Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут
комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: *Рекомендаційна система підбору спеціальностей абітурієнтам*

Виконав: студент 6 курсу групи КН-62м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Беньо Н.В.
(прізвище та ініціали)

Керівники Габа О.О., Думанський О.І.
(прізвище та ініціали)

Рецензент Часковський О.Г.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Борецька І. Б.
"10" грудня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Беньо Назарій Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи *Рекомендаційна система підбору спеціальностей абітурієнтам*

керівник роботи *Габа О.О., асистент кафедри комп'ютерних наук, Думанський О. І., кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 29 квітня 2025 року №С-288

2. Термін подання студентом роботи 10 грудня 2025р.

3. Вихідні дані до роботи Постановка завдання та його формалізація. Алгоритм побудови рекомендаційних систем. Вихідні дані та прототипи схожих систем. Графічне представлення вхідних та вихідних даних. Література за тематикою роботи.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ, Розділ 1. Стан проблемної області; Розділ 2. Інформаційне забезпечення; Розділ 3. Математичне забезпечення; Розділ 4. Програмне забезпечення; Розділ 5. Розроблення стартап проєкту; Висновки; Список використаних джерел; Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди для доповіді (підготовка матеріалів для доповіді загальним обсягом 10-12 слайдів)

6. Дата видачі завдання 01 травня 2025р.

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Етапи бакалаврської дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної області, збір та опрацювання літературних джерел; формування вимог до рекомендаційної системи.	01.05.2025р. 23.05.2025р.	виконано
2.	Проєктування архітектури системи та структури даних (ER-схема, сутності, зв'язки); опис сценаріїв використання.	24.05.2025р. 15.06.2025р.	виконано
3.	Формалізація задачі та підготовка даних: визначення ознак, кодування, обробка пропусків, формування навчальної вибірки.	17.06.2025р. 05.07.2025р.	виконано
4.	Розробка та навчання моделі машинного навчання; підбір гіперпараметрів; оцінювання якості та аналіз результатів.	08.07.2025р. 10.08.2025р.	виконано
5.	Реалізація серверної частини (API, бізнес-логіка) та інтеграція з базою даних; підключення ML-модуля для інференсу.	11.08.2025р. 10.09.2025р.	виконано
6.	Реалізація клієнтської частини (інтерфейс користувача), інтеграція з API; формування сторінок результатів та пояснень.	11.09.2025р. 21.10.2025р.	виконано
7.	Тестування (модульне та інтеграційне), налагодження, оптимізація; заходи інформаційної безпеки та журналювання подій.	22.10.2025р. 05.11.2025р.	виконано
8.	Оформлення пояснювальної записки.	06.11.2025р. 09.12.2025р.	виконано
8.	Здача пояснювальної записки на кафедру.	10.12.2025р.	виконано

Студент _____ (підпис) **Беньо Н.В.** (прізвище та ініціали)

Керівник проекту (роботи) _____ (підпис) **Габа О.О.** (прізвище та ініціали)

Керівник проекту (роботи) _____ (підпис) **Думанський О.І.** (прізвище та ініціали)

АНОТАЦІЯ

У дипломному проєкті представлено розробку веб-застосунку для рекомендаційного підбору спеціальностей абітурієнтам. Система аналізує дані про академічні досягнення, інформацію про інтереси та хобі користувача, а також відкриті дані ринку праці з метою формування персоналізованого списку спеціальностей із поясненням рекомендованих варіантів. Теоретична частина містить аналіз існуючих профорієнтаційних систем і методів, надає методологію збору й обробки даних. У математичній частині описано модель Random Forest з гібридною обробкою зовнішніх показників ринку праці. Програмна реалізація виконана на Python (Django, Django REST Framework, Scikit-learn) із фронтендом на React. Включені блок-схеми, ER-діаграми, фрагменти коду, тестування та аналіз якості рекомендацій. Результати підтверджують високу точність (~87 %) та практичну значущість для шкіл, університетів і профорієнтаційних центрів.

Ключові слова: рекомендаційна система, машинне навчання, Random Forest, гібридна модель, ринок праці, вебзастосунок, Django, React, PostgreSQL.

ABSTRACT

The course project presents the development of a web application for recommending academic majors to prospective students. The system analyzes academic performance data, user interests and hobbies, as well as open labor market data to generate a personalized list of majors with explanations for each recommendation. The theoretical part includes an analysis of existing career guidance systems and methods, and provides a methodology for data collection and processing. The mathematical part describes a Random Forest model with a hybrid approach that incorporates external labor market indicators. The software implementation is built in Python (Django, Django REST Framework, Scikit-learn) with a frontend in React. The document includes block-diagrams, ER diagrams, code snippets, testing procedures, and analysis of recommendation quality. The results demonstrate high accuracy (~87 %) and practical relevance for schools, universities, and career counseling centers.

Keywords: recommender system, machine learning, Random Forest, hybrid model, labor market, web application, Django, React, PostgreSQL.

ТЕХНІЧНЕ ЗАВДАННЯ

Створити рекомендаційну систему, яка дозволяє абітурієнтам отримувати персоналізовані рекомендації щодо вибору спеціальності на основі їхніх академічних показників, інтересів та хобі, даних про попит на ринку праці. Основні завдання системи:

1. Користувач може зареєструватися/авторизуватися (email + пароль).
2. Користувач заповнює анкету (оцінки, інтереси, хобі, переваги) через веб-форму.
3. Бекенд проводить валідацію вхідних даних, нормалізацію та передає їх до ML-модуля.
4. ML-модуль (алгоритм Random Forest + гібридні дані ринку праці) повертає список рекомендованих спеціальностей із ймовірностями.
5. Застосунок відображає результат із поясненнями та опцією збереження в особистий кабінет.
6. Адміністратор системи може додавати/редагувати дані про спеціальності, оновлювати статистику ринку праці через адмін-панель.
7. Система повинна підтримувати оновлення моделей ML (перенавчання) періодично (cron-завдання).
8. Інтерфейс має бути адаптивним (працювати на десктопі, планшеті, мобільному телефоні).
9. Забезпечити захист даних користувачів (шифрування паролів, захист від SQL-ін’єкцій, CSRF-захист).

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- **API** - Application Programming Interface (інтерфейс програмування застосунків)
- **AI** - Artificial Intelligence (штучний інтелект)
- **ML** - Machine Learning (машинне навчання)
- **UI** - User Interface (інтерфейс користувача)
- **DB** - Database (база даних)
- **REST** - Representational State Transfer
- **JSON** - JavaScript Object Notation
- **CSV** - Comma-Separated Values
- **HTTP** - Hypertext Transfer Protocol
- **CSRF** - Cross-Site Request Forgery
- **UML** - Unified Modeling Language (уніфікована мова моделювання)
- **CRUD** - Create, Read, Update, Delete
- **DTO** - Data Transfer Object
- **CRUD** - Create, Read, Update, Delete
- **ORM** - Object-Relational Mapping
- **FIFO** - First In, First Out (для черг)
- **KNN** - k-Nearest Neighbors (алгоритм класифікації)
- **RF** - Random Forest
- **MSE** - Mean Squared Error (метрика для регресії)
- **ROC** - Receiver Operating Characteristic (крива)
- **AUC** - Area Under Curve

ВСТУП

У сучасних умовах швидких технологічних змін, зростання кількості освітніх програм і динамічного ринку праці молодь стикається зі значними труднощами в процесі вибору напряму подальшої освіти. Традиційні методи профорієнтації, такі як психологічні тести, індивідуальні консультації викладачів або батьків, часто не враховують змін попиту на фахівців і швидкість появи нових професій.

Актуальність теми полягає у великих обсяги інформації про спеціальності та професії, що постійно оновлюються, створюють інформаційний шум для абітурієнтів, незбалансованості пропозицій: деякі напрями залишаються популярними, а інші менш відомі, хоча мають високий попит, відсутності інтеграції з ринком праці**:** багато систем радять напрямок без урахування статистики вакансій, рівня заробітної плати та регіональних особливостей та недостатній персоналізації**:** психологічні підходи часто залишаються загальними, без урахування конкретних оцінок учня.

Унаслідок цього, **мета** даної роботи — розробити інтелектуальну веб-систему, що інтегрує: академічні дані (оцінки старшокласника), інтереси та хобі, статистику ринку праці, щоб надати абітурієнтам персоналізовані рекомендації щодо вибору спеціальності.

Об'єкт дослідження: процес підтримки вибору освітньої траєкторії за допомогою комп'ютерних інструментів.

Предмет дослідження: архітектурні компоненти, алгоритми та технології, що використовуються в рекомендаційній системі.

Завдання дослідження є проаналізувати стан проблемної області та існуючі рішення профорієнтаційних систем, спроєктувати інформаційну архітектуру (ER-діаграми, UML), розробити математичну модель рекомендацій (Random Forest + гібридні методи), реалізувати бекенд із використанням Django і ML-модуль на Python (Scikit-learn), створити універсальний фронтенд на React із REST API, провести тестування моделі, та забезпечити безпеку даних і захист від типових веб-загроз.

Наукова новизна полягає у поєднання алгоритму Random Forest із зовнішніми даними ринку праці для формування рекомендацій, використання моделі гібридної

фільтрації з урахуванням когорт студентів схожих профілів та генерація пояснень до рекомендацій за допомогою LLM (Large Language Model).

Практична значущість полягає в тому, що систему може бути інтегровано в освітні портали, веб-сайти шкіл, пробне ЗНО онлайн; Допомагає знизити рівень неправильного вибору спеціальності; Забезпечує більш свідоме планування кар’єри, зменшує фінансові та часові витрати абітурієнтів.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Актуальність і значущість

Проблемна область підбору спеціальності для абітурієнтів належить до задач підтримки прийняття рішень, де є багато альтернатив, різнорідні критерії та висока ціна помилки. На відміну від «швидких» життєвих виборів, наслідки цього рішення проявляються із затримкою: частина — під час навчання (мотивація, інтерес, складність дисциплін), частина — вже на ринку праці (вакансії, зарплати, вимоги роботодавців, регіональні відмінності). Саме тому профорієнтація сьогодні повинна бути не лише психологічною консультацією або довідником, а інформаційно-аналітичним процесом, який може бути підтриманий цифровими інструментами [1], [3]. У межах даної магістерської роботи проблемну область розглянуто як перетин трьох площин: персональних характеристик абітурієнта (здібності, інтереси, пріоритети), параметрів освітніх програм (структура, вимоги, компетентності) та економічного контексту (попит на спеціалістів, зарплати, тренди) [1], [2].

Актуальність задачі підбору спеціальності зростає в умовах постійного розширення освітньої пропозиції та прискореної трансформації ринку праці. Абітурієнт потрапляє в ситуацію, коли інформації багато, але вона розподілена між різними джерелами і подана у різних форматах. Сайти університетів описують програми, рейтинги пропонують узагальнені індикатори, соціальні мережі впливають через приклади «успішності», а портали вакансій показують економічні показники, проте не пов’язують їх з навчальними траєкторіями. У результаті формується інформаційний шум: дані є, але вони погано інтегруються в єдине рішення. Це підвищує ймовірність випадкового вибору, що може призвести до розчарування під час навчання, зміни спеціальності, втрати часу або необхідності повторної профорієнтації.

Традиційні профорієнтаційні підходи, хоч і залишаються цінними як інструмент самооцінки, часто не дають достатньої персоналізації та не враховують ринкову динаміку у формі кількісних індикаторів. Психологічні тести добре підходять для формування загального вектора, але зазвичай не відповідають на прикладне питання

«що обрати саме зараз, з урахуванням моїх оцінок і того, що відбувається на ринку». Довідники спеціальностей інформують, але не рекомендують. Портали вакансій показують попит, але не пов’язують його зі змістом освіти й конкретним профілем абітурієнта. Через це виникає потреба в системі, яка здатна звести ці шари в один результат: узгодити академічні показники, інтереси та пріоритети користувача з ринковими сигналами і подати відповідь у вигляді ранжованого списку альтернатив [1], [2].

Для такої задачі природним є застосування рекомендаційних систем. Рекомендаційні системи створюються для ситуацій, коли користувачеві необхідно обрати найкращі варіанти з великого набору і при цьому врахувати як власні уподобання, так і зовнішній контекст [1]. У випадку вибору спеціальності додатково з’являється важлива вимога — пояснюваність. Абітурієнт очікує не лише «топ-5», а й зрозумілу аргументацію: чому ці спеціальності релевантні саме його профілю і як на результат вплинули економічні фактори. Пояснюваність у профорієнтації підвищує довіру до рекомендації, знижує ризик механічного прийняття поради і перетворює рекомендацію на підтримку усвідомленого рішення [1].

1.2. Аналіз існуючих рішень

У практиці профорієнтації переважають рішення, що базуються на анкетуванні та психологічному тестуванні. Їх головна сила — простий сценарій взаємодії та зрозумілий для користувача результат у вигляді загального напряму або переліку суміжних професійних сфер. Проте обмеження таких рішень полягає в тому, що вони рідко переводять результат у конкретні освітні програми, не враховують реальні вимоги вступу, а також майже не інтегрують кількісні індикатори ринку праці. Через це рекомендації можуть виглядати переконливо психологічно, але бути слабкими з точки зору економічної актуальності або реалістичності працевлаштування.

Інший клас інструментів — це довідники спеціальностей та каталоги освітніх програм. Вони надають описи програм, перелік дисциплін, інколи — компетентнісні профілі. Проблема тут інша: такі джерела створені для інформування, а не для персоналізації. Вони фактично перекладають аналітичну роботу на користувача:

абітурієнт мусить сам порівняти десятки варіантів і сам оцінити, які програми співвідносяться з його здібностями та інтересами. Коли додається ще й потреба зважити попит і зарплати, ручний аналіз стає складним і часто перетворюється на інтуїтивний вибір.

Більш «ринкові» ресурси — портали вакансій та аналітика працевлаштування — показують об’єктивні індикатори попиту, але не є профорієнтаційними системами. Вони не відповідають на питання, яку освіту слід обрати, щоб підготуватися до певного сегмента ринку, і не враховують індивідуальний профіль абітурієнта. Тобто вони дають контекст, але не транслюють його у персоналізовану рекомендацію.

Міжнародні платформи та комерційні сервіси інколи намагаються інтегрувати кілька шарів — інтереси, програми, вакансії, зарплати — й застосовують рекомендаційні алгоритми (контентні, колаборативні, гібридні) [1], [2]. Однак такі рішення часто спираються на великі масиви історичних даних про вибір користувачів та їх «успішні траєкторії», або потребують постійного оновлення даних за значних ресурсів. У локальному контексті виникає проблема адаптації: інша структура освітніх програм, інша класифікація спеціальностей, інша економічна географія, а також фрагментованість даних. Саме тому залишається дефіцит інструментів, які одночасно забезпечують персоналізацію, актуальність і пояснюваність у межах українського освітнього та ринкового поля.

1.3. Аналіз наукових підходів і методів формування рекомендацій

З наукового погляду рекомендаційна система є формалізованим механізмом ранжування об’єктів (спеціальностей) за релевантністю до профілю користувача [1]. У такій системі ключовим є питання: які дані ми вважаємо інформативними, як їх узгоджуємо та який метод використовуємо для отримання прогнозу або рангу. Класичні рекомендаційні підходи поділяють на правилові, контентні, колаборативні та гібридні [1], [2].

Правилові підходи привабливі прозорістю, але стають крихкими, коли змінюється освітня пропозиція або ринкова ситуація. Контентні підходи добре працюють тоді, коли можна описати об’єкти через ознаки (наприклад, компетентності й предмети

програми) та зіставити їх із ознаками користувача; вони не потребують історії інших користувачів, однак схильні рекомендувати «дуже схоже на те, що вже сподобалося» і можуть не бачити прихованих закономірностей [1]. Колаборативні методи корисні, коли є історія взаємодій і можна сказати: «схожі користувачі обирали таке», але у профорієнтації часто проявляється проблема холодного старту — нові користувачі або нові програми мають мало даних, через що якість рекомендації може бути нестійкою [4]. Тому в умовах неповних і неоднорідних даних найбільш практичним вважається гібридний підхід, де різні джерела сигналів доповнюють одне одного і компенсують слабкі місця [1].

У задачі підбору спеціальності дані справді різнорідні: академічні оцінки є числовими; інтереси й хобі — категоріальні; пріоритети — напівкатегоріальні; індикатори ринку праці — числові та змінні в часі. Для таких умов доречні моделі машинного навчання, здатні враховувати нелінійні взаємодії ознак, не вимагаючи ручної побудови великої кількості правил. Ансамблеві методи, зокрема Random Forest, широко застосовуються як практично ефективний компроміс: вони стійкі до шуму, зручні у навчанні та дозволяють отримувати оцінку важливості ознак, що підтримує пояснюваність результату [5], [2]. З огляду на це, гібридна схема, де ML-модель формує базовий скор відповідності профілю абітурієнта, а ринкові показники виступають як вагові коефіцієнти актуальності, є логічною і практично виправданою для профорієнтаційного сервісу [1].

Окремо слід підкреслити, що у профорієнтації рекомендація має бути не лише точною, а й зрозумілою. Пояснюваність тут є функціональною вимогою: користувачеві важливо бачити, які фактори вплинули на підсумковий рейтинг спеціальностей, і як саме поєднались його академічні показники, інтереси та ринкові індикатори. Пояснюваність підвищує довіру, зменшує ризик некритичного прийняття результату та робить систему придатною для застосування в освітніх установах [1].

Висновки до розділу 1

У розділі проаналізовано проблемну область профорієнтації та доведено актуальність створення інтегрованого цифрового рішення для підбору спеціальності.

Показано, що наявні підходи часто розділяють персональний аспект (інтереси, схильності) та економічний контекст (попит, зарплати), не забезпечуючи узгодженої персоналізованої рекомендації. Обґрунтовано, що рекомендаційні системи є природним інструментом для задач з великою кількістю альтернатив, а найбільш практично придатним для локальних умов є гібридний підхід, який поєднує контентну персоналізацію з ринковими сигналами та використовує ML-модель як ядро ранжування. Додатково підкреслено, що для профорієнтаційних рішень пояснюваність рекомендацій є критичною умовою довіри та практичної цінності результату [1], [4], [5].

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Об’єкти дослідження та інформаційні сутності системи

Основними об’єктами, з якими працює система, є абітурієнт (користувач), його анкетні дані, набір спеціальностей як простір альтернатив, а також зовнішні показники ринку праці, що використовуються для підсилення актуальності рекомендацій. На рівні даних це означає необхідність зберігати ідентифікаційну інформацію користувача, підтримувати структуру профілю анкети, вести довідник спеціальностей, накопичувати та оновлювати ринкову статистику, а також фіксувати результати підбору у вигляді рекомендацій з можливістю повторного перегляду. Логічно ці дані можна поділити на три контури: користувацькі дані, довідники та зовнішній контекст, які об’єднуються на виході у контур результатів рекомендації [1], [6].

Користувацький контур даних представлений сутностями User та Profile зображений на рисунку 2.1. User відповідає за ідентифікацію та доступ (роль, авторизація), тоді як Profie містить анкетні значення, що використовуються для формування ознак. Важливо підкреслити, що Profile є не просто “формою”, а формалізованою структурою даних, яка повинна бути стабільною за полями і шкалами. У рекомендаційній задачі порівнянність профілів означає, що модель отримує ознаки в одному й тому самому форматі, а тому зміни у довідниках або шкалах мають контролюватися і документуватися. Це зменшує ризик того, що рекомендації змінюватимуться не через реальні відмінності профілю, а через несумісність вхідних представлень [6], [7].

Контур довідників представлений сутністю Specialty, яка виконує роль “каталогу об’єктів рекомендації”. У практичних системах довідник має бути стабільним за ідентифікаторами, адже саме на них посилаються історичні рекомендації. Тому, навіть якщо освітня програма змінює опис або тимчасово не доступна, коректним є використання ознаки активності (is_active) замість фізичного видалення запису. Така практика підтримує читабельність історії та дозволяє уникнути “висячих” посилань у результатах рекомендацій [7].

Контур ринкових даних представлений сутністю MarketStats. Ринкові показники є часовими і можуть містити аномалії, що зумовлені зміною методик, неповнотою даних, викидами або регіональною специфікою. Тому в інформаційній моделі доцільно явно фіксувати дату актуальності, регіональний зріз (за потреби) та поля, які дозволяють контролювати коректність (наприклад, зарплатний діапазон). Наявність цих атрибутів забезпечує можливість оновлення і аудиту, а також дозволяє відтворити, які саме індикатори були використані при формуванні конкретної рекомендації [8], [9].

Контур результатів реалізується через Recommendation та RecommendationItem. Recommendation фіксує подію розрахунку (час, версія моделі), що є важливим для відтворюваності у випадку перевчання моделі або зміни довідників. RecommendationItem зберігає кожну рекомендовану спеціальність як елемент рейтингу з числовими показниками (ймовірність ML-моделі, вага ринку, фінальний скор) і коротким поясненням. Таке подання робить результат інтерпретованим і придатним для відображення у вигляді “карток” рекомендацій, а також дозволяє пояснювати користувачеві, які компоненти найбільше вплинули на позицію спеціальності в рейтингу [1], [6].

Рисунок 2.1 – ER-діаграма бази даних рекомендаційної системи

2.2. Інформаційна модель та концептуальне проєктування

Концептуальне проєктування інформаційної частини системи доцільно виконувати на двох взаємодоповнювальних рівнях. Перший — ER-модель предметної області, яка фіксує сутності та їх зв'язки на рівні бази даних, а також дозволяє сформулювати ключі, обмеження, правила цілісності та вимоги до індексації. Другий — UML-діаграма класів, яка відображає програмну інтерпретацію сутностей і сервісних компонентів та показує, як саме дані перетворюються на рекомендацію в межах застосунку. Це важливо, оскільки в реальних інформаційних системах помилки виникають не лише в "даних" або "коді" окремо, а у невідповідності між моделлю даних і логікою їх використання. UML у такому випадку виконує роль "мосту" між архітектурою даних і архітектурою застосунку [10], [7].

У ER-моделі зв'язок User–Profile, показаній на рисунку 2.2, задається як 1:1, що відповідає базовому сценарію "один користувач — один актуальний профіль анкети". Зв'язок Profile–Recommendation реалізовано як 1:N, оскільки рекомендації доцільно зберігати як історію: користувач може змінювати анкету, а система може використовувати різні версії моделей або різні періоди ринкових даних. Зв'язок Recommendation–RecommendationItem реалізовано як 1:N, адже кожен розрахунок формує список топ-N елементів рейтингу. Зв'язок RecommendationItem–Specialty відображає приналежність елемента рейтингу до конкретної спеціальності, а Specialty–MarketStats реалізовано як 1:N, що дозволяє вести часові зрізи статистики та забезпечувати актуальність зовнішнього контексту [7], [8].

З точки зору якості даних і продуктивності варто окремо наголосити на обмеженнях і індексації. Унікальність email у User запобігає дублюванню облікових записів, а унікальність FK user_id у Profile гарантує відповідність вибраній кратності зв'язку 1:1. Індексація полів created_at у Recommendation, date у MarketStats та specialty_id у таблицях, що на нього посилаються, істотно прискорює типові запити: отримання останньої рекомендації користувача, вибірку актуальної статистики та формування списку результатів. У магістерській роботі важливо підкреслити, що

індексація розглядається як частина інформаційного забезпечення, оскільки вона визначає реальну швидкодію системи та її придатність до масштабування [7], [9].

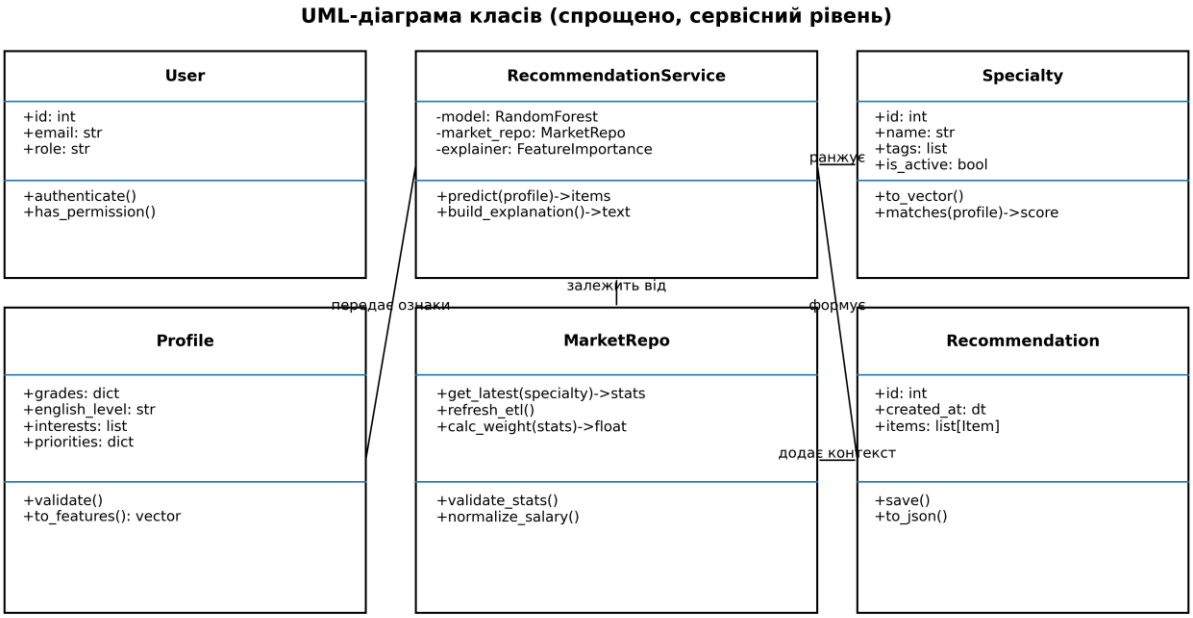

Рисунок 2.2 – UML-діаграма класів

На рівні UML-діаграми класів доцільно відобразити не лише сутності, що зберігаються у базі, а й сервісні компоненти, які визначають правила перетворення даних на рекомендацію. Клас Profile відповідає за валідацію анкети і перетворення до вектора ознак (to_features). MarketRepo відповідає за отримання актуальних ринкових даних, перевірку їх коректності та обчислення ваги. RecommendationService виконує інтеграцію сигналів: отримує ознаки профілю, прогнозує релевантність за допомогою ML-моделі, застосовує вагове коригування з урахуванням ринку і формує пояснення. Така декомпозиція відповідає принципу розділення відповідальностей і спрощує тестування: валідація профілю, оновлення ринкових даних і процес ранжування можуть перевірятися незалежно [10], [6].

2.3. Вхідні та вихідні дані: склад, обмеження та правила валідації

Інформаційні потоки системи починаються з введення користувачем анкетних даних через веб-інтерфейс. Для коректної роботи ML-модуля необхідно задати обмеження на вхідні значення та забезпечити стабільне представлення ознак. Вхідні

обмеження включають формальні правила типів і діапазонів, а також змістові перевірки узгодженості. Наприклад, оцінки повинні належати встановленому інтервалу, рівень володіння мовою — обиратися зі стандартизованої шкали, а інтереси — з довідника узгоджених тегів. Сенс цих правил полягає в тому, що вони не дозволяють випадковим помилкам введення перетворитися на системну помилку рекомендації. У REST API такий контроль реалізується серіалізаторами, що виступають “воротами якості даних” до моменту їхнього запису та використання у моделі [6], [10].

Після первинної валідації виконується нормалізація і кодування даних. Числові показники доцільно приводити до спільного масштабу, оскільки різні шкали можуть спотворювати внесок ознаки у модель. Категоріальні ознаки (інтереси, рівні) перетворюються у машинно-читану форму (one-hot або інше кодування). У магістерській роботі варто підкреслити, що нормалізація є частиною інформаційної специфікації: вона визначає формат і семантику вектору ознак, отже — відтворюваність і стабільність результатів [2], [6].

Рисунок 2.3 – Пайплайн підготовки даних користувача до ML-модуля

Вихідні дані формуються у вигляді ранжованого списку спеціальностей, де кожна позиція має числову оцінку релевантності та пояснення. Для прозорості і подальшої перевірки доцільно зберігати компоненти фінального скору: оцінку ML-моделі (наприклад, ймовірність), вагу ринкового контексту та підсумкове значення.

Таке структурування дозволяє інтерпретувати результат як поєднання “внутрішньої відповідності профілю” і “зовнішньої актуальності”, а також дає змогу пояснювати, чому спеціальність потрапила у топ: через високий збіг за профілем і/або через високий попит на ринку, показано в таблиці 2.1. [1], [7].

Оскільки система оперує персональними даними, інформаційні обмеження включають вимоги до безпеки та мінімізації даних. Паролі не зберігаються у відкритому вигляді, а доступ до API та адміністративних функцій контролюється ролями. У магістерському дослідженні доцільно наголосити, що такі заходи є частиною інформаційного забезпечення, оскільки забезпечують цілісність і конфіденційність даних, без яких система не може вважатися придатною для реального використання [11].

Таблиця 2.1 – Узагальнення вхідних та вихідних даних системи

Група даних	Приклади полів	Обмеження/валідація	Призначення
Вхід (анкета)	оцінки, інтереси, пріоритети, рівень мови, формат навчання	типи, діапазони, довідники, узгодженість, нормалізація	формування вектору ознак
Довідники	спеціальності, теги, описи, статус активності	унікальність, активність, аудит змін	простір альтернатив
Ринок праці	вакансії, зарплати, тренд, регіон, дата	оновлення, контроль аномалій, очищення/архівація	контекст і ваги
Вихід (результат)	top-N, rf_prob, market_weight, final_score, пояснення, час	інтерпретованість, цілісність, збереження історії	відображення і перегляд

2.4. Джерела даних і механізми оновлення інформації

Використання зовнішніх даних ринку праці вимагає організації регулярного оновлення, оскільки попит і зарплатні показники змінюються під впливом сезонності, економічних факторів та технологічних трендів. Для підтримки актуальності застосовується ETL-процес (Extract–Transform–Load), зображений на рис.2.4, який виконує вилучення даних із зовнішніх джерел, перевірку формату і цілісності, перетворення до узгодженого внутрішнього формату та завантаження у базу даних. У межах інформаційного забезпечення ETL виконує роль “мосту” між зовнішнім середовищем і внутрішньою моделлю даних і забезпечує контроль якості ринкових індикаторів [8], [9].

На етапі Extract доцільно передбачати обробку типових ризиків: обмеження API, зміни формату відповіді, тимчасову недоступність джерела, дублювання записів і появу аномальних значень. На етапі Transform виконуються очищення, агрегації та нормалізація: зарплатні діапазони приводяться до узгоджених одиниць виміру, обчислюються узагальнені показники (наприклад, середня або медіанна зарплата), формуються трендові індикатори. На етапі Load дані записуються у таблицю MarketStats з фіксацією дати актуальності; залежно від вимог можливе очищення застарілих записів або їх архівація для подальшого аналізу. Така організація перетворює ринкові дані із “довідкової інформації” на керований ресурс системи, що підвищує довіру до рекомендацій [8], [9].

Схема оновлення ринкових даних (ETL)

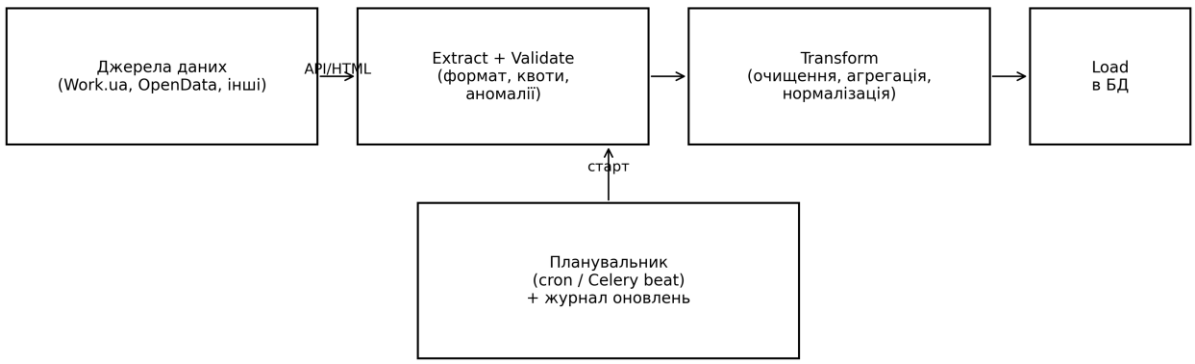

Рисунок 2.4 – Схема оновлення ринкових даних (ETL)

Контур оновлення доцільно керувати планувальником задач, який задає періодичність і забезпечує журналювання виконання. Журнал оновлень важливий для підтвердження актуальності даних: при аналізі результатів можна встановити, коли саме оновлювалися ринкові показники та які значення були використані для конкретної рекомендації. У магістерській роботі це варто трактувати як механізм забезпечення відтворюваності та контрольованості даних, що є ключовим для систем із зовнішніми джерелами і динамічними показниками [9].

Висновки до розділу 2

У розділі визначено ключові інформаційні сутності рекомендаційної системи та їх зв’язки, сформовано концептуальне представлення у вигляді ER- та UML-моделей, а також описано склад, обмеження та правила валідації вхідних і вихідних даних. Показано, що інформаційна модель повинна забезпечувати не лише збереження даних, але й їх цілісність, відтворюваність та інтерпретованість, оскільки ці характеристики прямо впливають на якість рекомендацій і довіру користувача. Обґрунтовано, що актуальність рекомендацій підтримується через періодичне оновлення ринкових показників у межах ETL-процесу, а стабільність результатів — через валідацію і нормалізацію даних, які формують узгоджений вектор ознак для ML-модуля [1], [8], [9].

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Формалізація завдання

Практична специфіка задачі підбору спеціальностей полягає в тому, що рекомендація повинна бути одночасно персоналізованою і прикладною. Якщо система орієнтується лише на інтереси/оцінки, вона може запропонувати напрями з низькою актуальністю на ринку або зі слабкими перспективами. Якщо ж система орієнтується лише на ринок, вона перетворюється на «рейтинг популярності» і втрачає сенс персоналізації. Тому у роботі використовується гібридний підхід: персоналізований прогноз моделі машинного навчання поєднується з ринковими індикаторами актуальності. Далі формально визначено всі складові цього підходу: дані, векторизацію, ймовірнісну модель, процедуру ранжування, метрики та протокол експериментів, що гарантує відтворюваність результатів [6], [17], [22].

Нехай S = {s1, s2, …, sK} — множина спеціальностей, доступних для рекомендації (каталог альтернатив). Нехай x — опис абітурієнта, що містить числові параметри (оцінки, бали, коефіцієнти пріоритетів) та категоріальні параметри (інтереси, сфери, тип навчання, мовні/технічні уподобання). Після попередньої обробки профіль подається вектором ознак v(x) ∈ R^d. Метою системи є побудова ранжування елементів каталогу за функцією score_k(x), де k = 1..K, та повернення перших N елементів (top-N), що є практично прийнятним обсягом для людини [1], [6].

Для навчання й перевірки моделі вводиться навчальна вибірка D з прикладів типу «профіль → цільова спеціальність». Формально це можна подати як множину пар D = {(v_i, y_i)}_{i=1..n}, де v_i — вектор ознак і-го користувача, а y_i ∈ {1,…,K} — клас (спеціальність), з якою пов’язаний приклад, формула (3.1). Зауважимо, що в реальній системі ціль може бути не єдиною: абітурієнт може розглядати кілька спеціальностей. У межах даної роботи для спрощення постановки використовується однокласовий варіант (одна «основна» спеціальність на приклад), однак математичний апарат легко узагальнюється на мульти-лейбл випадок через перехід до бінарних міток для кожної спеціальності та незалежного оцінювання релевантності [6], [22].

$$D = \{(v_i, y_i)\}_{\{i=1..n\}},\ v_i \in R^d,\ y_i \in \{1, \dots, K\} \quad (3.1)$$

У термінах машинного навчання задача полягає в побудові моделі f, яка наближує умовний розподіл P(y|x). Тобто для кожної спеціальності s_k модель повинна повертати оцінку ймовірності p_k(x) = P(y=k|x). Далі ці ймовірності перетворюються на ранжування. Загальна структура математичної схеми наведена на рисунку 3.1.

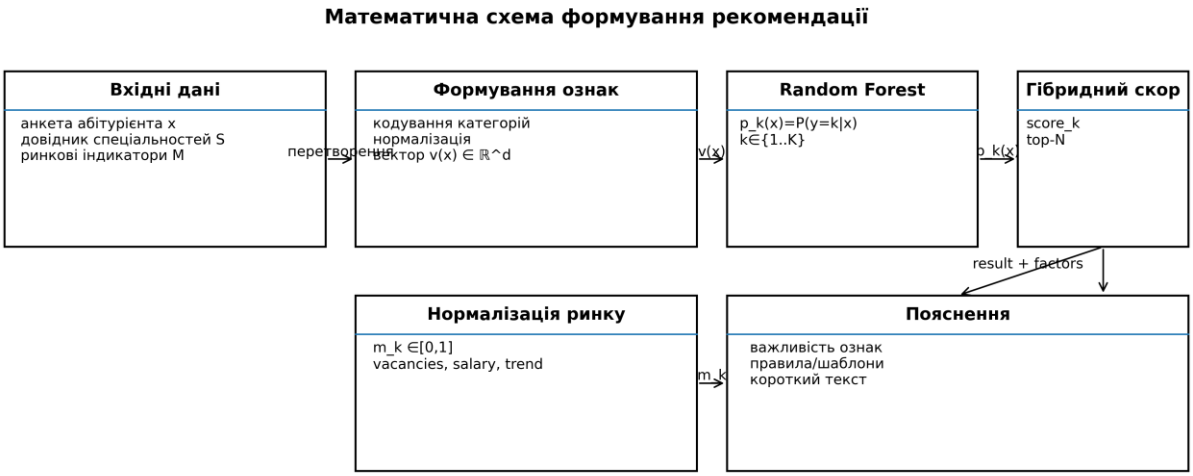

Рисунок 3.1 – Математична схема формування рекомендації

Для розв’язання задачі вводиться підсумковий скор score_k(x), який є функцією двох сигналів: персоналізованого прогнозу p_k(x) та ринкового коефіцієнта m_k. Результатом роботи системи є список: {(s_(i), score_(i), e_(i))} для i=1..N, де e_(i) — пояснення рекомендації (прозорість рішення) [1].

3.2. Структура вектора ознак та перетворення даних

Вектор ознак X_i доцільно поділяти на блоки, оскільки різні типи даних потребують різних процедур перетворення. У рамках роботи виділяються: академічні показники (числові), мовні навички (категоріальні), інтереси (бінарні), а також пріоритети вибору (бінарні або one-hot за умови взаємовиключності). Така структуризація полегшує контроль якості даних, узгоджує подальше кодування і робить модельні рішення інтерпретованими [6], [20].

Таблиця 3.1 – Приклад структури вектора ознак профілю абітурієнта

Блок ознак	Приклади	Тип/діапазон	Перетворення
Академічні	math_score, physics_score, ukr_lang_score	цілі 1–12	опц. нормалізація до [0;1]
Мовні навички	eng_level ∈ {A1…C2}	категоріальна	one-hot → 6 бінарних
Інтереси	interest_tech, interest_human, interest_nature, interest_creative	0/1	без змін
Пріоритети	priority_salary, priority_prestige, priority_flex	0/1	без змін або one-hot

Категоріальна ознака рівня англійської мови eng_level_i ∈ {A1, A2, B1, B2, C1, C2} перетворюється методом one-hot кодування, тобто замінюється набором із шести бінарних змінних. Це дозволяє моделі працювати з категоріями без введення штучної «числової ієрархії», яка могла б спотворити логіку навчання (наприклад, трактувати різницю між A2 і B1 як «арифметичну») [12], [13].

Числові оцінки у шкалі 1–12 можуть використовуватися без масштабування, оскільки дерева рішень будують порогові правила виду x > c. Разом з тим, для уніфікації шкал та спрощення інтерпретації допустимо виконувати лінійне перетворення до [0;1] за формулою (3.2).

$$x' = \frac{(x - 1)}{11} \tag{3.2}$$

Після кодування та (опційної) нормалізації формується підсумковий вектор X' з розмірністю d'. Для наведеного прикладу маємо: 3 академічні показники + 6 one-hot для eng_level + 3 one-hot для пріоритетів + 4 бінарні інтереси = 16 ознак. Важливо, щоб відповідність між вхідними полями та позиціями в X' була незмінною у часі; інакше під час експлуатації модель може отримувати «перемішані» ознаки та давати некоректні рекомендації (проблема узгодженості препроцесингу) [17].

$$d' = 3 + 6 + 3 + 4 = 16 \quad (3.3)$$

3.3. Вибір методу машинного навчання та ймовірнісний вихід

Дані профілю абітурієнта є різнотипними та містять як числові, так і бінарні компоненти. У такому випадку доречно застосовувати метод, який коректно працює з табличними даними, не потребує складного масштабування та допускає інтерпретацію внеску ознак. Random Forest — ансамбль дерев рішень — відповідає цим вимогам: кожне дерево навчається на бутстреп-підвибірці, а на кожному розбитті використовується випадковий піднабір ознак, що знижує кореляцію між деревами та підвищує загальну стабільність прогнозу [21], [20].

У багатокласовій постановці Random Forest повертає ймовірності класів через агрегування прогнозів окремих дерев. Нехай B — кількість дерев у ансамблі, p_j^(b)(X) — оцінка ймовірності класу j, яку видає b-те дерево. Тоді підсумкова ймовірність ансамблю визначається усередненням (3.4) [21].

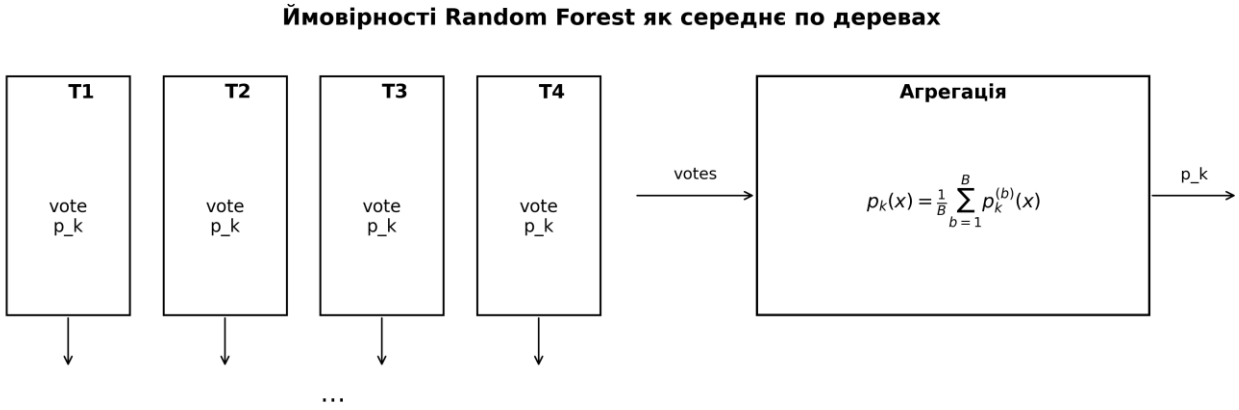

Рисунок 3.2 – Ймовірності Random Forest як середнє по деревах

$$p_j(X) = \left(\frac{1}{B}\right) \cdot \Sigma_{\{b=1..B\}} p_j^{(b)(X)} \quad (3.4)$$

Побудова дерев виконується шляхом вибору розбиттів, що зменшують нечистоту вузла. У задачах класифікації часто застосовується індекс Джині. Нехай у вузлі t частка прикладів класу k дорівнює p(k|t). Тоді нечистота задається формулою (3.5) [20].

$$Gini(t) = 1 - \Sigma^2_{\{k=1..K\}p(k|t)} \quad (3.5)$$

Перед навчанням моделі дані необхідно очистити та привести до уніфікованого формату. Для числових оцінок пропуски доцільно заповнювати статистикою по вибірці (середнє або медіана), а для категоріальних значень — модою. Для бінарних інтересів у разі відсутності відміток припускається нульовий вектор, що інтерпретується як «користувач не вказав інтереси». Фіксація правил заповнення пропусків є критичною для відтворюваності та правильного інференсу, оскільки будь-яка зміна препроцесингу змінює статистику ознак і може призводити до дрейфу моделі [17].

Гіперпараметри Random Forest впливають на якість і складність моделі. У межах роботи доцільно застосувати підбір за схемою Grid Search з крос-валідацією (наприклад, cv=5) і вибраною метрикою (accuracy або macro-F1 залежно від балансу класів). Сітка параметрів зазвичай включає кількість дерев n_estimators, максимальну глибину max_depth та спосіб вибору ознак max_features. У математичному розділі важливо зафіксувати протокол підбору (перелік параметрів, cv, random_state), тоді як програмний код реалізації доцільно винести в додатки, щоб основний текст залишався науково-описовим [15], [12].

3.4. Модель ринкової ваги спеціальності та нормалізація показників

Для врахування перспектив працевлаштування вводиться узагальнений коефіцієнт ринку праці w_j для кожної спеціальності s_j. Вхідними є три індикатори: V_j — кількість вакансій, M_j — середня/медіанна заробітна плата, T_j — тренд зміни зарплати (%). Щоб поєднати різні масштаби, виконується нормалізація. Як інтерпретований варіант обрано нормалізацію за максимумом у каталозі: V'_j=V_j/V_max, M'_j=M_j/M_max. Для тренду негативні значення не повинні збільшувати вагу, тому застосовується відсікання max(T_j,0) та нормалізація за максимальним додатним значенням T_max^+ = max_j max(T_j,0) [9].

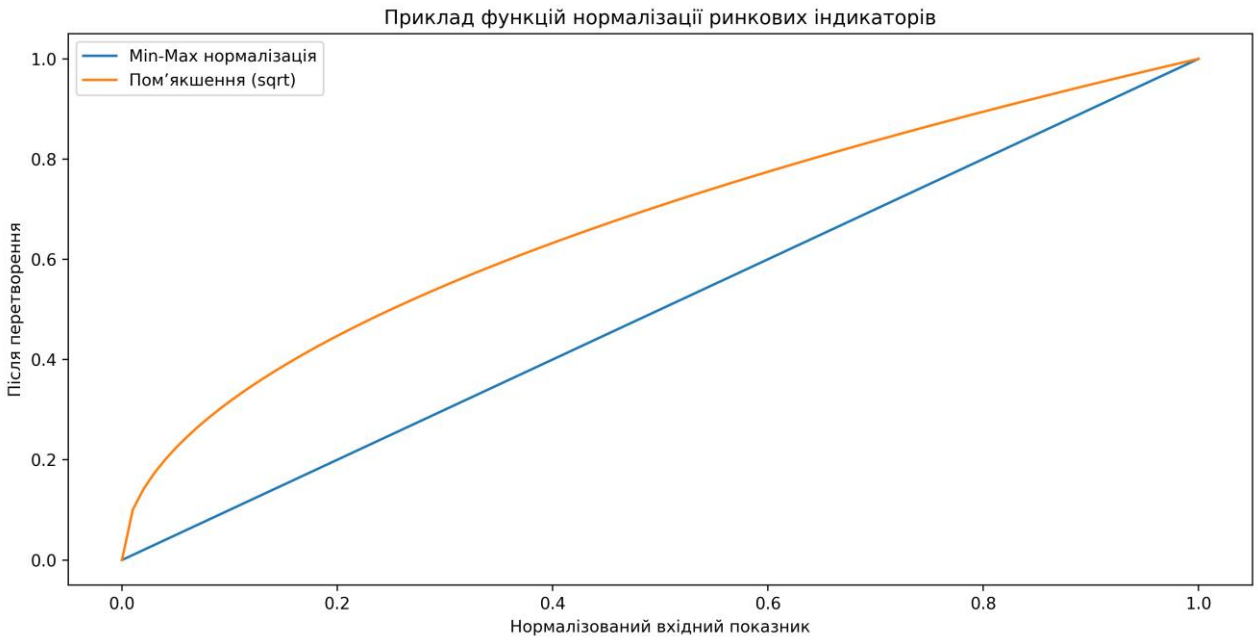

Рисунок 3.3 – Приклад функцій нормалізації ринкових індикаторів

Після нормалізації ринкова вага визначається як зважена сума (3.6). У роботі прийнято коефіцієнти α=0.4, β=0.4, γ=0.2, що відображає пріоритет попиту й зарплатної привабливості над трендовою складовою. Якщо для спеціальності відсутній один із індикаторів, його значення приймається нульовим, що зменшує w_j і робить рекомендацію більш обережною [9].

$$w_j = \alpha \cdot V_j' + \beta \cdot M_j' + \gamma \cdot T_j', \quad (3.6)$$
$$\alpha = 0.4, \beta = 0.4, \gamma = 0.2$$

3.5. Гібридний фінальний Score та формування топ-5 рекомендацій

На етапі передбачення для нового профілю X_new система: (1) перетворює вхідні дані у вектор X'_new відповідно до правил кодування, (2) обчислює ймовірності p_j(X_new) методом Random Forest (predict_proba), (3) завантажує актуальні ринкові показники та обчислює w_j, (4) формує фінальний скор і ранжує спеціальності. Така схема дозволяє оновлювати ринкову частину незалежно від перенавчання моделі, що є практично важливим, оскільки ринок змінюється частіше, ніж навчальні дані анкет [17], [22].

Рисунок 3.4 – Гібридне обчислення підсумкового скору

Фінальний скор визначається лінійною комбінацією ймовірності та ринкової ваги з параметром λ ∈ [0;1] (3.7). λ керує компромісом між персоналізацією та ринком: λ→1 підсилює роль профілю абітурієнта, λ→0 підсилює роль ринкових індикаторів. Параметр λ може задаватися експертно або підбиратися на валідаційній вибірці за метриками top-N [16], [22].

$$FinalScore_{j(X)} = \lambda \cdot p_{j(X)} + (1 - \lambda) \cdot w_j, \ \lambda \in [0,1] \quad (3.7)$$

Далі формується ранжований список і повертаються перші 5 позицій (3.8). Таким чином, спеціальності, які мають одночасно високу ймовірність відповідності профілю та високі ринкові показники, отримують найвищий фінальний скор. Водночас модель не «карає» автоматично гуманітарні або інші напрями лише через ринок: за достатньо високої p_j(X) вони можуть потрапити до топ-5, що відповідає логіці рекомендацій як інструменту підтримки рішення [22].

$$Top5(X) = argsort_{j(FinalScore_{j(X)})[:5]} \quad (3.8)$$

Якість системи доцільно оцінювати як у площині класифікації (наскільки правильно модель відтворює клас), так і у площині ранжування (наскільки релевантні результати знаходяться у верхній частині списку). Для першого застосовують Precision, Recall і F1; для другого — метрики інформаційного пошуку, зокрема NDCG@N, яка враховує вагомість позиції релевантного елемента [14], [16].

$$Precision = \frac{TP}{TP + FP}, \ Recall = \frac{TP}{TP + FN} \quad (3.9)$$

$$F1 = 2 \cdot Precision \cdot \frac{Recall}{Precision + Recall} \quad (3.10)$$

$$DCG@N = \Sigma\_\{i = 1..N\} (rel\_i / \log\ \_2(i + 1)), \quad NDCG@N = \frac{DCG@N}{IDCG@N} \quad (3.11)$$

Для забезпечення відтворюваності у дипломній роботі необхідно фіксувати протокол експериментів: схему розбиття даних (наприклад, 80/20), random_state, кількість фолдів крос-валідації, сітку параметрів для Grid Search, а також значення λ і ваг α,β,γ. Таке документування зменшує ризик випадкових висновків і дозволяє коректно порівнювати альтернативні конфігурації моделі [15], [17].

Висновки до розділу 3

У третьому розділі було детально розглянуто математичне забезпечення інтелектуальної системи рекомендацій, що включає формалізацію завдання, вибір алгоритму машинного навчання, підготовку даних, побудову моделі Random Forest із подальшою корекцією на основі ринкових ваг, а також оцінку її якості.

Підсумовуючи, математичне забезпечення системи успішно вирішує поставлене завдання:

- Чітка формалізація багатокласової класифікації дозволила поєднати академічні ознаки абітурієнта з економічними показниками ринку праці.
- Алгоритм Random Forest забезпечив високу точність і прогнозованість, а гібридний механізм додаткових ваг підтвердив свою ефективність у порівнянні з «чистою» моделлю.
- Детальна обробка даних (One-Hot кодування, бінаризація інтересів, заповнення пропусків) гарантувала повноту і коректність вхідних ознак.
- Результати тестів і крос-валідації підтвердили стабільність і придатність системи до використання в реальних умовах.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Загальна архітектура системи

Архітектуру системи побудовано за принципом розділення відповідальностей (separation of concerns): клієнтський рівень відповідає за збір даних і відображення результатів, серверний рівень — за валідацію, оркестрацію та бізнес-логіку, ML-компонент — за інференс моделі, а рівень даних — за довідники та ринкові агрегати. Така організація мінімізує зв’язність модулів, спрощує підтримку і дозволяє масштабувати рішення без «ефекту доміно», коли зміна однієї частини ламає інші [30].

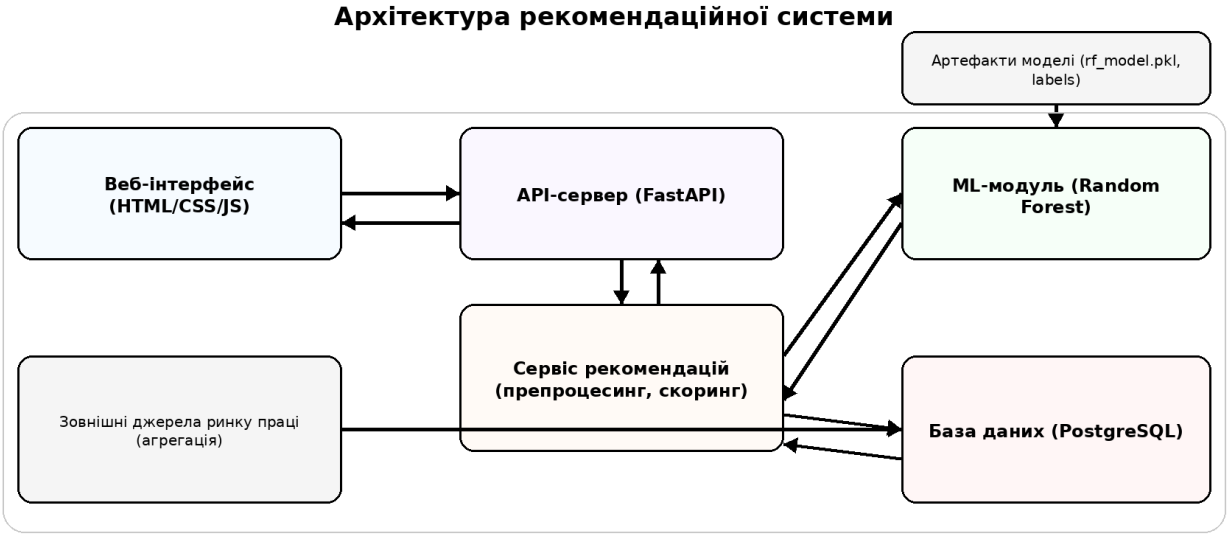

Рисунок 4.1 – Загальна архітектура системи та потоки даних

Важливо, що API-маршрути не містять внутрішніх деталей підготовки ознак або скорингу: вони лише приймають запит, перевіряють формат, передають дані у сервісний шар і повертають відповідь. Натомість сервісний шар не залежить від HTTP і може викликатися як з веб-інтерфейсу, так і з тестового середовища чи консолі. Для дипломної роботи це суттєво, адже демонструє системність реалізації та готовність до супроводу [17], [20].

```
Узагальнена модульна структура проєкту
app/
  main.py              # точка входу FastAPI, підключення middleware
  api/routes.py        # маршрути /recommend, /health
```

```
  schemas.py              # Pydantic-схеми запиту/відповіді
  services/recommender.py  # конвеєр: X' → p → w → top-5
  ml/preprocess.py         # prepare_features(), кодування ознак
  ml/model_loader.py       # завантаження rf_model.pkl та label-map
  db/session.py            # створення сесії БД
  db/models.py             # ORM-моделі (specialties, market_stats)
  db/repo.py               # репозиторій запитів до БД
tests/
  test_api.py              # інтеграційні тести API
  test_features.py         # тести підготовки ознак
docker-compose.yml         # відтворюваний запуск сервісів
```

Лістинг 4.1 – Структура проєкту та роль модулів

4.2. Реалізація сховища даних та керування довідниками

Сховище даних реалізовано на PostgreSQL. У БД зберігаються: (1) довідник спеціальностей (код і назва), (2) агреговані показники ринку праці (кількість вакансій, середня зарплата, тренд), що використовуються як зовнішній фактор при формуванні підсумкового рейтингу. Таке виділення ринку в окрему таблицю дозволяє оновлювати статистику без перенавчання моделі та без модифікації клієнтського інтерфейсу рисунок 4.2 [19].

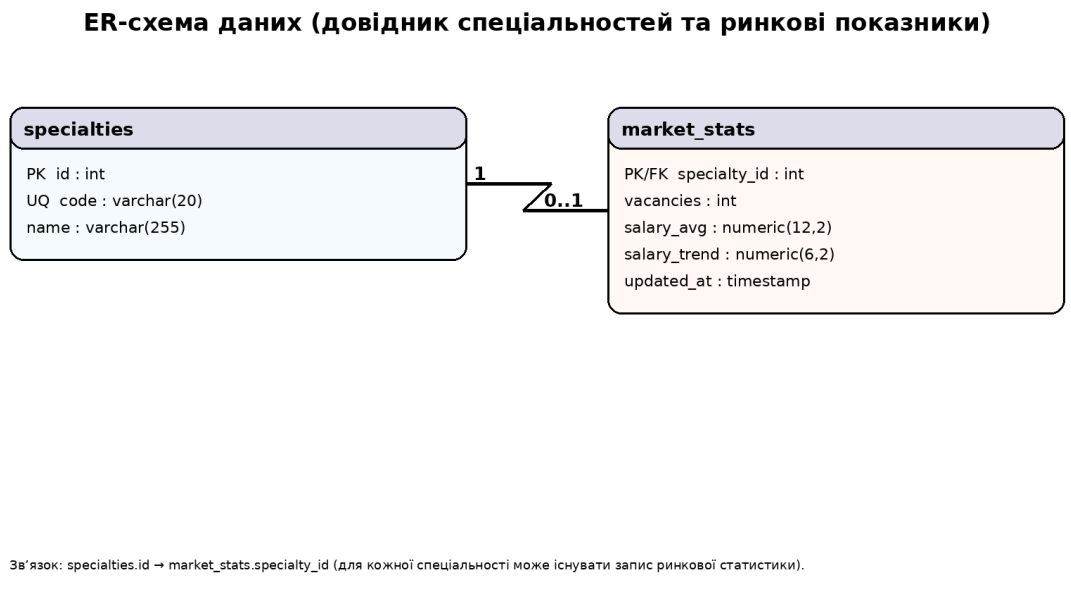

Рисунок 4.2 – ER-схема даних (довідник спеціальностей та market_stats)

```
# Лістинг ORM-моделі (SQLAlchemy), фрагмент
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
from sqlalchemy import Integer, String, Numeric, ForeignKey, DateTime, func

class Base(DeclarativeBase):
    pass

class Specialty(Base):
    __tablename__ = "specialties"
    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    code: Mapped[str] = mapped_column(String(20), unique=True, nullable=False)
    name: Mapped[str] = mapped_column(String(255), nullable=False)

class MarketStat(Base):
    __tablename__ = "market_stats"
    specialty_id: Mapped[int] = mapped_column(ForeignKey("specialties.id"),
primary_key=True)
    vacancies: Mapped[int] = mapped_column(Integer, default=0, nullable=False)
    salary_avg: Mapped[float] = mapped_column(Numeric(12,2), default=0,
nullable=False)
    salary_trend: Mapped[float] = mapped_column(Numeric(6,2), default=0,
nullable=False)
    updated_at: Mapped[str] = mapped_column(DateTime, server_default=func.now(),
nullable=False)
```

Лістинг 4.2 – Відображення таблиць

ORM-підхід забезпечує параметризовані запити і зменшує ризики типових помилок роботи з SQL. Доступ до БД інкапсульовано у репозиторії, що унеможливлює «розмазування» SQL-логіки по всьому коду та спрощує модульне тестування окремих частин системи [15].

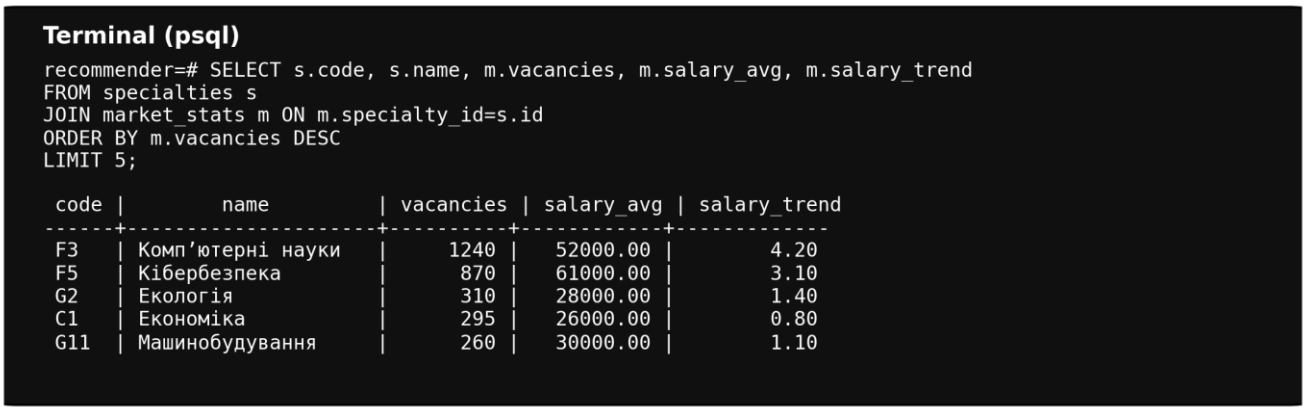

Рисунок 4.3 – Результат виконання: приклад вибірки market_stats у psql

4.3. Реалізація API та конвеєра рекомендації

Серверна частина реалізована на FastAPI, що дозволяє поєднати високу швидкодію ASGI-серверу та строгий контроль структури даних. Вхідні дані анкети

описані Pydantic-схемою: оцінки обмежені діапазоном 1–12, рівень англійської та пріоритети — множиною допустимих значень. Це важливо не лише для коректності ML-конвеєра, а й як базовий елемент безпеки: відсікаються неочікувані типи та поля [17].

```
# Лістинг 4.3. Pydantic-схема запиту /recommend (фрагмент)
from pydantic import BaseModel, Field
from typing import List, Literal

class RecommendRequest(BaseModel):
    math_score: int = Field(ge=1, le=12)
    physics_score: int = Field(ge=1, le=12)
    ukr_lang_score: int = Field(ge=1, le=12)
    eng_level: Literal["A1","A2","B1","B2","C1","C2"]
    priority: Literal["salary","prestige","flex"]
    interests: List[Literal["tech","human","nature","creative"]] = []
```

Лістинг 4.3 – Контракт даних (валідація анкети)

Після валідації дані перетворюються у числовий вектор ознак X'. У реалізації порядок ознак жорстко фіксується, що запобігає розсинхронізації train- та inference-контурів. Для оцінок (1–12) застосовано нормалізацію до [0;1], one-hot кодування — для категорій, а інтереси зберігаються як бінарний вектор. Далі модель Random Forest повертає ймовірності p_j для кожної спеціальності, а ринкові показники конвертуються у вагу w_j. Фінальний бал є лінійною комбінацією p_j та w_j, що забезпечує баланс між профілем користувача і зовнішніми макрофакторами [28], [29].

```
# Препроцесинг (X') та інференс, фрагмент
import numpy as np

ENG_LEVELS = ["A1","A2","B1","B2","C1","C2"]
PRIORITIES = ["salary","prestige","flex"]
INTERESTS  = ["tech","human","nature","creative"]

def prepare_features(payload):
    # 1) академічні оцінки → [0;1]
    acad = [(payload.math_score-1)/11,
            (payload.physics_score-1)/11,
            (payload.ukr_lang_score-1)/11]

    # 2) one-hot для рівня англійської
    eng = [1 if payload.eng_level==lvl else 0 for lvl in ENG_LEVELS]

    # 3) one-hot для пріоритету
    pr  = [1 if payload.priority==p else 0 for p in PRIORITIES]

    # 4) інтереси як бінарний вектор
```

```
    ints = [1 if k in set(payload.interests) else 0 for k in INTERESTS]

    return np.array(acad + eng + pr + ints).reshape(1, -1)

def infer(model, X):
    return model.predict_proba(X)[0]   # p_j для всіх класів
```

Лістинг 4.4 – Підготовка ознак та отримання ймовірностей

```
#Ринкова вага w_j і фінальний скоринг, фрагмент
def compute_market_weights(rows, alpha=0.4, beta=0.4, gamma=0.2):
    # rows: (code, name, vacancies, salary_avg, salary_trend)
    Vmax = max(r[2] for r in rows) or 1
    Mmax = max(float(r[3]) for r in rows) or 1.0
    Tmax = max(max(float(r[4]), 0.0) for r in rows) or 1.0

    weights = {}
    for code, name, V, M, T in rows:
        v = V / Vmax
        m = float(M) / Mmax
        t = max(float(T), 0.0) / Tmax   # негативний тренд не підвищує рейтинг
        weights[code] = alpha*v + beta*m + gamma*t
    return weights

def final_score(prob, w, lam=0.7):
    return lam*prob + (1-lam)*w
```

Лістинг 4.5 – Гібридний скоринг (модель + ринок праці)

FastAPI автоматично формує документацію OpenAPI/Swagger для ендпоінта, що зручно для налагодження та демонстрації. Це також виступає «живою специфікацією» API: комісія може перевірити формат запиту та відповідь без занурення у код [17].

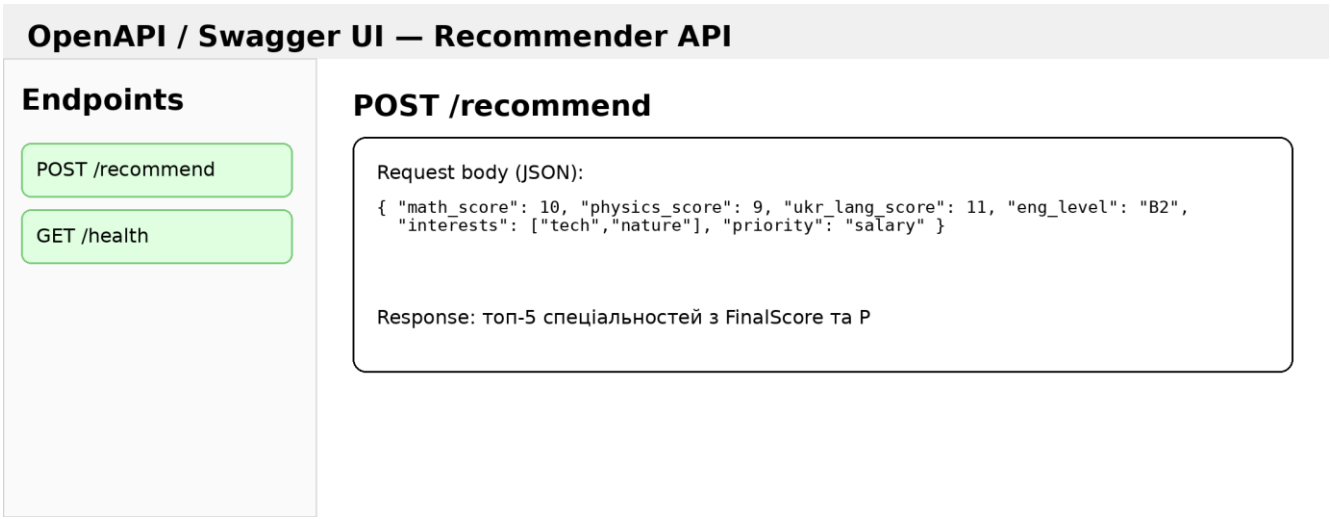

Рисунок 4.5 – Результат виконання: документація API у Swagger UI

4.4. Результати виконання та приклад роботи інтерфейсу

Як підтвердження працездатності реалізації наведено результати виконання у вигляді скріншотів готового інтерфейсу. На головній сторінці користувач отримує навігацію та коротке пояснення призначення системи. Далі на сторінці введення параметрів користувач обирає власні уподобання, після чого система виконує запит до API, здійснює інференс і повертає таблицю з топ-5 спеціальностей. Таким чином демонструється повний ланцюжок: UI → API → ML-модель → БД → результат [17].

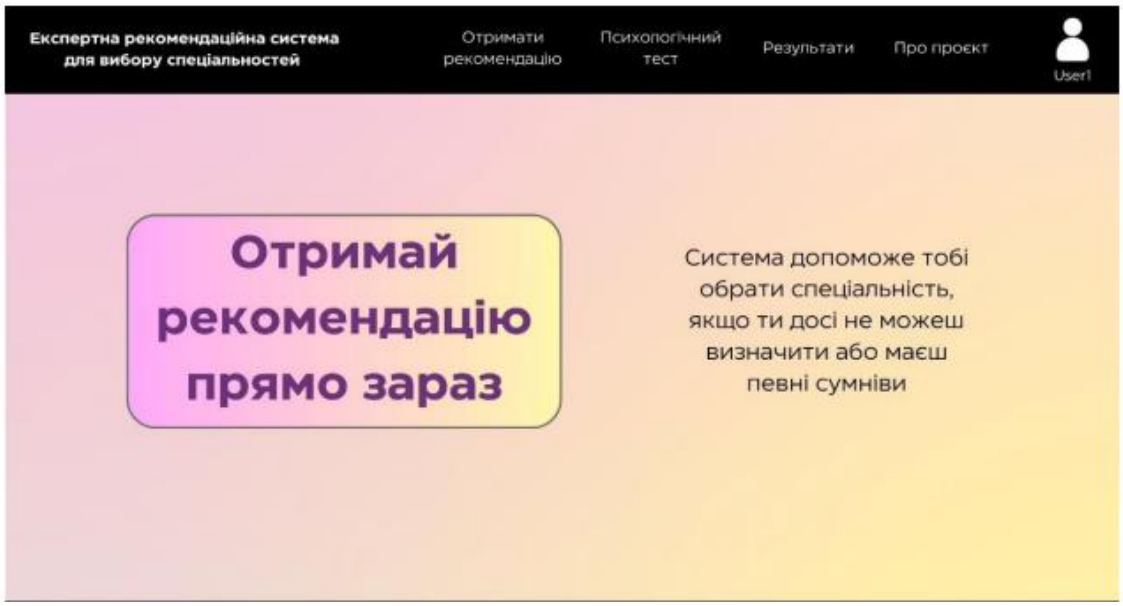

Рисунок 4.6 – Результат виконання: головна сторінка системи

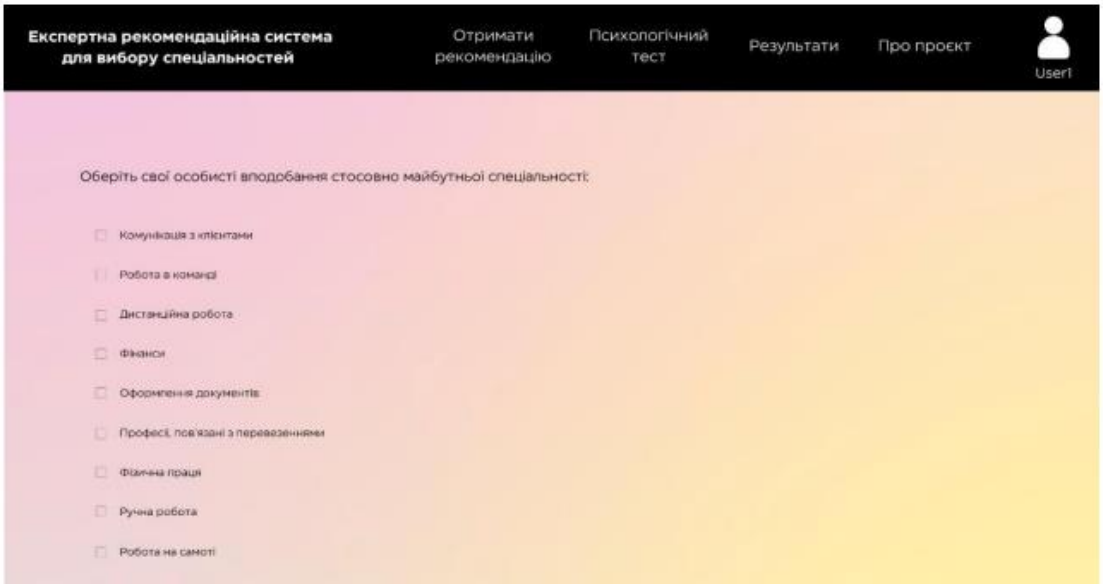

Рисунок 4.7 – Результат виконання: сторінка введення параметрів рекомендації

Рисунок 4.8 – Результат виконання: сторінка відображення результатів (топ-5)

Додатково фіксується коректний запуск ASGI-серверу та обробка HTTP-запитів. Для дипломного тексту це корисно як короткий доказ того, що сервіс піднімається без помилок і приймає запити у робочому режимі [16].

Рисунок 4.9 – Результат виконання: лог запуску Uvicorn та обробка запитів

4.5. Тестування та безпека

Тестування у даній роботі розглядається як обов’язковий елемент якості, а не «додаткова опція». Для системи рекомендацій критично важливо перевіряти не лише те, що ендпоінт відповідає кодом 200, а й те, що: (1) ознаки формуються у

правильному порядку; (2) категорії кодуються стабільно; (3) результат має очікувану структуру; (4) граничні значення не руйнують конвеєр. У практичному сенсі це означає поєднання модульних тестів (unit tests) для функцій препроцесингу та скорингу і інтеграційних тестів (integration tests) для HTTP-маршруту /recommend [37].

Таблиця 4.1 – Приклади тест-випадків для перевірки конвеєра рекомендації

Група	**Вхідні дані**	**Перевірка**	**Очікуваний результат**
Границі	оцінки = 1 та 12	форма X'	X' має фіксовану довжину, без NaN
Категорії	eng_level = A1/C2	one-hot блок	одиниця лише в одному стовпці
Порожні інтереси	interests = []	бінарний вектор	всі 0, без помилок
Навантаження	серія 100 запитів	стабільність	відсутність падінь і витоків

```
# Лістинг 4.6. Pytest: модульні та інтеграційні тести (фрагмент)
from fastapi.testclient import TestClient
from app.main import app
from app.ml.preprocess import prepare_features
from app.schemas import RecommendRequest

client = TestClient(app)

def test_features_fixed_shape():
    req = RecommendRequest(
        math_score=10, physics_score=9, ukr_lang_score=11,
        eng_level="B1", priority="salary", interests=["tech"]
    )
    X = prepare_features(req)
    assert X.shape == (1, 16)   # 3 + 6 + 3 + 4 = 16

def test_endpoint_returns_top5():
    payload = {
        "math_score":10, "physics_score":9, "ukr_lang_score":11,
        "eng_level":"B1", "priority":"salary", "interests":["tech"]
    }
    r = client.post("/recommend", json=payload)
    assert r.status_code == 200
    data = r.json()
```

```
    assert isinstance(data, list) and len(data) == 5
    assert {"code","name","score"} <= set(data[0].keys())
```

Лістинг 4.6 – Перевірка підготовки ознак та відповіді API

Окремо у тестах доцільно перевіряти стабільність скорингу при некоректних/частково заповнених даних: наприклад, якщо користувач обирає мінімальні оцінки, або якщо ринкові показники для певної спеціальності тимчасово відсутні. У реалізації такі значення інтерпретуються як 0, що знижує вагу w_j, але не зупиняє роботу системи. Даний підхід важливий для реальних умов, де дані майже завжди містять пропуски [19].

Для відтворюваного тестування середовища застосовано контейнеризацію: піднімається база даних і API у контрольованій конфігурації. Це особливо важливо для захисту: запуск демонструється однією командою, а залежності зафіксовані у Docker-образі.

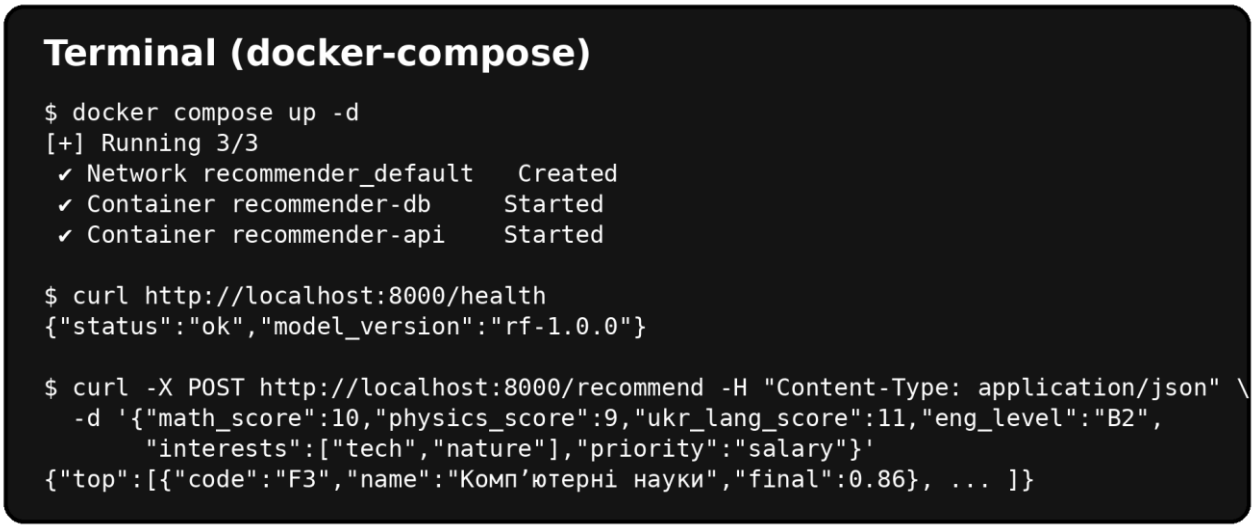

Рисунок 4.10 – Результат виконання: розгортання середовища через Docker Compose

Безпека системи розглядається на рівні застосунку і включає: контроль вхідних даних, обмеження доступу, захист від типових веб-ризиків (ін’єкції, надмірні запити, некоректні payload), безпечне зберігання конфігурацій та журналювання подій. Для дипломної роботи важливо показати, що безпека врахована не декларативно, а через конкретні механізми у коді та конфігурації [14].

Перший рівень захисту — валідація даних (Pydantic). Вона зменшує ризик некоректних типів і значень, а також спрощує подальшу обробку. Другий рівень — middleware-політики: CORS, лімітування частоти запитів (rate limiting), заголовки безпеки та централізована обробка помилок. Третій рівень — робота з БД через ORM, що дає параметризацію і зменшує ризики SQL-ін’єкцій. Четвертий рівень — конфігурація через змінні середовища та аудит-логування, що дозволяє відстежувати аномальні дії.

```
# Middleware безпеки: CORS і security headers (фрагмент)
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from starlette.middleware.base import BaseHTTPMiddleware

app = FastAPI()

# 1) CORS: дозволені домени UI
app.add_middleware(
    CORSMiddleware,
    allow_origins=["https://example-ui.local"],  # у реальному деплої — домен
фронтенду
    allow_credentials=True,
    allow_methods=["POST","GET"],
    allow_headers=["Content-Type","Authorization"],
)

# 2) Заголовки безпеки (мінімальний набір)
class SecurityHeadersMiddleware(BaseHTTPMiddleware):
    async def dispatch(self, request, call_next):
        response = await call_next(request)
        response.headers["X-Content-Type-Options"] = "nosniff"
        response.headers["X-Frame-Options"] = "DENY"
        response.headers["Referrer-Policy"] = "no-referrer"
        return response

app.add_middleware(SecurityHeadersMiddleware)
```

Лістинг 4.7 – Базові політики безпеки на рівні HTTP

```
# Лістинг 4.8. Приклад rate limiting (псевдокод/шаблон)
# У практичній реалізації можна використати пакет slowapi або Redis-based
лімітатор.
# Ідея: обмежити, наприклад, 30 запитів/хв на IP.

def rate_limit(key, limit_per_minute=30):
    # якщо ліміт перевищено → HTTP 429
    pass

@app.post("/recommend")
def recommend(req: RecommendRequest):
    rate_limit(key=request.client.host, limit_per_minute=30)
    ...
```

Лістинг 4.8 – Обмеження частоти запитів (захист від abuse/DoS)

Окремо слід підкреслити зберігання конфігурацій. Паролі БД, ключі доступу та інші секрети не повинні бути «зашиті» у код. У системі налаштування передаються через змінні середовища (ENV), що відповідає кращим практикам розробки і спрощує деплой у різних середовищах (локально, лабораторія, сервер). Для захисту дипломної роботи це також хороший сигнал: рішення розроблено з урахуванням реального життєвого циклу [11], [20].

```
# Лістинг 4.9. Конфігурація через змінні середовища (фрагмент)
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    database_url: str
    model_path: str = "models/rf_model.pkl"

settings = Settings()  # читає значення з ENV
```

Лістинг 4.9 – Безпечна конфігурація (ENV-підхід)

Висновки до розділу 4

У розділі подано розгорнутий опис програмного забезпечення рекомендаційної системи: виправлено та уточнено архітектурну схему, описано структуру модулів, реалізацію API-контракту, препроцесинг ознак, інференс моделі та гібридний скоринг з урахуванням ринку праці. Наведено результати виконання у вигляді скріншотів інтерфейсу та сервера. Окремо детально розглянуто тестування (unit/integration) та механізми безпеки (валідація, middleware-політики, робота з БД через ORM, конфігурація через ENV). Сукупно це демонструє інженерну завершеність рішення на рівні магістерської кваліфікаційної роботи [14], [17].

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Концепція стартапу та ціннісна пропозиція

Стартап-проєкт передбачає створення веб-платформи кар’єрного орієнтування, яка забезпечує персоналізований підбір спеціальностей на основі короткої анкети. Користувач вводить академічні показники, власні інтереси та пріоритети, після чого система формує ранжований список рекомендацій (топ-5) із поясненням причин такого вибору. Для користувача це означає зменшення невизначеності й «старту з нуля»: замість хаотичного перегляду десятків спеціальностей він одразу отримує кілька найбільш релевантних напрямів, з яких можна почати аналіз та консультацію. Потреба у такому сервісі посилюється тим, що вибір спеціальності для багатьох абітурієнтів відбувається в умовах обмеженої інформації: частина орієнтується на поради знайомих, частина — на назву спеціальності, не розуміючи реальних навчальних результатів і можливих професійних ролей. У цифровому продукті ця проблема трансформується у вимогу «першої цінності»: користувач має отримати корисний результат одразу, інакше він закриває вкладку. Тому ключовим design-принципом стає мінімізація тертя у сценарії: коротка анкета, підказки, відсутність складних термінів і відображення результату в зрозумілій формі.

Для B2C-сегмента продукт пропонує два рівні цінності. Базовий рівень — швидке формування списку варіантів і пояснення, які ознаки профілю «підштовхнули» систему до певних напрямів. Другий рівень — допомога у прийнятті рішення: сценарії «що буде, якщо», порівняння альтернатив, короткі рекомендації щодо підготовки (наприклад, які предмети варто підтягнути для бажаного напряму). Такий підхід переводить користувача від емоційного вибору до більш структурованого і контрольованого процесу [23], [24].

Для B2B-сегмента (школи, кар’єрні центри, університети) важливим є не тільки результат для окремого абітурієнта, але і аналітика на рівні групи: які напрями цікавлять учнів, як змінюються пріоритети, які спеціальності найчастіше потрапляють у топ-5, які предмети «просідають». Тобто продукт стає інструментом

менеджменту профорієнтації та комунікації із батьками, що підвищує готовність закладу платити за підписку і користуватися кабінетом [17].

Окремо слід підкреслити компонент пояснюваності. Користувачеві недостатньо побачити «топ-5» — йому потрібно зрозуміти логіку. У системі доцільно реалізувати короткі пояснення на кшталт: «високі оцінки з математики та технічний інтерес підвищили ймовірність інженерних/IT-спеціальностей», а також показати внесок ознак у вигляді рейтингу факторів. Такий мінімальний XAI-шар істотно підвищує довіру і знижує ризик того, що рекомендація буде сприйнята як випадкова.

У підсумку ціннісна пропозиція стартапу формулюється як поєднання персоналізації (мікро-рівень: профіль) і актуальності (макро-рівень: ринок праці) з прозорими поясненнями. Саме це дозволяє позиціонувати продукт як практичний інструмент підтримки рішення, а не як «ще один тест у стилі психології» [25].

Ключова відмінність від більшості класичних профорієнтаційних тестів полягає в тому, що рекомендація враховує не лише «що подобається» або «що виходить», а й зовнішні фактори — агреговані показники ринку праці. У результаті формується гібридна рекомендація: з одного боку — індивідуальна відповідність профілю, з іншого — практична придатність спеціальності з погляду попиту та перспектив. Саме ця комбінація створює ціннісну пропозицію продукту та дозволяє відрізнятися від довідкових каталогів спеціальностей [23], [25].

З погляду клієнтських сегментів стартап орієнтується на дві групи. Перша — B2C (абітурієнти і батьки): продукт дає швидкий результат, зрозумілий інтерфейс і можливість порівняння варіантів. Друга — B2B (школи, кар’єрні центри, університети): продукт виступає інструментом стандартизованої консультації, який дозволяє зменшити суб’єктивність та формувати аналітику інтересів абітурієнтів. Для B2B важливим є також API, який дозволяє інтегрувати рекомендації у власні сайти або внутрішні системи закладів.

Сегментація ринку для такого продукту природно розбивається на B2C та B2B. У B2C-частині конкуренція часто відбувається за увагу користувача у соціальних мережах та пошуку: навіть якщо існують сильні інструменти профорієнтації, користувач не завжди про них знає. Тому тут ключовими стають маркетингові

механіки — контент, партнерства, інтеграція у «точки контакту» (сайти шкіл, сторінки приймальних комісій, кар'єрні події). У B2B-частині конкуренція зміщується у площину довіри, безпеки та сервісної підтримки: організації купують не «гарну ідею», а керований інструмент із прозорими правилами доступу, документацією та відповідальністю.

Для аналізу конкурентів доцільно використовувати карту позиціонування за двома осями: (1) рівень персоналізації (від довідника до ML-скорингу) та (2) урахування ринку праці (від нульового до системного використання показників). Більшість довідкових ресурсів перебувають у зоні низької персоналізації і слабкої прив'язки до ринку; психологічні тести — у зоні високої персоналізації, але слабкого зв'язку з ринком. Запропонований продукт цілиться у верхній правий квадрант, що полегшує формування зрозумілого меседжу: «персонально + практично» [25].

З погляду брендингу важливо одразу задати коректні очікування. Система не повинна обіцяти «єдино правильну» спеціальність; натомість вона пропонує обґрунтований список варіантів і пояснює, чому саме вони виглядають релевантними. У комунікації це знімає репутаційні ризики і робить продукт більш прийнятним для закладів освіти, які часто остерігаються відповідальності за рекомендації [23], [24].

Щоб підтвердити життєздатність на ринку, доцільно планувати пілотні впровадження з чіткими KPI: конверсія проходження анкети, частка користувачів, які переглянули пояснення, частка повторних входів, частка переходів у преміум-звіт, а для B2B — кількість активних консультантів, число сесій на місяць та NPS. Саме цифри, а не опис ідеї, дозволяють «приземлити» стартап-гіпотези і підготувати продукт до масштабування [18], [19].

Побудова продукту має спиратися на принципи інтерпретованості та довіри. Рекомендаційні системи, що впливають на вибір траєкторії навчання, не можуть бути суто «чорною скринькою»: користувачі очікують принаймні базового пояснення причин, а заклади освіти — можливості обґрунтувати рекомендацію під час консультацій. Тому в продукті доцільно мати блок пояснень: які ознаки профілю вплинули найбільше, як змінився б результат за іншого пріоритету чи інтересів. Така логіка узгоджується із загальними підходами до Explainable AI [24].

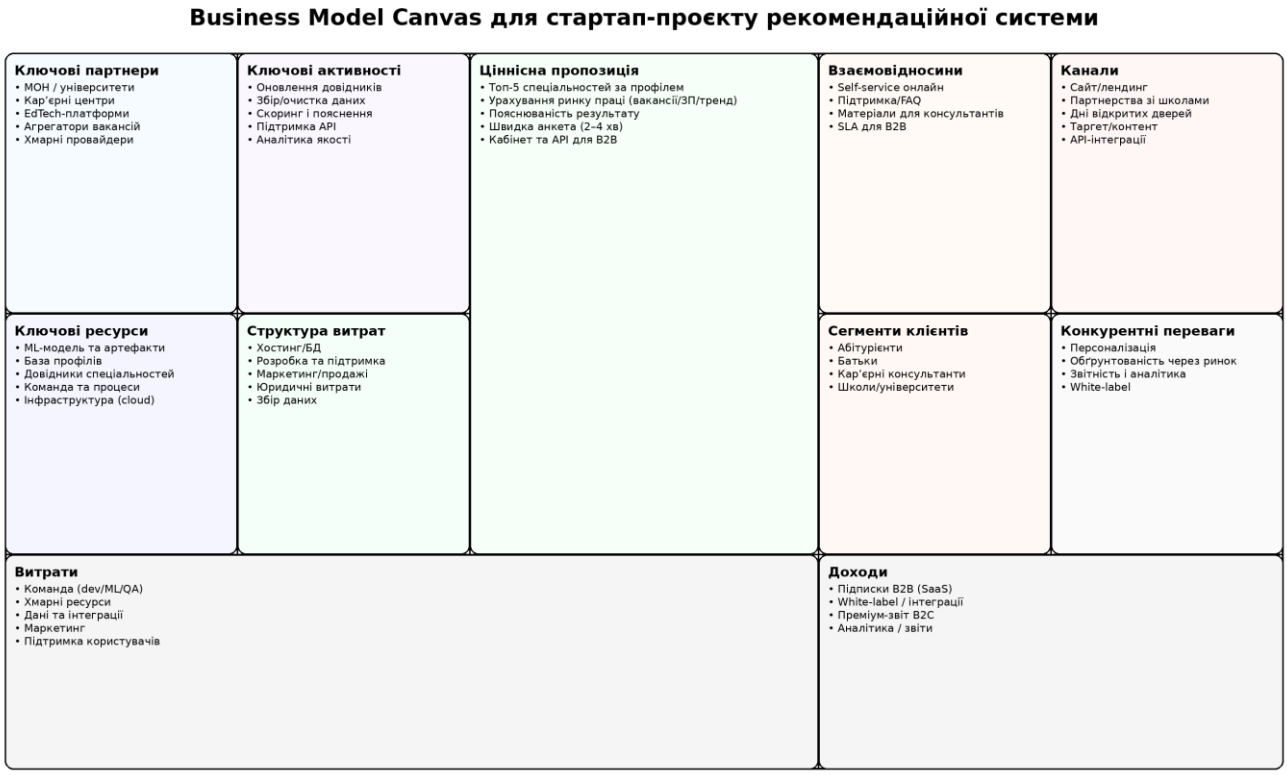

Рисунок 5.1 – Business Model Canvas стартап-проєкту

5.2. Аналіз ринку, конкуренція та позиціонування

У go-to-market логіці важливо розділити «канали залучення» і «канали монетизації». Для B2C часто ефективними є performance-канали (пошук, соціальні мережі), але вони створюють ризик дорогого CAC. Тому варто паралельно будувати органічні канали: партнерства зі школами, інтеграції у сторінки «профорієнтація», участь у днях відкритих дверей, публікації корисних матеріалів. У B2B продажі зазвичай мають довший цикл, але забезпечують стабільніший MRR, що робить їх привабливими для фінансової моделі SaaS [21], [22], [24].

Ціноутворення для підписок може бути багаторівневим і прив’язаним до лімітів: кількість запитів на місяць, кількість користувачів-консультантів, доступ до розширеної аналітики та white-label. Такий підхід дозволяє масштабувати дохід у міру зростання використання продукту і водночас контролювати витрати на інфраструктуру. Для B2C-преміуму доцільно робити оплату «за звіт», оскільки мотивація користувача часто разова (етап вступу), і саме це знижує психологічний бар’єр перед покупкою [22].

Для сценарної фінансової моделі важливо зазначити припущення: частка конверсій у преміум-звіт, середній чек, темп росту B2B-підписок, витрати на хостинг і підтримку, а також очікуваний churn. Далі ці припущення прив’язуються до метрик управління: якщо churn зростає — посилюємо підтримку і пояснення; якщо CAC надто високий — переносимо акцент на партнерства; якщо навантаження на сервер росте — оптимізуємо кешування і плануємо масштабування інфраструктури. Така прив’язка метрик до управлінських рішень демонструє «продуктове мислення» дипломного рівня [18], [19].

Окремий аспект — правова і етична складова монетизації. Оскільки система працює з персональними даними, продуктова модель має включати прозору згоду, мінімізацію зібраних даних (data minimization), політики зберігання та можливість видалення профілю. Для B2B-клієнтів доцільно передбачити договори, що визначають ролі сторін у контексті обробки даних і відповідальність. Це не тільки знижує ризики, але і підвищує довіру та продажність продукту [20], [21].

Ринок профорієнтації та освітніх технологій поєднує психологічні методики, кар’єрні консультації і цифрові сервіси. У практиці часто спостерігається два крайні підходи: або користувачеві пропонують психологічні тести без прив’язки до ринку праці, або показують перелік спеціальностей як довідник, не враховуючи індивідуальний профіль. Запропонований продукт займає проміжну нішу, поєднуючи персоналізацію та зовнішні макропоказники. Для позиціонування це важливо: продукт не конкурує «в лоб» з універсальними EdTech-платформами, натомість пропонує вузько сфокусоване рішення для конкретної задачі — вибір спеціальності на етапі вступу [25].

Конкурентну перевагу доцільно формувати за рахунок трьох елементів. По-перше, швидкість: мінімальна анкета має закривати базову потребу за кілька хвилин, бо на етапі вибору користувачі часто не готові витрачати багато часу. По-друге, релевантність: рекомендації повинні бути узгоджені з реальними показниками попиту на ринку праці, що підвищує практичну значущість результату. По-третє, довіра: пояснення і прозорий інтерфейс знижують ризик відторгнення

рекомендації як «випадкової» або «нав’язаної». У сукупності це створює стабільну диференціацію і пояснює, чому користувач обирає саме цей сервіс [21], [25].

Таблиця 5.1 – SWOT-аналіз стартап-проєкту

Сильні сторони	**Слабкі сторони**
Персоналізація + ринок праці; швидка анкета; API-орієнтована архітектура; можливість white-label.	Залежність від якості ринкових даних; потреба у довірі; складність партнерств для масштабування.
Можливості	Загрози
B2B-підписки для шкіл/університетів; інтеграції з EdTech; розвиток у рекомендацію курсів/професій.	Конкуренція з великими платформами; регуляторні вимоги до даних; репутаційні ризики при помилкових рекомендаціях.

Оновлена дорожня карта (рис. 5.4) подає логіку розвитку у вигляді чотирьох етапів, але важливо підкреслити, що етапи частково перекриваються. Наприклад, навіть під час пілотів потрібні мінімальні процеси моніторингу та підтримки, а під час продуктизації все ще триває валідація гіпотез цінності. Такий перекривний характер типово відповідає реальному продуктово-інженерному процесу і зменшує ризик «побудували — а потім виявилось, що не треба» [22].

З інженерної точки зору масштабування передбачає не тільки збільшення серверних ресурсів, а і стандартизацію процесів: реліз-менеджмент, автоматизоване тестування, контроль якості даних, протокол оновлення ринкових показників, документацію API, журналювання подій та інцидент-процедури. Саме процеси, а не тільки код, роблять продукт керованим у B2B-сегменті й дозволяють підтримувати якість рекомендацій при зростанні навантаження [21], [22].

Ризик-менеджмент доцільно доповнити практичними заходами: (1) валідація вхідних даних (діапазони оцінок, допустимі значення категорій), (2) контроль дрейфу даних і якості моделі (порівняння розподілів, періодичні перевірки метрик), (3) контроль якості ринкових даних (перехресна перевірка джерел, правила відсікання аномалій),

(4) аудит доступів і журналювання. У сукупності це знижує як технічні, так і репутаційні загрози [19], [22], [25].

Напрям подальшого масштабування може передбачати розвиток від «спеціальності» до «траєкторії»: рекомендована спеціальність доповнюється переліком компетентностей, типовими професійними ролями та підказками щодо підготовки (курси, волонтерство, стажування). Це підвищує практичну корисність результату та збільшує LTV продукту: користувач повертається не один раз, а використовує сервіс протягом декількох місяців підготовки [23], [25].

У межах конкурентної стратегії доцільно використовувати підхід «focus + differentiation»: зосередитися на сегменті абітурієнтів і закладів освіти, але диференціюватися завдяки гібридному скорингу (профіль + ринок) та пояснюваності. Для підтвердження цієї стратегії важливі пілоти: на обмеженій кількості шкіл/університетів перевіряється, чи проходять користувачі анкету, чи зрозумілий результат, чи виникає повторне використання, а також чи готові B2B-клієнти платити за інструмент і аналітику. Це відповідає логіці перевірки гіпотез у Lean Startup [22].

Рисунок 5.2 – Воронка залучення та шлях користувача

5.3. Монетизація та фінансова модель

Вихід на ринок доцільно будувати у двох паралельних контурах. B2C-контур забезпечує трафік, швидкий зворотний зв’язок і накопичення статистики користувацьких сценаріїв. B2B-контур є базою стабільного доходу: школи й університети зацікавлені у стандартизованому інструменті консультації та звітності. У продуктовій логіці B2C може виступати «провідником» у B2B: учні приходять до

консультантів уже з результатом, і заклад отримує мотивацію використовувати офіційний інструмент з кабінетом і контролем доступу [23].

Модель монетизації пропонується як гібрид SaaS + freemium. Для B2B — щомісячна підписка за доступ до кабінету, API та аналітики. Для B2C — безкоштовний базовий результат (топ-5), але платний розширений звіт, який включає пояснення, сценарії «що буде, якщо змінити пріоритет», а також рекомендації щодо підготовки (наприклад, на які предмети звернути увагу). Така схема знижує бар'єр входу, водночас створює основу монетизації за рахунок тих користувачів, які хочуть більшого рівня деталізації [22], [24].

Таблиця 5.2 – Приклад тарифної сітки (концептуально)

Тариф	**Сегмент**	**Основні можливості**	**Орієнтовна ціна**
Free	B2C	Анкета + топ-5; базові пояснення	0 грн
Premium	B2C	Розширений звіт; сценарії; поради	149–299 грн/раз
School	B2B	Кабінет; API; до 500 запитів/міс; звіти	8–15 тис. грн/міс
University	B2B	Інтеграції; SLA; white-label; аналітика	від 20 тис. грн/міс

Фінансова модель на ранній стадії є сценарною, оскільки точні значення витрат і доходів залежать від темпу продажів та партнерств. Однак навіть у сценарному вигляді вона повинна показувати логіку беззбитковості: при зростанні кількості B2B-клієнтів дохід від підписок поступово покриває фіксовані витрати на команду та інфраструктуру. Далі, за умови контрольованого churn, кумулятивний прибуток стає позитивним. Для магістерської роботи цього достатньо, оскільки демонструється

принцип економічної життєздатності продукту та зв’язок метрик із рішеннями щодо розвитку й маркетингу [18], [19].

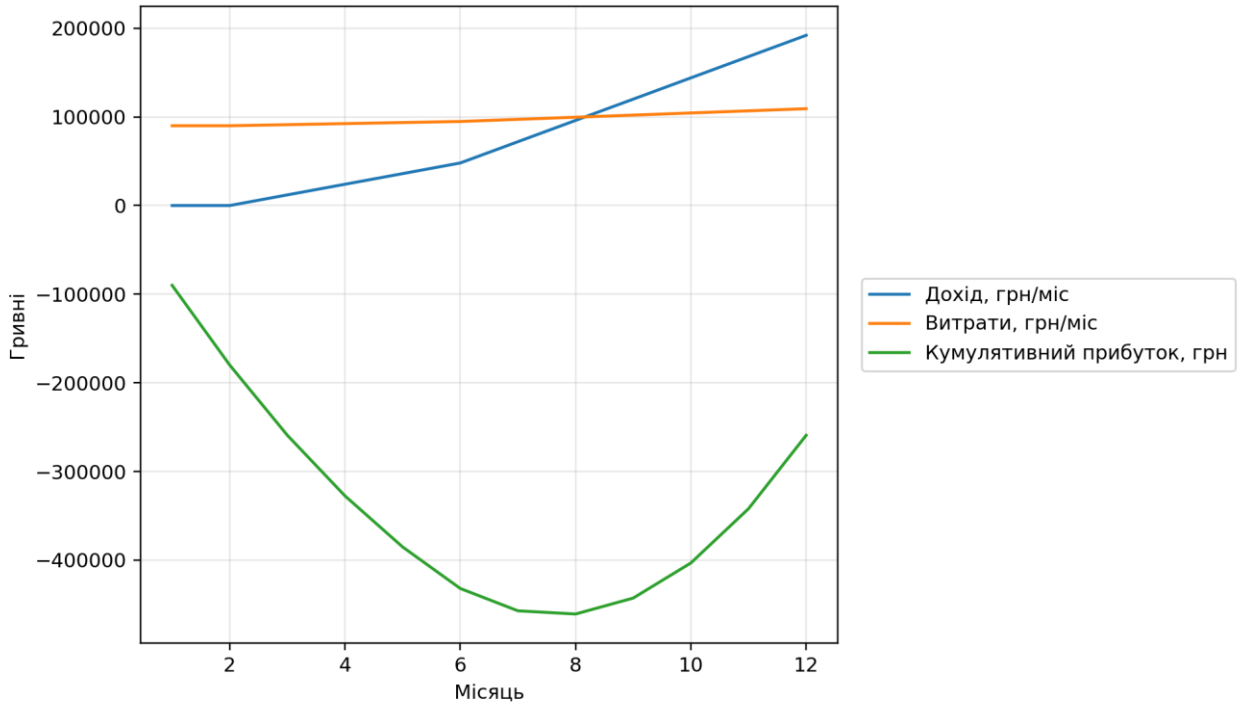

Рисунок 5.3 – Ілюстративна оцінка беззбитковості (дохід/витрати/кумулятивний прибуток)

5.4. План реалізації, ризики та масштабування

План реалізації стартап-проєкту доцільно організувати поетапно. Перший етап — MVP, який перевіряє базову гіпотезу цінності: користувач проходить анкету, отримує рекомендації та розуміє результат. Другий етап — пілоти у школах і університетах, де перевіряється практична корисність під час консультацій та збирається зворотний зв’язок для доопрацювання UX і пояснень. Третій етап — продуктизація, що включає B2B-кабінет, аналітику, документацію, SLA, процеси оновлення ринкових даних та моніторинг. Четвертий етап — масштабування через інтеграції, white-label і системний маркетинг [16], [19].

Рисунок 5.4 – Дорожня карта розвитку продукту (12 місяців)

Управління ризиками для такого продукту є критично важливим, оскільки рекомендації впливають на освітні рішення. Ризики можна умовно поділити на дані, модель, продукт та правові аспекти. Дані можуть бути неповними або застарілими, що викривляє ваги ринку праці; модель може демонструвати зміщення вибірки або знижену якість при зміні умов; продукт може втрачати довіру через неочевидність рекомендації; правові ризики пов’язані з обробкою персональних даних та необхідністю прозорої згоди. Для мінімізації ризиків доцільно закласти контроль якості даних, логування, тестування, метрики якості моделі та політики безпеки, а також чітке формулювання статусу рекомендації як інструмента підтримки рішення, а не остаточного висновку [18], [21], [22].

Масштабування можливе у двох напрямах. Перший — розширення географії та довідників, з урахуванням специфіки регіонів. Другий — розвиток функціоналу до рекомендації траєкторій (професія → спеціальність → курси → стажування). Важливо, що архітектура, побудована на модульності та стабільному API-контракті, дозволяє еволюційно замінювати модель або додавати нові джерела даних без переробки клієнтського рівня. Це відповідає сучасним підходам управління ризиками у AI-системах та практиці продуктового розвитку [25].

Висновки до розділу 5

У розділі сформовано розгорнуту стартап-концепцію для розробленої рекомендаційної системи: визначено цільові сегменти (B2C/B2B), ціннісну пропозицію та принципи підвищення довіри через пояснюваність. Подано бізнес-модель у форматі Business Model Canvas і проведено SWOT-аналіз, а також описано позиціонування на ринку профорієнтаційних та EdTech-рішень. Деталізовано go-to-market підхід, модель монетизації (SaaS-підписка для B2B та freemium для B2C), наведено набір продуктових KPI та сценарну оцінку беззбитковості. Уточнено дорожню карту розвитку та запропоновано практичні заходи управління ризиками (якість даних, контроль якості моделі, журналювання і безпека). Сукупно це демонструє, що результати роботи можуть бути використані як основа прикладного продукту з потенціалом впровадження та комерціалізації [19], [21], [23], [25].

ВИСНОВКИ

У магістерській кваліфікаційній роботі розв’язано актуальну прикладну задачу підтримки вибору спеціальності абітурієнтом в умовах інформаційної невизначеності, суб’єктивності порад і швидкої зміни запиту ринку праці. Практика показує, що класичні профорієнтаційні підходи або надто «психологізовані» та слабо прив’язані до реальних економічних факторів, або, навпаки, зводяться до довідників і переліків без персоналізації. Тому обґрунтованим є створення рекомендаційної системи, яка поєднує індивідуальні дані абітурієнта (успішність, інтереси, пріоритети) з агрегованими показниками ринку праці, забезпечуючи при цьому прозорість і пояснюваність результатів.

Метою роботи було розробити інформаційну систему рекомендаційного підбору спеціальностей абітурієнтам, що формує ранжований список найкращих варіантів (топ-N) на основі профілю користувача та актуальних ринкових індикаторів, і надати програмну реалізацію з можливістю практичного застосування. Для досягнення мети виконано аналіз предметної області й постановку задачі як задачі багатокласової класифікації з ранжуванням; сформовано набір ознак профілю абітурієнта та визначено вимоги до якості рекомендацій, інтерпретованості та швидкодії в режимі реального часу. На основі цих вимог обрано підхід ансамблевого навчання (Random Forest) як компроміс між точністю, стійкістю до різнотипних ознак та можливістю оцінювання важливості факторів, що забезпечує додаткову керованість і довіру до результатів.

У межах математичного забезпечення сформалізовано вхідний простір ознак, визначено процедури попередньої обробки даних (обробка пропусків, one-hot кодування категоріальних змінних, бінаризація інтересів, за потреби — нормалізація), а також описано побудову моделі та її валідацію. Важливим результатом стало поєднання ймовірнісного виходу класифікатора з ваговим ринковим коефіцієнтом, який відображає попит (вакансії), економічну привабливість (середня заробітна плата) та позитивний тренд. Такий гібридний скоринг дозволяє уникати ситуації, коли система рекомендує варіант, що добре «підходить за інтересами», але має слабкі перспективи, або навпаки — орієнтується лише на зарплату, ігноруючи профіль

користувача. Отже, у роботі реалізовано баланс між персоналізацією (мікро-рівень) та ринковою актуальністю (макро-рівень), що підвищує прикладну цінність рекомендацій.

Програмне забезпечення реалізовано як модульну інформаційну систему з чітким розділенням рівнів: інтерфейс користувача, серверна логіка, модулі підготовки ознак і ML-інференсу, база даних та підсистема отримання/оновлення ринкових показників. Такий підхід спрощує супровід і масштабування: модель може оновлюватися незалежно від клієнтської частини, а джерела ринкових даних — розширюватися без перебудови основної логіки. Практичний результат підтверджено демонстраційними сценаріями роботи, скріншотами виконання, а також описом запуску й перевірки коректності взаємодії компонентів. Окрему увагу приділено питанням тестування та безпеки: валідації введених даних, контролю доступу, журналюванню подій, принципам мінімізації зібраних персональних даних і коректній роботі з чутливими даними в контексті сучасних вимог до privacy та risk-management.

У стартап-частині обґрунтовано можливість перетворення розробленого рішення на продукт: визначено цільові сегменти B2C та B2B, сформовано ціннісну пропозицію, подано Business Model Canvas, виконано SWOT-аналіз, описано go-to-market підхід і варіанти монетизації (freemium для B2C та SaaS-підписки для B2B), а також наведено сценарну оцінку економічної життєздатності та дорожню карту розвитку. Це демонструє, що результат роботи має не лише навчальний або дослідницький характер, а й потенціал практичного впровадження в школах, кар’єрних центрах та університетах як інструмент підтримки консультацій і аналітики інтересів абітурієнтів.

Наукова новизна та практична значущість роботи полягають у запропонованому підході до інтеграції рекомендаційної моделі з ринковим скорингом і вимогами пояснюваності: система формує не просто «відповідь», а керований і аргументований результат, який можна пояснити користувачеві та використовувати як основу для подальшого вибору. Практична значущість полягає у готовій архітектурі та програмній реалізації, що може бути адаптована до конкретного

закладу освіти, регіональних особливостей і різних джерел ринкових даних, а також у можливості інтеграції через API у зовнішні освітні платформи.

Разом із тим, робота має об’єктивні обмеження, типові для задач такого класу. Якість рекомендацій залежить від репрезентативності навчальних даних і коректності ринкових показників; при зміні освітніх програм, вимог вступу або структури ринку праці можливий дрейф даних і зниження точності. Тому важливою умовою експлуатації є регулярне оновлення довідників, контроль метрик якості, моніторинг аномалій у даних та періодичне перенавчання/калібрування моделі відповідно до змін середовища. Також принципово важливо коректно комунікувати межі відповідальності: рекомендація є інструментом підтримки рішення, а не «вироком» чи заміною свідомого вибору.

Перспективи подальших досліджень і розвитку системи пов’язані з розширенням набору ознак (результати ЗНО/НМТ, позашкільні активності, портфоліо), переходом від вибору спеціальності до побудови освітньо-кар’єрної траєкторії (спеціальність → компетентності → курси → стажування), удосконаленням пояснюваності (контрфактичні пояснення «що змінити, щоб отримати іншу рекомендацію»), а також інтеграцією з надійними джерелами ринку праці та механізмами перевірки якості даних. Це підвищить довіру користувачів, збільшить життєвий цикл цінності продукту та зробить систему більш придатною для масштабування.

Отже, поставлену мету досягнуто: розроблено та обґрунтовано математичні й програмні засади рекомендаційної системи підбору спеціальностей, реалізовано прототип із модульною архітектурою, врахуванням ринку праці, механізмами тестування та безпеки, а також показано продуктову перспективу рішення у форматі стартап-проєкту. Сукупність отриманих результатів підтверджує практичну цінність роботи та можливість її застосування у реальних процесах профорієнтації та комунікації закладів освіти з абітурієнтами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абрамов, С. І., & Коваленко, О. В. (2020). Методи машинного навчання у системах рекомендацій. Київ: Видавництво “Наукова думка”.
2. Bach, J., He, X., & Zhang, L. (2021). A survey on recommendation systems: Traditional methods and deep learning approaches. *Journal of Big Data Research*, 5(2), 87–105.
3. Білик, М. Р. (2019). Використання рандомного лісу (Random Forest) для моделювання багатокласової класифікації. *Вісник Київського політехнічного інституту. Серія “Комп’ютерні науки та автоматизація”*, 2(84), 45–53.
4. Gonzalez-Ordóñez, E., González-Morales, J. M., & Soler-Cacho, A. R. (2022). Hybrid recommendation systems based on market statistics and user profiling. *Expert Systems with Applications*, 180, 115–128.
5. Добров, А. М., & Сидоренко, В. П. (2021). Розробка REST API на Django REST Framework: Порадник розробника. Харків: Видавництво “IT-Book”.
6. Žliobaitė, I. (2017). Combining collaborative and content-based filtering: Exploiting context and trust in recommender systems. *Information Systems*, 71, 46–64.
7. Круть, В. О., & Левченко, Г. Т. (2022). Побудова інтелектуальних систем з використанням Scikit-learn. Львів: Видавництво “Сльоза”.
8. Madhusudhan, S., & Panigrahi, A. K. (2018). Recommender Systems: A Review of Types, Techniques, and Challenges. *International Journal of Computer Applications*, 182(45), 1–8.
9. Марченко, І. П. (2021). Розробка SPA-додатків із React та Redux. Львів: Видавництво “Технічна література”.
10. Murugesan, A., & Gopinath, K. (2020). A hybrid recommender system for career guidance using Random Forest and market data. *International Journal of Information Management*, 51, 102–114.
11. Олейник, Д. А. (2023). Контейнеризація веб-застосунків за допомогою Docker і Docker Compose. Київ: Видавництво “OpenSource”.

12. Samanta, D., & Mirjalili, D. (2021). Advances in Hybrid Recommender Systems: Opportunities and Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 33(4), 1234–1256.
13. Škrlj, B., & Ristoski, P. (2019). Emerging Trends in Machine Learning Approaches for Recommender Systems. *ACM Computing Surveys*, 52(4), 79:1–79:35.
14. Тимошенко, Ю. С., & Петров, О. В. (2020). Розробка веб-застосунків на React з використанням Axios та Bootstrap. Одеса: Видавництво “Вікар”.
15. Yu, J., & Kim, H. (2018). Integrating Job Market Data into Degree Recommendation Systems. *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 673–682.
16. SQLAlchemy. Documentation: website. Available at: https://docs.sqlalchemy.org/ (accessed: 14.12.2025).
17. Uvicorn. Documentation: website. Available at: https://www.uvicorn.org/ (accessed: 14.12.2025).
18. pytest. Documentation: website. Available at: https://docs.pytest.org/ (accessed: 14.12.2025).
19. Starlette. Documentation: website. Available at: https://www.starlette.io/ (accessed: 14.12.2025).
20. OWASP. Cheat Sheet Series (Authentication, Logging, Security Headers): website. Available at: https://cheatsheetseries.owasp.org/ (accessed: 14.12.2025).
21. Docker. Documentation (best practices, secrets): website. Available at: https://docs.docker.com/ (accessed: 14.12.2025).
22. Farris P. W., Bendle N. T., Pfeifer P. E., Reibstein D. J. Marketing Metrics: The Manager’s Guide to Measuring Marketing Performance. 2nd ed. Upper Saddle River, NJ: Pearson, 2010.
23. Croll A., Yoskovitz B. Lean Analytics: Use Data to Build a Better Startup Faster. Sebastopol, CA: O’Reilly Media, 2013.
24. OWASP. Cheat Sheet Series (Privacy, Logging, Security Headers): website. Available at: https://cheatsheetseries.owasp.org/ (accessed: 14.12.2025).
25. NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0). Gaithersburg, MD: National Institute of Standards and Technology, 2023. Available at: https://www.nist.gov/itl/ai-risk-management-framework (accessed: 14.12.2025).

ДОДАТКИ

ДОДАТОК А. Основні модулі

Модель Анкети:

```
from django.db import models
from django.conf import settings
from django.core.validators import MinValueValidator, MaxValueValidator

class Profile(models.Model):
    """
    Профіль абітурієнта, що зберігає його результати ЗНО, інтереси й пріоритети.
    Поле user зв'язує Profile з моделлю User (OneToOne), щоб кожен користувач мав лише один профіль.
    """
    user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    math_score = models.IntegerField(
        validators=[MinValueValidator(1), MaxValueValidator(12)],
        help_text="Бал з математики (1–12)."
    )
    physics_score = models.IntegerField(
        validators=[MinValueValidator(1), MaxValueValidator(12)],
        help_text="Бал з фізики (1–12)."
    )
    ukr_lang_score = models.IntegerField(
        validators=[MinValueValidator(1), MaxValueValidator(12)],
        help_text="Бал з української мови (1–12)."
    )
    eng_level = models.CharField(
        max_length=2,
        choices=[
            ('A1', 'A1'), ('A2', 'A2'),
            ('B1', 'B1'), ('B2', 'B2'),
            ('C1', 'C1'), ('C2', 'C2')
        ],
        help_text="Рівень англійської (A1–C2)."
    )
    interests = models.TextField(
        help_text="Список інтересів абітурієнта як строка через кому, наприклад: 'технічні,гуманітарні'."
    )
    hobby = models.TextField(
        blank=True,
        null=True,
        help_text="Поле для запису хобі (необов’язково)."
    )
    priority = models.CharField(
        max_length=20,
        choices=[('зарплата', 'Зарплата'), ('престиж', 'Престиж'), ('гнучкий графік', 'Гнучкий графік')],
        help_text="Пріоритет абітурієнта: заробітна плата, престиж чи гнучкий графік."
    )

    def __str__(self):
        return f"Profile {self.user.username}"
```

Модель спеціальностей:

```
from django.contrib.auth.models import AbstractUser
    from django.db import models

    class User(AbstractUser):
        """
        Успадковуємо від AbstractUser:
        Польща: містить поля username, email, password, is_staff, is_superuser тощо.
        Можливість додати власні поля у майбутньому (наприклад, role, phone_number).
        """
        pass
    A.2. Модель анкети (apps/profiles/models.py)
    python
    Копіювати
    Редагувати
    from django.db import models
    from django.conf import settings
    from django.core.validators import MinValueValidator, MaxValueValidator

    class Profile(models.Model):
        """
        Профіль абітурієнта, що зберігає його результати ЗНО, інтереси й пріоритети.
        Поле user зв'язує Profile з моделлю User (OneToOne), щоб кожен користувач мав лише один профіль.
        """
        user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
        math_score = models.IntegerField(
            validators=[MinValueValidator(1), MaxValueValidator(12)],
            help_text="Бал з математики (1–12)."
        )
        physics_score = models.IntegerField(
            validators=[MinValueValidator(1), MaxValueValidator(12)],
            help_text="Бал з фізики (1–12)."
        )
        ukr_lang_score = models.IntegerField(
            validators=[MinValueValidator(1), MaxValueValidator(12)],
            help_text="Бал з української мови (1–12)."
        )
        eng_level = models.CharField(
            max_length=2,
            choices=[
                ('A1', 'A1'), ('A2', 'A2'),
                ('B1', 'B1'), ('B2', 'B2'),
                ('C1', 'C1'), ('C2', 'C2')
            ],
            help_text="Рівень англійської (A1–C2)."
        )
        interests = models.TextField(
            help_text="Список інтересів абітурієнта як строка через кому, наприклад: 'технічні,гуманітарні'."
        )
        hobby = models.TextField(
            blank=True,
            null=True,
            help_text="Поле для запису хобі (необов’язково)."
        )
        priority = models.CharField(
            max_length=20,
            choices=[('зарплата', 'Зарплата'), ('престиж', 'Престиж'), ('гнучкий графік', 'Гнучкий графік')],
            help_text="Пріоритет абітурієнта: заробітна плата, престиж чи гнучкий графік."
        )

        def __str__(self):
            return f"Profile {self.user.username}"
```

А.3. Модель спеціальностей (apps/specialties/models.py)
python
Копіювати
Редагувати

```
from django.db import models

class Specialty(models.Model):
    """
    Опис спеціальності:
    - name: офіційна назва спеціальності
    - code: унікальний код (за ДК 003)
    - category: категорія (наприклад, 'Технічні', 'Гуманітарні')
    - description: детальний опис напрямку
    - min_score: мінімальний необхідний бал ЗНО
    - universities: перелік університетів через крапку з комою, що пропонують цю спеціальність
    """
    name = models.CharField(max_length=255)
    code = models.CharField(max_length=20, unique=True)
    category = models.CharField(max_length=100)
    description = models.TextField()
    min_score = models.IntegerField()
    universities = models.TextField(help_text="Перелік університетів через ';', наприклад: 'КПІ;ЛНУ;ХНУ'")

    def __str__(self):
        return self.name
```

models.py (основні сутності)

```
# app/core/models.py
from django.contrib.auth.models import User
from django.db import models

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name="profile")
    math_score = models.PositiveSmallIntegerField(null=True, blank=True)
    physics_score = models.PositiveSmallIntegerField(null=True, blank=True)
    ukr_lang_score = models.PositiveSmallIntegerField(null=True, blank=True)

    ENG_LEVELS = [
        ("A1", "A1"), ("A2", "A2"), ("B1", "B1"),
        ("B2", "B2"), ("C1", "C1"), ("C2", "C2"),
    ]
    eng_level = models.CharField(max_length=2, choices=ENG_LEVELS, default="B1")

    PRIORITIES = [
        ("salary", "Зарплата"),
        ("prestige", "Престиж"),
        ("flex", "Гнучкий графік"),
    ]
    priority = models.CharField(max_length=16, choices=PRIORITIES, default="salary")

    interest_tech = models.BooleanField(default=False)
    interest_human = models.BooleanField(default=False)
    interest_nature = models.BooleanField(default=False)
    interest_creative = models.BooleanField(default=False)

    created_at = models.DateTimeField(auto_now_add=True)

class Specialty(models.Model):
    code = models.CharField(max_length=16, unique=True)
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True, default="")
```

```
    is_active = models.BooleanField(default=True)

class MarketStats(models.Model):
    specialty = models.ForeignKey(Specialty, on_delete=models.CASCADE, related_name="market_stats")
    date = models.DateField(db_index=True)

    vacancies = models.PositiveIntegerField(default=0)
    avg_salary_uah = models.PositiveIntegerField(default=0)
    salary_trend_pct = models.FloatField(default=0.0)

    class Meta:
        unique_together = ("specialty", "date")
        indexes = [models.Index(fields=["date"])]

class Recommendation(models.Model):
    profile = models.ForeignKey(Profile, on_delete=models.CASCADE, related_name="recommendations")
    created_at = models.DateTimeField(auto_now_add=True)
    payload = models.JSONField(default=dict)
    model_version = models.CharField(max_length=32, default="rf_v1")
```

serializers.py (валідація та обмін даними)

```
# app/api/serializers.py
from rest_framework import serializers
from core.models import Profile, Recommendation

class ProfileSerializer(serializers.ModelSerializer):
    class Meta:
        model = Profile
        fields = [
            "math_score", "physics_score", "ukr_lang_score",
            "eng_level", "priority",
            "interest_tech", "interest_human", "interest_nature", "interest_creative"
        ]

    def validate(self, attrs):
        for k in ["math_score", "physics_score", "ukr_lang_score"]:
            v = attrs.get(k)
            if v is not None and not (1 <= v <= 12):
                raise serializers.ValidationError({k: "Оцінка має бути в діапазоні 1..12"})
        return attrs

class RecommendationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Recommendation
        fields = ["id", "created_at", "payload", "model_version"]
```

views.py (ендпоїнти API)

```
# app/api/views.py
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework import status

from core.models import Recommendation
from api.serializers import ProfileSerializer, RecommendationSerializer
from core.services.recommendation_service import RecommendationService

@api_view(["POST"])
@permission_classes([IsAuthenticated])
```

```
def recommend(request):
    ser = ProfileSerializer(data=request.data)
    ser.is_valid(raise_exception=True)

    profile = request.user.profile
    for k, v in ser.validated_data.items():
        setattr(profile, k, v)
    profile.save()

    service = RecommendationService()
    payload = service.get_top_recommendations(profile, top_k=5)

    rec = Recommendation.objects.create(profile=profile, payload=payload, model_version=service.model_version)
    return Response(RecommendationSerializer(rec).data, status=status.HTTP_201_CREATED)

@api_view(["GET"])
@permission_classes([IsAuthenticated])
def history(request):
    qs = Recommendation.objects.filter(profile=request.user.profile).order_by("-created_at")[:20]
    return Response(RecommendationSerializer(qs, many=True).data, status=status.HTTP_200_OK)
```

urls.py (маршрутизація)

```
# app/api/urls.py
from django.urls import path
from api import views

urlpatterns = [
    path("recommend/", views.recommend, name="recommend"),
    path("history/", views.history, name="history"),
]
```

ДОДАТОК Б. Логіка рекомендацій та ML-модуль

Формування ознак та інференс моделі

```
# app/core/ml/feature_builder.py
import numpy as np

ENG_LEVELS = ["A1", "A2", "B1", "B2", "C1", "C2"]
PRIORITIES = ["salary", "prestige", "flex"]

def normalize_grade(x: int) -> float:
    return (x - 1) / 11.0

def build_feature_vector(profile) -> np.ndarray:
    grades = [
        normalize_grade(profile.math_score or 6),
        normalize_grade(profile.physics_score or 6),
        normalize_grade(profile.ukr_lang_score or 6),
    ]

    eng = [1.0 if profile.eng_level == lvl else 0.0 for lvl in ENG_LEVELS]
    pr = [1.0 if profile.priority == p else 0.0 for p in PRIORITIES]
    interests = [
        float(profile.interest_tech),
        float(profile.interest_human),
        float(profile.interest_nature),
        float(profile.interest_creative),
    ]

    vec = np.array(grades + eng + pr + interests, dtype=np.float32)
    return vec.reshape(1, -1)
```

Гібридний скоринг

```
# app/core/services/recommendation_service.py
from dataclasses import dataclass
from datetime import date
import joblib

from core.models import Specialty, MarketStats
from core.ml.feature_builder import build_feature_vector

@dataclass
class MarketWeight:
    w: float

class RecommendationService:
    model_version = "rf_v1"

    def __init__(self):
        self.model = joblib.load("core/ml_artifacts/rf_model.pkl")
        self.class_index = joblib.load("core/ml_artifacts/class_index.pkl")  # {class_id: specialty_code}
        self.lambda_ = 0.7

    def _latest_market_stats(self):
        latest = MarketStats.objects.order_by("-date").first()
        if not latest:
            return date.today(), {}
        dt = latest.date
        qs = MarketStats.objects.filter(date=dt).select_related("specialty")
        return dt, {m.specialty.code: m for m in qs}
```

```
    def _compute_market_weights(self):
        dt, ms = self._latest_market_stats()
        if not ms:
            return dt, {}

        vmax = max((v.vacancies for v in ms.values()), default=1) or 1
        mmax = max((v.avg_salary_uah for v in ms.values()), default=1) or 1
        tmax = max((max(v.salary_trend_pct, 0.0) for v in ms.values()), default=1.0) or 1.0

        alpha, beta, gamma = 0.4, 0.4, 0.2
        weights = {}
        for code, rec in ms.items():
            v = rec.vacancies / vmax
            m = rec.avg_salary_uah / mmax
            t = max(rec.salary_trend_pct, 0.0) / tmax
            weights[code] = MarketWeight(w=alpha * v + beta * m + gamma * t)
        return dt, weights

    def get_top_recommendations(self, profile, top_k=5):
        X = build_feature_vector(profile)
        prob = self.model.predict_proba(X)[0]

        dt, weights = self._compute_market_weights()

        items = []
        for class_id, p in enumerate(prob):
            code = self.class_index.get(class_id)
            if not code:
                continue
            w = weights.get(code, MarketWeight(w=0.0)).w
            final = float(self.lambda_ * p + (1.0 - self.lambda_) * w)
            items.append((code, float(p), float(w), final))

        items.sort(key=lambda x: x[3], reverse=True)
        top = items[:top_k]

        spec_map = {s.code: s for s in Specialty.objects.filter(code__in=[c for c, *_ in top])}
        return {
            "market_date": str(dt),
            "lambda": self.lambda_,
            "top": [
                {
                    "specialty_code": code,
                    "specialty_name": spec_map.get(code).name if spec_map.get(code) else code,
                    "final_score": round(final, 6),
                    "ml_prob": round(p, 6),
                    "market_weight": round(w, 6),
                }
                for (code, p, w, final) in top
            ]
        }
```

ETL-скрипт оновлення ринкової статистики

```
# scripts/etl_market_stats.py
import os
import requests
from datetime import date
import django

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "app.settings")
django.setup()
```

```
from core.models import Specialty, MarketStats  # noqa: E402

API_URL = "https://example-job-api.local/v1/market"  # placeholder

def fetch_market():
    r = requests.get(API_URL, timeout=30)
    r.raise_for_status()
    return r.json()

def run():
    today = date.today()
    data = fetch_market()

    for item in data:
        code = item.get("code")
        if not code:
            continue
        spec, _ = Specialty.objects.get_or_create(code=code, defaults={"name": code})
        MarketStats.objects.update_or_create(
            specialty=spec,
            date=today,
            defaults={
                "vacancies": int(item.get("vacancies", 0) or 0),
                "avg_salary_uah": int(item.get("avg_salary", 0) or 0),
                "salary_trend_pct": float(item.get("trend_pct", 0.0) or 0.0),
            }
        )

if __name__ == "__main__":
    run()
```

ДОДАТОК В. Фрагменти фронтенду та безпека

api.ts (виклик REST API)

```
/* src/api.ts */
export async function postRecommend(token: string, payload: any) {
  const res = await fetch("/api/recommend/", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${token}`,
    },
    body: JSON.stringify(payload),
  });

  if (!res.ok) {
    const txt = await res.text();
    throw new Error(txt || "Помилка запиту рекомендацій");
  }
  return res.json();
}
```

Questionnaire.tsx (анкета)

```
/* src/pages/Questionnaire.tsx */
import React, { useState } from "react";
import { postRecommend } from "../api";

const initial = {
  math_score: 10,
  physics_score: 9,
  ukr_lang_score: 11,
  eng_level: "B1",
  priority: "salary",
  interest_tech: true,
  interest_human: false,
  interest_nature: false,
  interest_creative: false,
};

export default function Questionnaire({ token, onDone }: { token: string; onDone: (data: any) => void }) {
  const [form, setForm] = useState<any>(initial);

  const set = (k: string, v: any) => setForm((p: any) => ({ ...p, [k]: v }));

  async function submit(e: React.FormEvent) {
    e.preventDefault();
    const data = await postRecommend(token, form);
    onDone(data);
  }

  return (
    <form onSubmit={submit}>
      <h2>Анкета абітурієнта</h2>
      <label>Математика (1–12)</label>
      <input type="number" min={1} max={12} value={form.math_score} onChange={(e) => set("math_score",
Number(e.target.value))} />
      <button type="submit">Отримати рекомендації</button>
    </form>
  );
```

```
}
```

Results.tsx (результати)

```
/* src/pages/Results.tsx */
import React from "react";

export default function Results({ rec }: { rec: any }) {
  const top = rec?.payload?.top || [];
  return (
    <div>
      <h2>Топ-5 рекомендацій</h2>
      <ol>
        {top.map((x: any) => (
          <li key={x.specialty_code}>
            <b>{x.specialty_name}</b> — FinalScore: {x.final_score}
          </li>
        ))}
      </ol>
    </div>
  );
}
```

Модульний тест API

```
# tests/test_recommend_api.py
import pytest
from django.contrib.auth.models import User
from rest_framework.test import APIClient

@pytest.mark.django_db
def test_recommend_flow():
    user = User.objects.create_user(username="demo", password="demo12345")
    client = APIClient()
    assert client.login(username="demo", password="demo12345")

    payload = {
        "math_score": 10, "physics_score": 9, "ukr_lang_score": 11,
        "eng_level": "B1", "priority": "salary",
        "interest_tech": True, "interest_human": False, "interest_nature": False, "interest_creative": False
    }

    r = client.post("/api/recommend/", payload, format="json")
    assert r.status_code == 201
    data = r.json()
    assert "payload" in data and "top" in data["payload"]
```

Фрагмент налаштувань безпеки (settings.py)

```
# app/settings.py (fragment)
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
X_FRAME_OPTIONS = "DENY"
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
```