fund_portfolio_companies є ключовою для моделі даних. Вона не просто поєднує фонд і компанію, а зберігає контекст інвестиції. Завдяки цьому в майбутньому можна реалізувати більш точне ранжування: наприклад, надавати більшу вагу фондам, які інвестували як lead investor або брали участь у раундах, близьких до стадії стартапу користувача.

Для підтримки якості даних передбачені унікальні обмеження. Наприклад, одна й та сама портфельна компанія не повинна дублюватися для одного фонду в проміжній таблиці. Аналогічно зовнішні джерела можна ідентифікувати через

source_name та external_id, що дає змогу уникати повторного імпорту тих самих записів.

2.3. Реалізація бази даних у PostgreSQL та pgvector

Фізична реалізація бази даних орієнтована на PostgreSQL 16 із розширенням pgvector. У локальному середовищі база запускається через Docker Compose на образі pgvector/pgvector:pg16. Такий підхід забезпечує відтворюване середовище розробки, де кожен розробник може підняти однакову базу даних без ручної інсталяції PostgreSQL.

Розширення pgvector використовується для зберігання embedding_vector розмірності 1536, що відповідає моделі text-embedding-3-small. Порівняння виконується за допомогою cosine distance, а результат перетворюється у similarity score через вираз 1 - distance.

- PostgreSQL забезпечує транзакційність і надійність реляційних даних;
- pgvector додає можливість швидкого пошуку за близькістю векторів;
- Alembic використовується для керування міграціями;
- SQLAlchemy описує моделі фондів, користувачів і портфельних компаній;
- Docker Compose стандартизує локальне середовище.

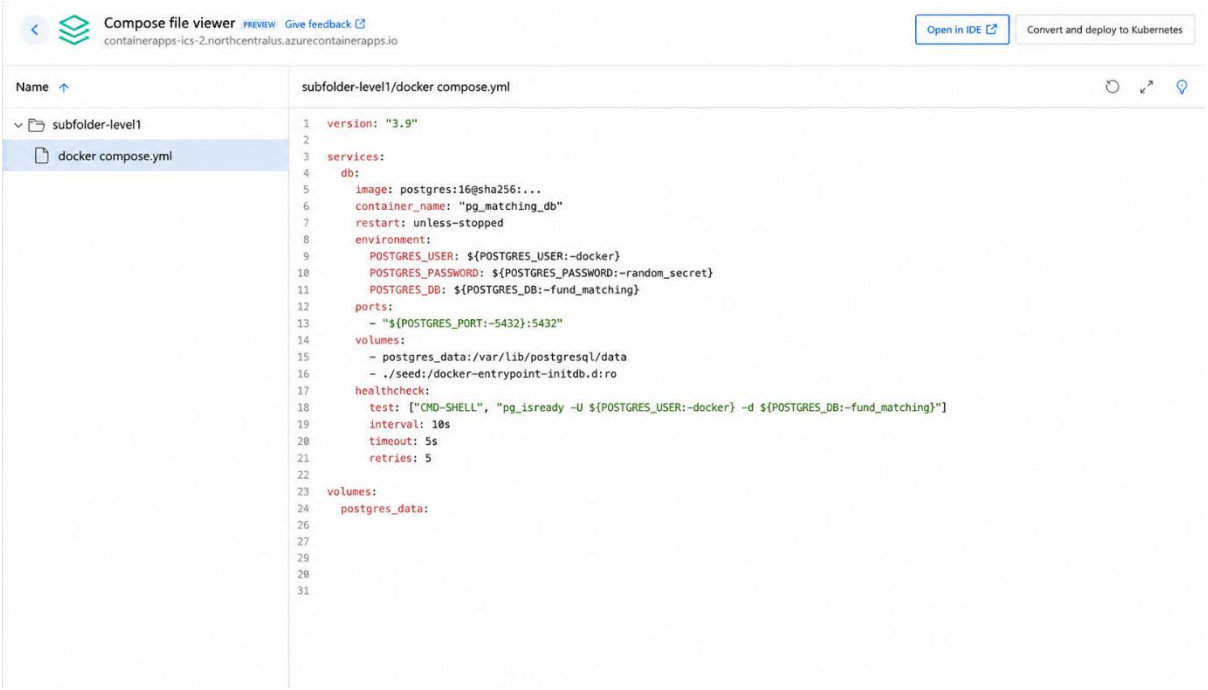


Рис. 2.2 – PostgreSQL/pgvector у Docker

PostgreSQL обрано як базу даних через поєднання зрілої реляційної моделі та можливості розширення через pgvector. Це дозволяє не вводити окрему спеціалізовану векторну базу на ранньому етапі розробки, а зберігати звичайні таблиці та векторні поля в одній СУБД. Для навчального проєкту це зменшує інфраструктурну складність і полегшує розгортання.

У локальному середовищі Docker Compose піднімає контейнер бази даних, задає користувача, пароль, назву бази, порт і volume для збереження даних. Таке рішення гарантує, що база не втрачається після перезапуску контейнера, а всі члени команди можуть отримати однакове середовище.

Міграції Alembic забезпечують контрольовані зміни структури. Замість ручного редагування таблиць розробник створює міграційний файл, перевіряє його, застосовує командою alembic upgrade head і може повернутися назад у разі помилки. Це особливо важливо для системи, де є багато зв’язків між таблицями та технічні поля embeddings.

Для продукційного середовища логічним наступним кроком є використання Supabase як керованої PostgreSQL-платформи. У такій конфігурації локальна база залишається для розробки, а Supabase використовується для тестового або демонстраційного середовища, де можна перевіряти роботу API з реальними підключеннями.

РОЗДІЛ 3
СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Загальна архітектура системи

Fund Matcher побудовано як веб-орієнтовану клієнт-серверну інформаційну систему. Серверна частина відповідає за автентифікацію, CRUD-операції, пошук, генерацію embeddings, імпорт даних і журналювання пошукових запитів. Клієнтська частина забезпечує введення профілю стартапу, перегляд результатів, адміністративні дії та зручну взаємодію користувача з API.

1. користувач вводить опис стартапу, стадію, сектор, географію та суму раунду;
2. frontend надсилає запит на FastAPI backend;
3. backend генерує embedding для опису стартапу через OpenAI API;
4. система застосовує rule-based filtering за стадією, сектором, географією та check size;
5. pgvector порівнює startup embedding з embeddings портфельних компаній;
6. результати агрегуються, ранжуються та повертаються користувачу.

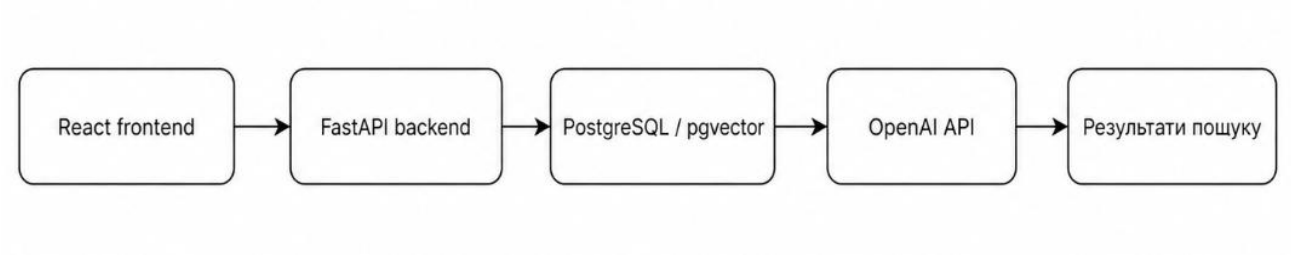


Рис. 3.1 – Загальна архітектурна схема Fund Matcher

Архітектуру системи можна розглядати як набір ізольованих рівнів. Presentation layer реалізований у frontend-додатку. API layer відповідає за обробку HTTP-запитів. Service layer містить бізнес-логіку: фільтрацію, обчислення балів, генерацію embeddings, роботу з OpenAI. Data access layer взаємодіє з PostgreSQL через SQLAlchemy.

Таке розділення підвищує підтримуваність проєкту. Наприклад, зміну формули combined score можна виконати у search_service без зміни frontend-компонентів або моделей бази даних. Аналогічно зміну способу автентифікації можна ізолювати в auth-модулі.

Функціональні вимоги системи включають реєстрацію та автентифікацію користувачів, зберігання фондів, імпорт портфельних компаній, пошук фондів, калібрування similarity threshold, генерацію intro message та перегляд історії пошуків. Нефункціональні вимоги включають швидкодію, безпеку, масштабованість, відтворюваність середовища і зрозумілу структуру коду.

3.2. Серверна частина: FastAPI, SQLAlchemy, Alembic

Серверна частина реалізована на FastAPI. Фреймворк забезпечує високу швидкість обробки HTTP-запитів, автоматичну генерацію OpenAPI-документації, підтримку Pydantic-схем і зручне розділення логіки за routers, services, schemas та models.

SQLAlchemy використовується для опису моделей бази даних і роботи з ORM-рівнем. Alembic відповідає за створення та застосування міграцій. Така комбінація дозволяє підтримувати контрольовану еволюцію схеми бази даних і синхронізувати структуру між локальним середовищем, тестовим сервером і Supabase.

- auth router: реєстрація, вхід, перевірка поточного користувача;
- funds router: робота з фондами та їхніми профілями;
- portfolio router: імпорт і керування портфельними компаніями;
- search router: основна логіка matching і calibration;
- admin router: перегляд та прийняття змін, доступний лише адміністратору.

FastAPI також корисний тим, що автоматично формує інтерактивну документацію API. Для курсового проєкту це має практичне значення: можна показати маршрути, схеми запитів, структуру відповідей і перевірити основні сценарії без написання окремого клієнта. Це полегшує тестування та демонстрацію системи.

Pydantic-схеми відіграють роль контракту між frontend і backend. Вони визначають, які поля має надсилати користувач для пошуку, які типи даних дозволені, які значення є необов'язковими, і яку структуру отримає клієнт у відповіді. Це зменшує кількість помилок під час інтеграції.

Серверна логіка побудована так, щоб помилки зовнішніх сервісів не руйнували весь застосунок. Якщо OpenAI API недоступний або ключ не налаштовано, система повертає контрольовану помилку з відповідним HTTP-статусом. Такий підхід є важливим для надійності, оскільки AI-модуль залежить від зовнішньої інфраструктури.

Автентифікація реалізована на основі JWT. Користувач після входу отримує токен, який надалі передається у захищені маршрути. Це дозволяє розділити ролі founder та admin і обмежити доступ до адміністративних операцій, пов'язаних з імпортом або зміною фондів.

3.3. Клієнтська частина: React та TypeScript

Клієнтська частина побудована на React 18 і TypeScript. Вона виконує роль інтерфейсу користувача для введення профілю стартапу, налаштування пошуку, перегляду знайдених фондів і аналізу схожих портфельних компаній. TypeScript зменшує кількість помилок під час роботи з API-відповідями та забезпечує кращу підтримуваність коду.

Інтерфейс має бути простим і орієнтованим на робочий сценарій засновника стартапу: швидко описати компанію, отримати релевантні фонди, побачити пояснення відповідності та за потреби згенерувати intro message для звернення до інвестора.

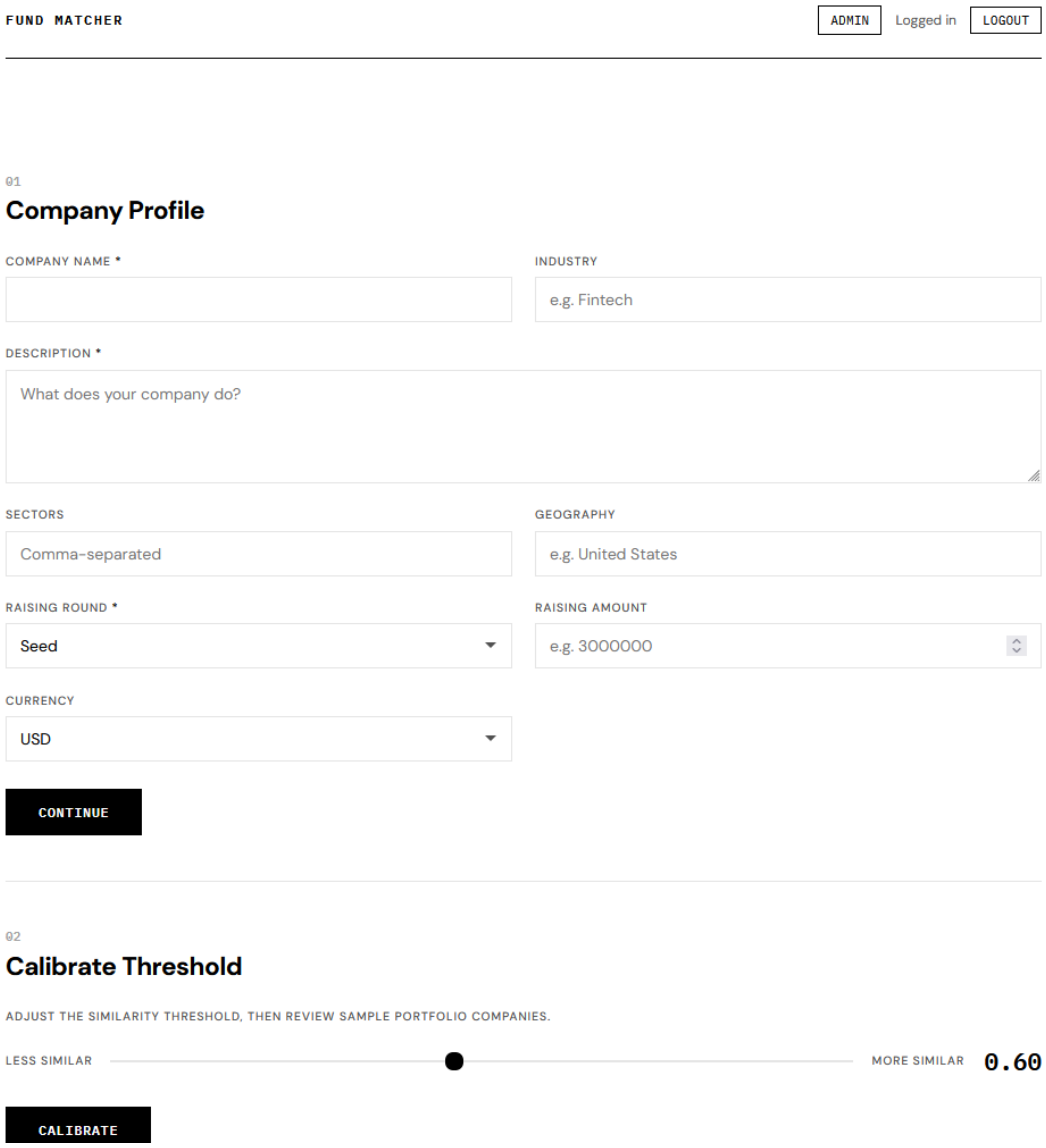


Рис. 3.2 – Скріншот форми пошуку Fund Matcher

Клієнтська частина повинна бути достатньо простою для користувача, який не є технічним спеціалістом. Головний сценарій передбачає введення опису стартапу, вибір стадії фінансування, секторів, географії, суми раунду та запуск пошуку. Після цього користувач має отримати результати у зрозумілому вигляді: назва фонду, короткий опис, причини відповідності та схожі портфельні компанії.

TypeScript допомагає підтримувати відповідність типів між API-відповідями та компонентами інтерфейсу. Якщо backend повертає FundMatchResult з полями rule_score, portfolio_score, combined_score і matching_portfolio_companies, frontend може описати ці дані типами й уникнути помилок при відображенні.

Окрему увагу слід приділяти станам інтерфейсу: loading, empty state, error state і success state. Для такого типу системи важливо не залишати користувача без пояснення, якщо пошук триває довше через генерацію embedding або якщо за заданим threshold не знайдено жодного результату.

Інтерфейс також може виконувати навчальну функцію. Наприклад, біля similarity threshold можна показувати підказку, що нижче значення дає більше

результатів, але менш точні збіги, а вище значення робить пошук суворішим. Це допомагає користувачу краще розуміти логіку системи.

3.4. Інтеграція, середовище розробки та конфігурації

Для локального запуску використовуються Python virtual environment, requirements.txt, Docker Compose, змінні середовища .env та команди Alembic. Файл .env містить DATABASE_URL, JWT_SECRET_KEY, OPENAI_API_KEY та інші параметри конфігурації. Секрети не повинні потрапляти в репозиторій.

GitHub використовується як система керування версіями. Репозиторій містить структуру backend, frontend, scripts, alembic та конфігураційні файли. Для командної розробки можуть використовуватись окремі гілки feature/* і Pull Request перед злиттям у main.

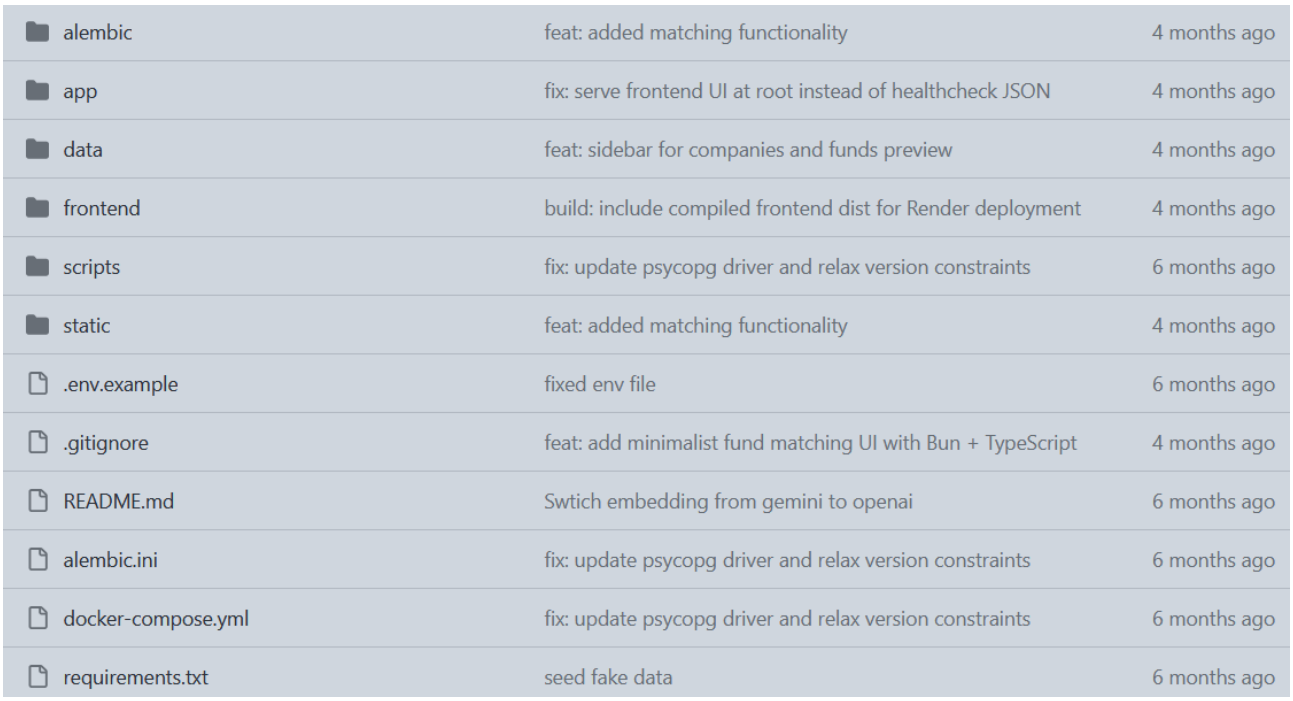


Рис. 3.3 – Скріншот структури репозиторію GitHub

Конфігурація системи винесена у змінні середовища. Це стандартний підхід для веб-застосунків, оскільки секретні ключі, URL бази даних і параметри API не повинні зберігатися безпосередньо в коді. У репозиторії може бути лише .env.example, який показує потрібні назви змінних без реальних секретів.

GitHub використовується як система контролю версій. Це дає змогу зберігати історію змін, створювати окремі гілки для нових функцій, повертатися до стабільних версій і документувати процес розробки. Навіть для індивідуального проєкту така організація є корисною, оскільки робить роботу відтворюваною.

Середовище розробки має бути максимально близьким до середовища розгортання. Якщо база даних у production використовує PostgreSQL, то локальна база також повинна бути PostgreSQL, а не SQLite або інша спрощена СУБД. Це зменшує ризик помилок, пов’язаних із різницею типів даних, SQL-синтаксису або поведінки індексів.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Реалізація серверної частини

Точка входу серверної частини створює FastAPI-застосунок, підключає CORS middleware, реєструє маршрути auth, funds, portfolio, search та admin, а також запускає фоновий scheduler під час старту застосунку. Наявність health endpoint дозволяє перевіряти працездатність сервісу під час локального запуску або майбутнього деплою.

API побудовано ресурсно-орієнтовано. Кожен router відповідає за окрему частину предметної області. Це дозволяє уникнути монолітного файлу з усією логікою та робить проєкт простішим для супроводу.

Серверна частина організована за принципом маршрути - схеми - сервіси - моделі. Маршрути відповідають за приймання HTTP-запитів, схеми перевіряють структуру даних, сервіси виконують бізнес-логіку, а моделі описують таблиці бази даних. Така структура зменшує зв’язність компонентів і дозволяє змінювати окремі частини без переписування всього застосунку.

Основний search endpoint працює як координатор декількох операцій. Він приймає опис стартапу, викликає генерацію embedding, запускає rule-based filtering, виконує векторний пошук, агрегує результати, записує інформацію про запит у search_queries і повертає структуровану відповідь frontend-частині.

Калібрування threshold реалізовано як окремий сценарій, оскільки користувачу не завжди зрозуміло, яке значення similarity є оптимальним. Система може показати приклади портфельних компаній на певному порозі, після чого користувач вирішує, чи потрібно зробити пошук ширшим або точнішим.

4.2. Реалізація моделей даних

Моделі Fund, FundEmbedding, FundSource, FundPendingChange, FundInternal, User, PortfolioCompany, FundPortfolioCompany та PortfolioCompanyEmbedding описують основні таблиці системи. Особливу роль відіграють embedding-таблиці, які зберігають вектори та службові поля needs_refresh і last_refreshed_at.

Модель User містить email, hashed_password, full_name, role, is_active та часові поля. Ролі admin і founder дозволяють розділити доступ до адміністративних операцій та звичайного пошуку.

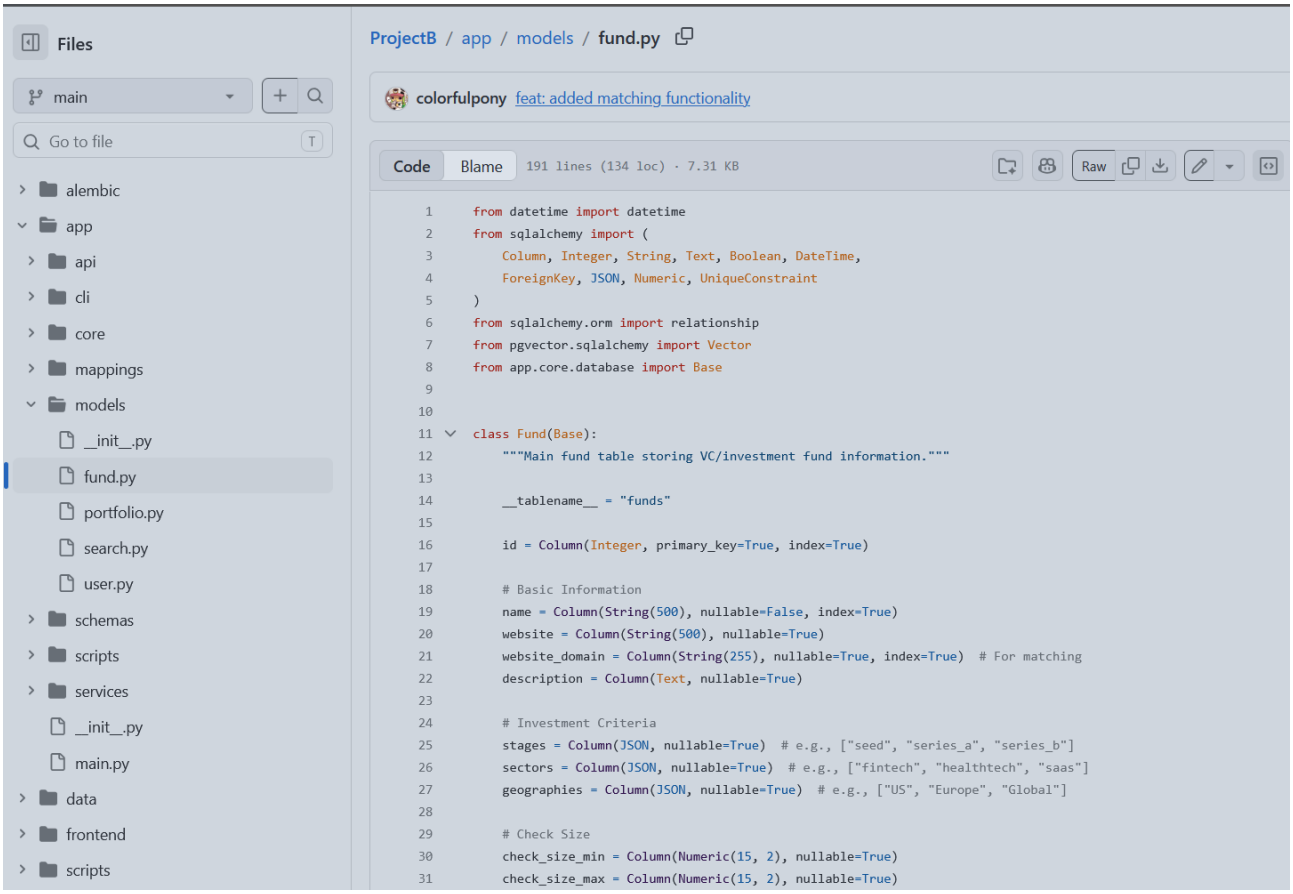


Рис. 4.1 – Фрагмент моделей SQLAlchemy

Модель Fund містить як базову інформацію, так і інвестиційні критерії. Поля stages, sectors і geographies зберігаються у JSON-форматі, що дозволяє одному фонду мати кілька значень без створення надмірної кількості проміжних таблиць на початковому етапі. Для навчального прототипу це практичний баланс між нормалізацією і простотою розробки.

Модель PortfolioCompany фіксує компанії, у які інвестували фонди. Окреме зберігання цих компаній дозволяє будувати пояснення результатів: система може не лише сказати, що фонд релевантний, а й показати, які саме портфельні компанії мають змістовну схожість із запитом користувача.

Модель PortfolioCompanyEmbedding зберігає embedding_vector, canonical_text, embedding_model і технічні поля оновлення. Винесення embeddings в окрему таблицю спрощує повторну генерацію векторів, коли змінюється опис компанії або використовується нова embedding-модель.

Модель User забезпечує базову автентифікацію і авторизацію. Поле role дозволяє розділити звичайного користувача-засновника та адміністратора, який має доступ до розширених операцій, таких як імпорт даних, перегляд внутрішніх полів або підтвердження змін.

4.3. Реалізація алгоритму пошуку та інтеграції з OpenAI

Алгоритм пошуку складається з двох взаємодоповнювальних частин: rule-based scoring і semantic scoring. Rule-based scoring оцінює відповідність фонду за стадією, сектором, географією та розміром чека. Semantic scoring шукає схожі портфельні компанії через pgvector і OpenAI embeddings.

Для розрахунку rule score застосовується вагова модель: stage match, sector match, geography match і check size match. Портфельний score враховує кількість знайдених схожих компаній і середню similarity. Після цього формується combined score, де семантична складова має більшу вагу, оскільки вона показує фактичну інвестиційну історію фонду.

Таблиця 4.1 –

Основні компоненти алгоритму пошуку Fund Matcher

Компонент	**Роль у matching logic**
Rule score	перевіряє формальні критерії фонду
Startup embedding	векторне представлення опису стартапу
Portfolio similarity	порівняння з портфельними компаніями фонду
Portfolio score	агрегація кількості та якості збігів
Combined score	фінальне ранжування фондів

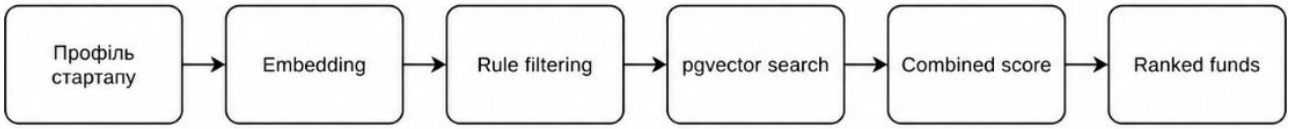


Рис. 4.2 – Блок-схема алгоритму пошуку

Додатково система підтримує calibration endpoint, який допомагає користувачу підібрати similarity threshold. Він показує приклади портфельних компаній, що проходять заданий поріг, і дозволяє зрозуміти, чи результати занадто широкі або надто вузькі.

Алгоритм пошуку складається з двох логічних частин. Перша частина є детермінованою: система перевіряє відповідність стадії, сектору, географії та розміру чека. Друга частина є семантичною: опис стартапу порівнюється з embeddings портфельних компаній, а не лише з категоріями фонду.

Rule score дозволяє не втрачати бізнес-логіку ринку. Наприклад, навіть якщо портфельні компанії семантично схожі на опис стартапу, фонд може бути нерелевантним, якщо він інвестує тільки у Series B, а користувач шукає pre-seed фінансування. Тому формальні критерії залишаються важливою частиною оцінки.

Portfolio score відображає фактичну інвестиційну історію фонду. Якщо у фонду є кілька портфельних компаній, схожих на стартап користувача, це підвищує довіру до рекомендації. Система враховує як кількість збігів, так і середню similarity.

Combined score поєднує обидва підходи. У поточній логіці більша вага надається семантичному портфельному збігу, оскільки він краще відображає

реальний досвід фонду. Водночас rule score не дозволяє повністю ігнорувати формальні обмеження інвестиційного мандату.

Інтеграція з OpenAI використовується не для прийняття непрозорого рішення, а для отримання embedding-вектора. Остаточна логіка ранжування залишається у власному коді системи, що робить її контрольованою, пояснюваною і придатною для тестування.

РОЗДІЛ 5
РОЗГОРТАННЯ, ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ

5.1. Процес розгортання та контейнеризація

Розгортання Fund Matcher починається з підготовки середовища: створення Python virtual environment, встановлення залежностей з requirements.txt, налаштування .env, запуску PostgreSQL/pgvector через Docker Compose та застосування міграцій Alembic. Така послідовність дозволяє відтворити однакове середовище на різних комп'ютерах.

- створити .env на основі .env.example і заповнити DATABASE_URL, JWT_SECRET_KEY, OPENAI_API_KEY;
- запустити контейнер бази даних командою docker-compose up -d;
- перевірити healthcheck контейнера PostgreSQL;
- застосувати міграції командою alembic upgrade head;
- завантажити початкові дані через seed або import scripts;
- запустити API сервер командою uvicorn app.main:app --reload;
- відкрити /docs або /health для перевірки API.

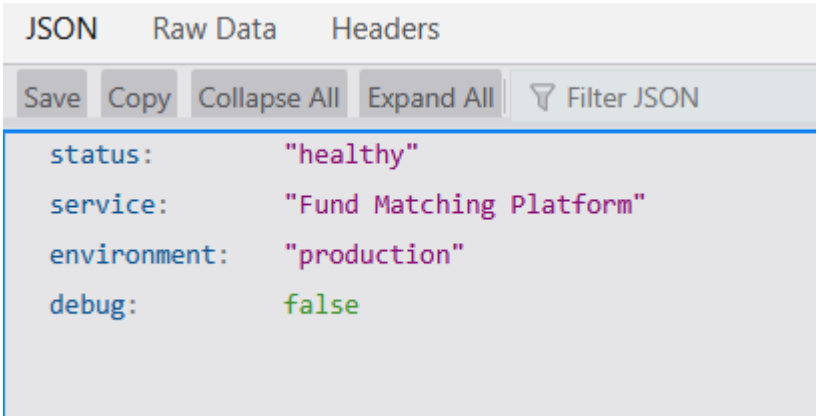


Рис. 5.1 – Скріншот успішного запуску backend і health endpoint

Контейнеризація є важливою не лише для запуску бази даних, а й для стабільності всього процесу розробки. Якщо середовище описане у docker-compose.yml, його можна швидко відтворити на іншому комп'ютері або сервері. Це зменшує типові помилки, коли застосунок працює в одного розробника, але не запускається в іншого через різні версії PostgreSQL або відсутність pgvector.

Процес розгортання складається з підготовки змінних середовища, запуску контейнерів, перевірки стану бази, застосування міграцій, завантаження початкових даних, запуску API та перевірки health endpoint. Така послідовність дозволяє швидко виявити, на якому саме етапі виникла проблема.

У майбутньому серверну частину можна також контейнеризувати окремим Dockerfile. Тоді docker-compose міститиме два основні сервіси: api та db. Це зробить локальний запуск ще простішим: одна команда підніматиме і backend, і базу даних.

5.2. Синхронізація з Supabase

Оскільки Fund Matcher використовує PostgreSQL, система може бути перенесена на Supabase як хмарне PostgreSQL-середовище. Supabase доцільно

використовувати для тестового або демонстраційного розгортання, оскільки він надає керовану базу даних, панель перегляду таблиць, connection string і підтримку розширень PostgreSQL, якщо вони доступні в обраному тарифі та конфігурації.

Синхронізація з Supabase передбачає зміну DATABASE_URL на Supabase connection string, застосування Alembic migrations до віддаленої бази та імпорт початкових даних. Важливо перевірити наявність pgvector у Supabase-проєкті, оскільки без нього semantic search не зможе працювати повноцінно:

- локальна база використовується для розробки та швидких експериментів;
- Supabase використовується як віддалена база для демонстрації або тестування;
- міграції Alembic підтримують однакову структуру локальної та віддаленої БД;
- секрети Supabase не повинні зберігатися в репозиторії;
- після синхронізації потрібно перевірити роботу search endpoint і наявність vector columns.

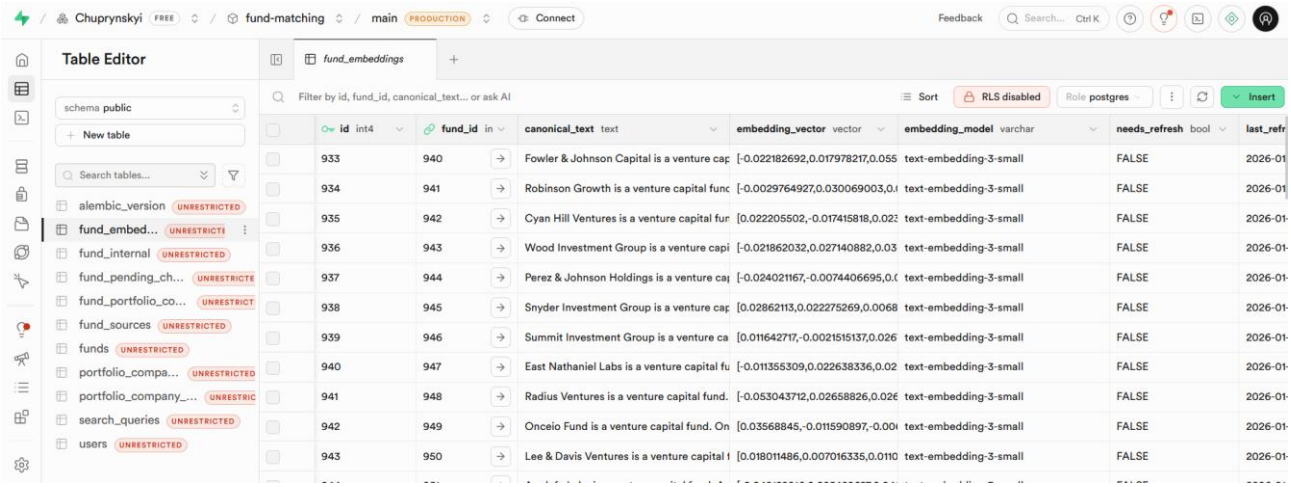


Рис. 5.2 – Скріншот Supabase Table Editor або Database settings

Supabase може використовуватися як керована PostgreSQL-платформа для демонстраційного або тестового середовища. Її перевага полягає в тому, що база даних не потребує ручного адміністрування сервера, але зберігає сумісність із PostgreSQL і може підтримувати розширення pgvector.

Синхронізація з Supabase повинна виконуватися контрольовано. Спочатку створюється проєкт і активується потрібне розширення, потім формується DATABASE_URL, після чого застосовуються Alembic migrations. Дані можна переносити через SQL dump, CSV import або окремі seed/import scripts.

Особливу увагу потрібно приділити секретам. DATABASE_URL для Supabase не повинен потрапляти в репозиторій. Для локальної розробки використовується .env, а для deployment-середовища - змінні середовища платформи, наприклад Render, Railway, Fly.io або інший сервіс.

Після синхронізації необхідно перевірити три речі: чи доступна база для API, чи застосовані всі міграції, чи працюють векторні запити. Простого

підключення до PostgreSQL недостатньо, оскільки пошукова логіка залежить саме від типу vector і операторів pgvector.

5.3. Бенчмаркінг продуктивності

Для оцінки продуктивності системи доцільно порівнювати два типи запитів: стандартні SQL-фільтри та семантичні запити з pgvector. Стандартні фільтри працюють швидше, оскільки використовують прості умови за індексованими полями. Семантичний пошук складніший, оскільки порівнює embedding користувацького запиту з векторами портфельних компаній.

Бенчмаркінг повинен вимірювати latency, кількість оброблених фондів, кількість портфельних компаній з embeddings, час генерації embedding через OpenAI API та час виконання SQL-запиту з pgvector. Для чесного порівняння OpenAI latency варто вимірювати окремо від latency бази даних.

Таблиця 5.1 –

Метрики оцінювання продуктивності системи Fund Matcher

Метрика	**Що вимірює**	**Очікуване застосування**
SQL filter latency	час фільтрації за stage, sector, geography, check size	оцінка базової швидкодії реляційного пошуку
Embedding generation time	час отримання vector representation для startup description	вплив зовнішнього OpenAI API
Vector query latency	час pgvector-пошуку по portfolio_company_embeddings	оцінка семантичного пошуку
Total request time	повний час від запиту до відповіді API	користувацьке сприйняття швидкості

01

Company Profile

COMPANY NAME *

NovaLedger AI

INDUSTRY

Fintech

DESCRIPTION *

NovaLedger AI is a financial automation platform for small and medium-sized businesses. The system uses artificial intelligence to categorize transactions, forecast cash flow, detect unusual expenses, and generate simple financial reports. The product helps founders and finance teams reduce manual accounting work and make faster decisions based on real-time business data.

SECTORS

fintech, artificial intelligence, SaaS

GEOGRAPHY

United States

RAISING ROUND *

Seed

RAISING AMOUNT

3000000

CURRENCY

USD

COMPANY	FUND	SIMILARITY	FUND CRITERIA
iPaidThat	Adelie Capital	0.745	MATCH
SaaS platform automating finance management and accounting for small and medium businesses.			
Helu	CommerzVentures	0.730	MATCH
Helu offers an integrated financial management platform that automates reporting and consolidates data for small to medium-sized enterprises (SMEs). Their service helps businesses streamline financial processes, reduce reporting time by 50%, and minimize errors, allowing teams to focus on growth rather than tedious tasks.			
Cash Director	EEC Ventures	0.728	MATCH
Cash Director is AI-enabled Digital CFO for SMEs. The service is based on a real-time accounting platform that automates daily financial tasks and helps manage day-to-day cash flow.			
FinnT	Diaspora Ventures	0.720	MATCH
FinnT offers AI-powered automation solutions specifically designed for enterprise finance, targeting companies with complex accounting needs. Their service automates repetitive and manual accounting tasks, allowing finance teams to focus on more strategic activities while ensuring data security and seamless integration with existing systems.			
Centinel	Abac Nest Ventures	0.709	MATCH
Centinel is an AI-powered SaaS platform that automates accounts payable and accounts receivable for SMBs, eliminating manual tasks and improving financial management.			
Regate	360 Capital	0.703	MATCH

Рис. 5.3 – Графік або таблиця результатів бенчмаркінгу

Бенчмаркінг у Fund Matcher доцільно проводити окремо для стандартних SQL-фільтрів і семантичних запитів. Звичайні фільтри за stage, sector, geography та check size працюють із невеликими полями й можуть бути оптимізовані індексами. Векторний пошук є дорожчим, оскільки порівнює embedding запиту з багатьма embedding-векторами портфельних компаній.

Для вимірювання продуктивності можна фіксувати час генерації embedding, час rule-based filtering, час pgvector-запиту, час агрегації результатів і

повний processing_time_ms. Такий поділ дозволяє зрозуміти, де саме виникає затримка: у зовнішньому OpenAI API, у базі даних або в Python-логіці агрегації.

Для невеликої навчальної бази повний перебір векторів може бути достатнім. Але при зростанні кількості портфельних компаній потрібно розглядати індекси pgvector, кешування embeddings, попередню фільтрацію за сектором або батчову обробку. Важливо, щоб оптимізація не погіршувала якість результатів.

Якісним результатом бенчмаркінгу є не лише середній час відповіді, а й порівняння сценаріїв. Наприклад, пошук із широкими фільтрами може обробляти значно більше фондів, ніж пошук із конкретною стадією та географією. Тому тестові профілі повинні відображати різні навантаження.

5.4. Перевірка точності пошуку

Точність пошуку перевіряється якісно, через реальні або наближені до реальних профілі стартапів. Для кожного профілю створюється опис компанії, галузь, стадія, географія та розмір раунду. Потім система повертає список фондів, а результати оцінюються за релевантністю: чи відповідає фонд стадії, чи має схожі портфельні компанії, чи логічним є пояснення similarity.

Для практичного тестування можна створити набір профілів: B2B SaaS, fintech, healthtech, AI infrastructure, climate tech. Для кожного профілю потрібно перевірити top-5 результатів і записати, чи є серед них фонди, які реально підходять за портфельною історією. Така перевірка не замінює математичну метрику precision/recall, але добре підходить для курсового прототипу.

- перевірити, чи не домінують лише формальні фільтри без семантичної близькості;
- перевірити, чи не повертаються фонди без відповідних portfolio matches;
- порівняти результати при різних similarity thresholds;
- проаналізувати помилкові збіги та занадто вузькі результати;
- зафіксувати приклади, де semantic search знайшов фонд, який не був би очевидним через стандартні фільтри.

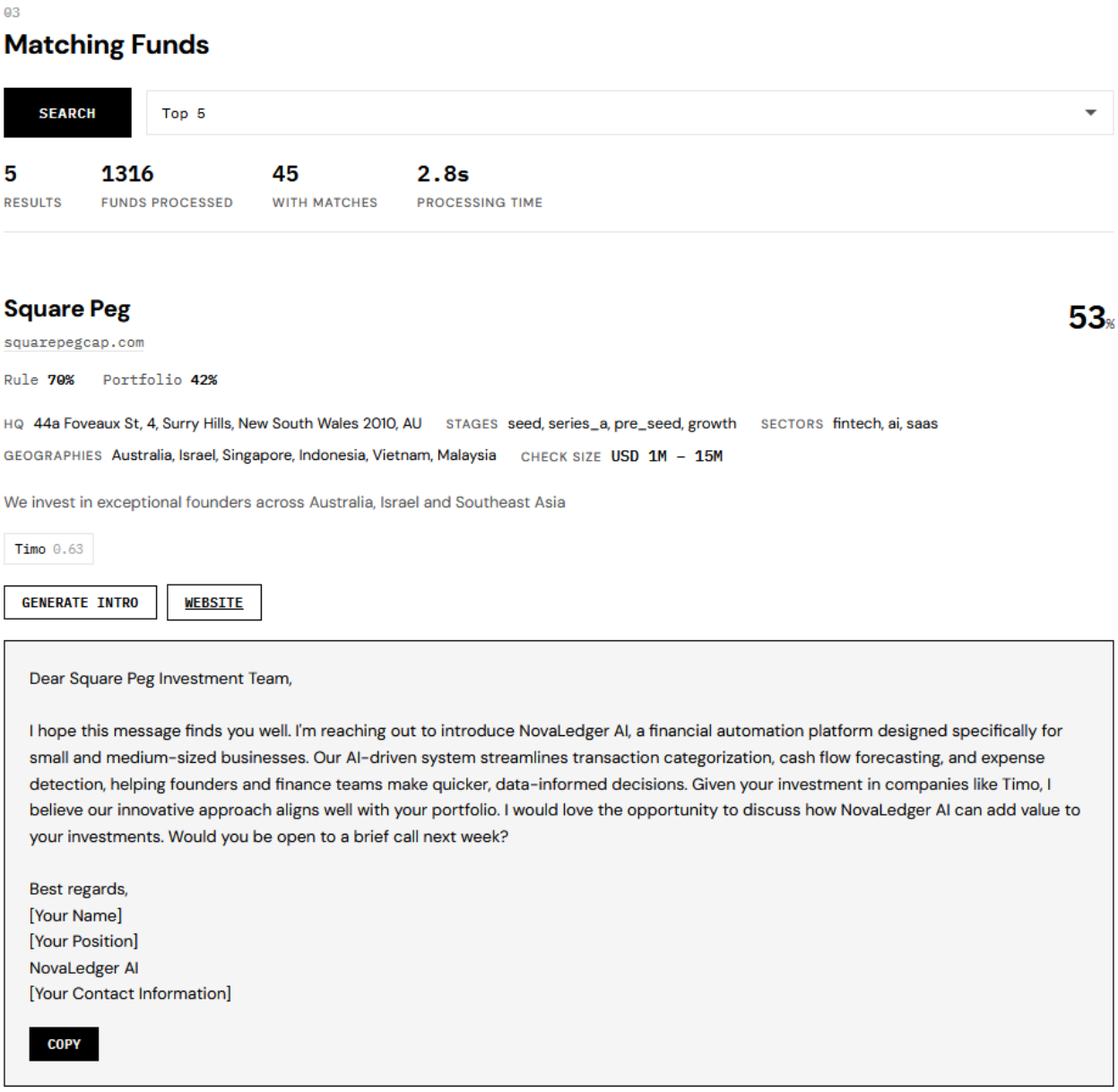


Рис. 5.4 – Приклад результатів пошуку для реального startup profile

Точність пошуку в такій системі не можна повністю оцінити лише автоматичним тестом, оскільки релевантність фонду має якісний характер. Тому доцільно використовувати набір реалістичних профілів стартапів і вручну оцінювати, чи логічними є отримані рекомендації.

Прикладами тестових профілів можуть бути: B2B SaaS для фінансової звітності, AI-платформа для автоматизації продажів, healthtech-сервіс для віддаленого моніторингу пацієнтів, developer tool для командної розробки або marketplace для логістики. Для кожного профілю перевіряється, чи потрапляють у топ результати фонди з відповідним портфелем.

Окремо тестується вплив threshold. Низький threshold повинен давати більше результатів, але серед них може бути більше слабких збігів. Високий threshold повинен залишати менше фондів, але з кращою семантичною схожістю. Калібрування допомагає користувачу знайти баланс між шириною і точністю пошуку.

Корисним критерієм є explainability check: якщо система показує фонд, вона повинна показати хоча б одну портфельну компанію, яка пояснює цю рекомендацію. Якщо пояснення виглядає нелогічним, потрібно переглянути canonical_text, embedding-підхід або формулу scoring.

Тестування точності також повинно враховувати негативні сценарії. Наприклад, якщо користувач вводить стартап із галузі, якої немає у базі, система не повинна штучно показувати нерелевантні фонди як високоякісні результати. У такому разі краще повернути менше результатів і запропонувати знизити threshold або розширити фільтри.

РОЗДІЛ 6
МОЖЛИВОСТІ ВДОСКОНАЛЕННЯ СИСТЕМИ

Fund Matcher є функціональним навчальним прототипом, однак система має значний потенціал для розвитку. Основні напрями вдосконалення стосуються якості даних, продуктивності пошуку, зручності інтерфейсу, безпеки та пояснюваності результатів.

6.1. Розширення джерел даних

Найважливішим напрямом є збільшення кількості фондів і портфельних компаній. Якість matching напряму залежить від повноти бази. Доцільно додати імпорт із CSV, ручне підтвердження змін адміністратором, дедуплікацію компаній за domain name та періодичне оновлення даних.

Найважливішим напрямом розвитку є наповнення бази реальними даними про фонди та портфельні компанії. Чим повніша база, тим кориснішою стає система. Однак при імпорті потрібно враховувати якість джерел, дублікати, різні написання назв компаній і неповні описи.

Доцільно реалізувати механізм deduplication. Наприклад, одна компанія може мати різні варіанти URL або назви, але фактично бути тим самим об’єктом. Для цього можна використовувати домен сайту, нормалізовану назву, зовнішні ідентифікатори та ручну перевірку адміністратора.

6.2. Покращення пошукової логіки

Поточна модель поєднує rule score і portfolio score. У майбутньому можна додати навчання ваг на основі зворотного зв’язку користувачів: чи відкрив користувач профіль фонду, чи зберіг його, чи відправив intro message. Це дозволить поступово адаптувати ранжування до реальної поведінки засновників.

Пошукову логіку можна покращити через ваги, які залежать від типу збігу. Наприклад, збіг за lead investment може бути важливішим, ніж участь у великому раунді як minor participant. Також можна враховувати давність інвестиції: новіші угоди краще відображають актуальну стратегію фонду.

Можна також додати hybrid search, який поєднує keyword search і vector search. Це корисно для технічних термінів, назв конкретних галузей або випадків, коли exact match має велике значення. Наприклад, якщо стартап працює з “cybersecurity compliance”, точний збіг частини термінів може бути важливим поряд із семантичною близькістю.

6.3. Покращення інтерфейсу користувача

На клієнтській частині доцільно реалізувати пояснювану картку фонду: чому цей фонд рекомендовано, які портфельні компанії схожі, які критерії збіглися, які не збіглися. Такий інтерфейс підвищує довіру користувача до результатів AI-assisted пошуку.

- додати збережені списки фондів;
- додати експорт результатів у CSV або PDF;
- додати генерацію персоналізованого intro message з можливістю редагування;

- додати dashboard із історією пошукових запитів;
- додати адміністративну панель перевірки pending changes.

Інтерфейс можна розширити сторінкою порівняння фондів. Користувач міг би обрати кілька результатів і побачити їх поруч: стадії, сектори, географія, check size, схожі портфельні компанії та сформований intro message. Це зробило б систему більш корисною для реального fundraising workflow.

Ще одним напрямом є збереження списків фондів. Засновник може створювати pipeline: selected, contacted, replied, rejected, meeting scheduled. Така функція перетворює Fund Matcher із пошукового інструмента на легку CRM-систему для fundraising-процесу.

Для адміністратора варто реалізувати окрему панель якості даних. У ній можна показувати фонди без embeddings, компанії з порожніми описами, дублікати доменів, записи з низькою data quality score та pending changes, які очікують перевірки.

ВИСНОВКИ

У курсовій роботі було розглянуто процес проєктування та розробки інформаційної системи Fund Matcher для пошуку і підбору венчурних фондів. Система вирішує практичну проблему засновників стартапів: пошук релевантних інвесторів серед великої кількості фондів із різними інвестиційними мандатами, галузевими пріоритетами та географічними обмеженнями.

У роботі проаналізовано екосистему венчурного капіталу, структуру стадій фінансування від Pre-seed до Series B, проблему інформаційної асиметрії та обмеження традиційних фільтрів. Було показано, що semantic search може знаходити приховані інвестиційні інтереси фондів через аналіз їхніх портфельних компаній.

Спроєктовано структуру бази даних на PostgreSQL з використанням pgvector. Описано сутності фондів, користувачів, портфельних компаній, embeddings, джерел даних і пошукових запитів. Показано, як розділення реляційних даних і векторних представлень спрощує підтримку системи та забезпечує масштабованість.

Розглянуто серверну частину на FastAPI, SQLAlchemy та Alembic, клієнтську частину на React і TypeScript, інтеграцію з OpenAI API, алгоритм rule-based та semantic scoring, а також підходи до контейнеризації, розгортання, синхронізації з Supabase, бенчмаркінгу та якісного тестування пошукової точності.

Fund Matcher демонструє сучасний підхід до побудови інформаційної системи: поєднання класичної бази даних, API-архітектури, векторного пошуку, AI-інтеграції та зрозумілого користувацького сценарію.

Під час розробки було показано, що якість інформаційної системи залежить не лише від вибору технологічного стеку, а й від правильного розуміння предметної області. У випадку венчурного фінансування важливо враховувати стадії, сектори, географію, розмір чека, портфельну історію та неповноту відкритих даних.

Fund Matcher демонструє практичне застосування PostgreSQL і pgvector для задачі, де звичайні SQL-фільтри не дають достатньої точності. Векторний пошук дозволяє знаходити змістовно близькі компанії, а комбіноване ранжування забезпечує баланс між формальними критеріями та фактичним інвестиційним досвідом фонду.

Отриманий результат можна розглядати як повноцінний навчальний прототип інформаційної системи. Він містить основні компоненти сучасного веб-застосунку: базу даних, backend API, frontend, автентифікацію, інтеграцію із зовнішнім AI-сервісом, контейнеризоване середовище та підготовку до хмарного розгортання.

Подальший розвиток системи пов’язаний із розширенням джерел даних, покращенням UX, додаванням CRM-функцій для fundraising-процесу, оптимізацією векторного пошуку та підвищенням пояснюваності результатів. Це підтверджує, що обрана тема має не лише навчальну, а й практичну цінність.ч

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. FastAPI Documentation [Електронний ресурс] – Режим доступу: https://fastapi.tiangolo.com/
2. React Documentation [Електронний ресурс] – Режим доступу: https://react.dev/
3. TypeScript Documentation [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/docs/
4. OpenAI Platform API Reference [Електронний ресурс] – Режим доступу: https://platform.openai.com/docs/api-reference
5. OpenAI Embeddings Guide [Електронний ресурс] – Режим доступу: https://platform.openai.com/docs/guides/embeddings
6. PostgreSQL Documentation [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/
7. pgvector Documentation [Електронний ресурс] – Режим доступу: https://github.com/pgvector/pgvector
8. Supabase Documentation [Електронний ресурс] – Режим доступу: https://supabase.com/docs
9. Supabase. pgvector: Embeddings and vector similarity [Електронний ресурс] – Режим доступу: https://supabase.com/docs/guides/database/extensions/pgvector
10. SQLAlchemy Documentation [Електронний ресурс] – Режим доступу: https://docs.sqlalchemy.org/
11. Alembic Documentation [Електронний ресурс] – Режим доступу: https://alembic.sqlalchemy.org/
12. Docker Documentation [Електронний ресурс] – Режим доступу: https://docs.docker.com/
13. GitHub Docs [Електронний ресурс] – Режим доступу: https://docs.github.com/
14. Crunchbase. Company and Investor Data Platform [Електронний ресурс] – Режим доступу: https://www.crunchbase.com/
15. PitchBook. Private Market Data and Venture Capital Research [Електронний ресурс] – Режим доступу: https://pitchbook.com/

16. OpenVC. Startup Investor Database [Електронний ресурс] – Режим доступу: https://www.openvc.app/
17. Python Documentation [Електронний ресурс] – Режим доступу: https://docs.python.org/3/
18. MDN Web Docs. HTTP Overview [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
19. Kleppmann M. Designing Data–Intensive Applications / Martin Kleppmann. – Sebastopol: O'Reilly Media, 2017. – 616 p.
20. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts / Abraham Silberschatz, Henry F. Korth, S. Sudarshan. – 7th ed. – New York: McGraw–Hill Education, 2019. – 1344 p.
21. Fowler M. Patterns of Enterprise Application Architecture / Martin Fowler. – Boston: Addison–Wesley, 2002. – 560 p.
22. Russell S., Norvig P. Artificial Intelligence: A Modern Approach / Stuart Russell, Peter Norvig. – 4th ed. – Hoboken: Pearson, 2021. – 1168 p.