ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтелектуальна рекомендаційна система для персоналізованого підбору вакансій у вебзастосунку»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ігор КНИШ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62
Ігор КНИШ

Керівник: Світлана ШЕВЧЕНКО
канд. пед. наук., доцент

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Книшу Ігору Олександровичу

1. Тема кваліфікаційної роботи: «Інтелектуальна рекомендаційна система для персоналізованого підбору вакансій у вебзастосунку»

керівник кваліфікаційної роботи Світлана Шевченко, канд. пед. наук

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з питань побудови рекомендаційних систем, методів багатокритеріального аналізу та персоналізації даних; статистичні дані ринку вакансій фізичної праці; описові характеристики вакансій (вік, місто, рівень оплати, клас небезпеки); вимоги до персоналізованого пошуку роботи; технічна документація фреймворку Django та супутніх вебтехнологій; структура тестових наборів вакансій, використаних для експериментів; результати тестування алгоритмів ранжування та порівняльного аналізу з класичними методами фільтрації.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- Проаналізувати сучасні підходи до персоналізованих рекомендацій та особливості підбору вакансій для фізичної праці.
- Запропонувати математичну модель оцінювання вакансій з ваговими коефіцієнтами та інтегральним рейтингом.
- Розробити метод комбінованого відбору: строгі фільтри (вік) + інтелектуальне ранжування за містом, оплатою та класом небезпеки.
- Реалізувати програмний модуль у Django з фільтрацією, розрахунком рейтингу та підтримкою неповних даних.
- Провести тестування, яке підтвердило підвищення релевантності результатів і швидкості пошуку.
- Надати рекомендації щодо розвитку: автоматичне визначення ваг та розширення критеріїв.

5. Перелік ілюстративного матеріалу: *презентація*

1. Порівняння з наявними рішеннями
2. Методика функціонування системи рекомендацій вакансій
3. Математична модель інтелектуального модуля методики
4. Діаграма використання
5. Блок схема алгоритму використання вакансій
6. Програмні засоби реалізації
7. Екранні форми додатку
8. Результат виведення потрібної вакансії
9. Порівняльний аналіз результатів моделювання

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	02.10-05.10	
2	Вивчення матеріалів щодо сучасних рекомендаційних систем та методів персоналізованого підбору вакансій.	06.10-09.10	
3	Дослідження алгоритмів рекомендацій (контентних, колаборативних, гібридних) та методів оцінювання релевантності.	10.10-13.10	
4	Аналіз впливу вагових коефіцієнтів і критичних параметрів (вік, місто, небезпека, оплата) на точність рекомендацій.	14.10-20.10	
5	Дослідження технологій реалізації вебзастосунків для систем рекомендацій (Python, Django, SQLite, HTML/CSS/JS).	21.10-28.10	

6	Розроблення та інтеграція математичної моделі персоналізованого підбору вакансій у вебзастосунок.	29.10-12.11	
7	Оформлення роботи: вступ, висновки, реферат	13.11-20.11	
8	Розробка демонстраційних матеріалів	20.11-27.11	
9	Попередній захист роботи	28.11	

Здобувач вищої освіти _____ Книш Ігор
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана Шевченко
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: _73__ стор., _7_ табл., _11__ рис., 37 джерел.

Мета роботи – удосконалення процесу підбору вакансій для користувача шляхом розроблення персоналізованих рекомендацій.

Об'єкт дослідження – процес функціонування рекомендаційних систем для пошуку вакансій фізичної праці.

Предмет дослідження – методи та алгоритми інтелектуального рекомендаційного аналізу, які дозволяють формувати персоналізовані пропозиції вакансій на основі поведінкових, контекстних та професійних даних користувача.

У роботі використано комплекс методів, що поєднують аналіз поведінки користувачів, методи багатокритеріального оцінювання, алгоритми ранжування та підходи до побудови персоналізованих рекомендаційних систем. Застосовано вагові коефіцієнти для моделювання впливу окремих критеріїв на релевантність вакансій (місто, рівень оплати, клас небезпеки, відповідність віковим обмеженням), а також алгоритми сортування й нормалізації значень для визначення пріоритетності результатів. Для реалізації функціоналу використано архітектурні підходи Django, об'єктно-орієнтовану структуру моделей, механізми обробки користувацького вводу та модулі серверної логіки.

Проведено аналіз сучасних методів підбору вакансій і принципів роботи рекомендаційних систем, включно з контентно-орієнтованими, колаборативними та гібридними моделями, а також алгоритмами, які застосовуються на популярних онлайн-платформах. Виявлено їхні обмеження у контексті фізичної праці: слабке врахування вікових характеристик, відсутність оцінки рівня небезпеки, недостатня індивідуалізація рекомендацій та відсутність персоналізованого ранжування за кількома критеріями одночасно.

Розроблено та оптимізовано метод персоналізованого підбору вакансій на

основі вагових коефіцієнтів і комбінованого критеріального аналізу. Запропонована математична модель дозволяє враховувати різний вплив характеристик вакансії на підсумковий рейтинг та адаптивно оцінювати пропозиції при неповних або часткових вхідних даних. Побудовано багаторівневу логіку: строгий фільтр (вікові обмеження) та м'який інтелектуальний аналіз (місто, оплата, небезпека). Реалізовано серверний модуль ранжування, що формує інтегральний показник корисності вакансії та сортує результати у порядку зменшення релевантності.

Проведено експериментальне тестування розробленого методу у декількох сценаріях з різними комбінаціями параметрів користувача. Результати демонструють покращення точності рекомендацій у порівнянні зі звичайним пошуком через фільтри, зменшення часу на вибір вакансії, підвищення релевантності перших позицій у списку та коректну роботу алгоритму за умови неповних вхідних даних. Також підтверджено, що запропонований підхід є більш ефективним для сегменту фізичної праці, оскільки враховує специфічні параметри (небезпечність, вікові обмеження), які традиційні платформи часто ігнорують.

Отримані результати підтверджують ефективність впровадженого алгоритму та демонструють перспективність його подальшого розвитку. Подальші дослідження можуть бути спрямовані на впровадження машинного навчання для автоматичного визначення вагових коефіцієнтів, розширення набору критеріїв оцінки вакансій, а також інтеграцію поведінкових факторів користувача для підвищення інтелектуальності системи.

КЛЮЧОВІ СЛОВА: РЕКОМЕНДАЦІЙНА СИСТЕМА, ПЕРСОНАЛІЗАЦІЯ, РАНЖУВАННЯ ВАКАНСІЙ, ВАГОВІ КОЕФІЦІЄНТИ, ФІЛЬТРАЦІЯ ДАНИХ, БАГАТОКРИТЕРІАЛЬНИЙ АНАЛІЗ, ФІЗИЧНА ПРАЦЯ, DJANGO, ІНТЕЛЕКТУАЛЬНИЙ АЛГОРИТМ.

ABSTRACT

Text part of the master's qualification work: _73_ pages, _7_ pictures, _11_ table, _37_ sources.

The purpose of the work is to improve the job-matching process for users by developing personalized recommendations.

Object of research – is the functioning process of recommendation systems for finding manual-labor vacancies.

Subject of research – is the methods and algorithms of intelligent recommendation analysis that enable the formation of personalized job proposals based on user behavioural, contextual, and professional data.

A set of methods was used in the work, combining user behaviour analysis, multi-criteria evaluation techniques, ranking algorithms, and approaches to building personalized recommendation systems. Weight coefficients were applied to model the influence of individual criteria on vacancy relevance (city, salary level, danger class, compliance with age limits), as well as sorting and normalization algorithms for determining the priority of results. Django architectural approaches were used to implement the functionality, including an object-oriented model structure, user input processing mechanisms, and backend logic modules.

An analysis of modern job-matching methods and principles of recommendation systems was carried out, including content-based, collaborative, and hybrid models, as well as algorithms used by popular online platforms. Their limitations in the context of manual labor were identified: insufficient consideration of age characteristics, absence of danger-level evaluation, low level of recommendation individualization, and lack of personalized multi-criteria ranking.

A personalized job-matching method was developed and optimized based on weight coefficients and combined criteria analysis. The proposed mathematical model takes into account the different impact of vacancy characteristics on the final rating and adaptively evaluates offers when the input data is incomplete. A multi-level logic was constructed: strict filtering (age restrictions) and soft intelligent analysis (city, payment, danger). A backend ranking module was implemented, generating an integral usefulness score for each vacancy and sorting results in descending order of relevance.

Experimental testing of the developed method was carried out in several scenarios with different combinations of user parameters. The results show improved recommendation accuracy compared to standard filter-based search, reduced time required to choose a vacancy, higher relevance of the top positions, and correct algorithm operation with incomplete input data. It is also confirmed that the proposed approach is more effective for manual-labor segments, as it considers specific parameters (hazard level, age restrictions) often ignored by traditional platforms.

The obtained results confirm the effectiveness of the implemented algorithm and demonstrate the prospects for further development. Future research may focus on integrating machine learning to automatically determine weight coefficients, expanding the set of vacancy evaluation criteria, and incorporating user behavioural factors to increase the system's intelligence.

KEYWORDS: RECOMMENDATION SYSTEM, PERSONALIZATION, JOB RANKING, WEIGHT COEFFICIENTS, DATA FILTERING, MULTI-CRITERIA ANALYSIS, MANUAL LABOR, DJANGO, INTELLIGENT ALGORITHM.

ЗМІСТ

ВСТУП

Сучасний ринок праці характеризується високою динамікою, постійною зміною вимог роботодавців та зростанням кількості цифрових платформ для пошуку вакансій. Попри доступність інтернет-сервісів, процес самостійного добору роботи все частіше перетворюється на складне завдання, що вимагає аналізу сотень пропозицій та порівняння їх з індивідуальними можливостями кандидата. Особливо це стосується фізичної праці, де вакансії часто мають специфічні обмеження: вікові рамки, рівень небезпеки, територіальні умови, мінімальну оплату.

Більшість популярних платформ застосовують виключно базові фільтри, не пропонуючи механізмів інтелектуальної персоналізації. Через це користувач отримує значну кількість нерелевантних вакансій, що збільшує час пошуку та не гарантує підбору оптимального варіанту. Водночас традиційні моделі рекомендацій у сфері послуг та медіа не враховують критичних параметрів доступності вакансій для конкретного кандидата. Таким чином, виникає потреба у створенні системи, здатної поєднати жорсткі обмеження з адаптивним ранжуванням пропозицій. Цим і підтверджена актуальність та важливість даного дослідження.

Мета роботи – удосконалення процесу підбору вакансій для користувача шляхом розроблення персоналізованих рекомендацій.

Об’єкт дослідження – процес формування рекомендацій у системах підбору вакансій фізичної праці.

Предмет дослідження – методи та алгоритми інтелектуального рекомендаційного аналізу, які дозволяють формувати персоналізовані пропозиції вакансій на основі поведінкових, контекстних та професійних даних користувача.

Для досягнення мети необхідно розв’язати такі завдання:

- проаналізувати існуючі підходи до рекомендацій у сервісах пошуку роботи та виявити їх обмеження щодо врахування характеристик фізичної праці;
- сформулювати вимоги до системи персоналізованого підбору вакансій з урахуванням вікових обмежень та ключових критеріїв релевантності;
- розробити математичну модель ранжування вакансій на основі вагових коефіцієнтів та формалізації показників відповідності;
- виконати програмну реалізацію інтелектуального модуля рекомендацій у вебзастосунку з використанням сучасних технологій;
- провести тестування алгоритму на базі реальних сценаріїв взаємодії користувача з системою та оцінити ефективність за показниками релевантності та швидкості пошуку;

- здійснити порівняльний аналіз результатів роботи інтелектуальної моделі з класичним фільтраційним підходом.

Ця робота базується на використанні методів системного аналізу, математичного моделювання, багатокритеріальної оцінки, а також методів програмної інженерії для реалізації вебзастосунку.

Наукова новизна одержаних результатів. Запропоновано метод персоналізованого підбору вакансій, що поєднує строгий відбір за віковими обмеженнями з гнучким багатокритеріальним ранжуванням, що підвищує якість рекомендацій у сфері фізичної праці.

Практичне значення одержаних результатів. Розроблено вебзастосунок, який дозволяє автоматизувати пошук вакансій, скоротити час прийняття рішення та підвищити точність пропозицій за рахунок урахування індивідуальних параметрів користувача.

Апробація результатів магістерської роботи.
Результати роботи були представлені на конференціях та підтвердили актуальність і практичну цінність запропонованого підходу:

1. Книш І.О., Шевченко С.М. Інтелектуальний модуль підбору вакансій у вебзастосунку для пошуку роботи // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року. Збірник тез. – К.: ДУІКТ, 2025. – С. 75-77.
2. Книш І.О., Шевченко С.М. Інтелектуальний веб-застосунок для пошуку фізичної роботи з урахуванням сучасних телекомунікаційних технологій // Науково-практична конференція «Розвиток телекомунікацій», 26 травня 2025 року, Київ. – Київ: ДУІКТ, 2025. − С. 25-27

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Рекомендаційні системи

Рекомендаційні системи є одним з ключових компонентів сучасних інформаційних технологій, що забезпечують інтелектуальний підхід до обробки даних та автоматичного формування персоналізованих пропозицій для користувача. В умовах різкого збільшення кількості цифрових даних та вакансій на ринку праці, рекомендаційні системи стають критично важливим елементом вебзастосунків, допомагаючи користувачеві швидко знаходити необхідну інформацію без потреби переглядати сотні нерелевантних об’єктів.

Сьогодні такі системи широко застосовуються у сферах електронної комерції, медіастрімінгу, соціальних мереж, освітніх платформ, служб пошуку роботи та на ринку рекрутингу. Основна ідея рекомендаційних систем полягає в автоматичному аналізі профілю користувача, історії взаємодії, явних та неявних уподобань, а також параметрів об’єктів, що дозволяє формувати індивідуально підібраний набір варіантів.

У контексті пошуку роботи рекомендаційні системи виконують функцію інтелектуального посередника між роботодавцем та шукачем. Кандидат отримує лише ті вакансії, які відповідають його професійним, демографічним та особистим параметрам. Це дозволяє значно скоротити час пошуку, підвищити якість збігів та ефективність взаємодії.

Хоча рекомендаційні системи активно розвиваються лише останні 20 років, теоретичні основи закладались ще у 1970-х роках — у рамках теорії інформаційного пошуку та статистичних моделей схожості документів. Перші прототипи з’явилися у 1990-х, коли вчені Массачусетського технологічного

інституту розробили перші моделі колаборативної фільтрації. У цей період з’явилися:

- перші моделі "Users like you also liked…";
- алгоритми рекомендацій у електронних бібліотеках;
- системи рекомендацій музики (Tapestry, GroupLens).

Потім, у 2000-х, рекомендаційні системи стали ключовим елементом Amazon, YouTube, Netflix, Facebook та Google.

У 2010–2024 роках рекомендаційні системи стали базою:

- соцмереж (Facebook, TikTok);
- маркетплейсів (Amazon, eBay, AliExpress);
- стрімінгових платформ (Netflix, Spotify);
- HR-платформ (LinkedIn, Indeed, Google Jobs).

Вони перетворилися зі звичайного фільтра у повноцінний інтелектуальний механізм на основі машинного навчання, глибинних нейронних мереж та аналізу поведінкових даних [1].

Рекомендаційна система у сфері пошуку роботи вирішує такі задачі:

- виявлення релевантних вакансій – на основі характеристик кандидата та параметрів вакансії;
- зменшення інформаційного навантаження – користувачу не потрібно переглядати всі вакансії вручну;
- персоналізація – зміна порядку відображення вакансій згідно з індивідуальними уподобаннями;
- врахування комплексних факторів – вік, місто, рівень небезпеки, оплата — усе це впливає на підсумкову релевантність;
- підвищення якості збігів – вакансії ранжуються за важливістю для конкретного користувача;
- оптимізація взаємодії на платформі – користувач швидше знаходить підходяще місце роботи → зростає задоволеність сервісом.

Типова рекомендаційна система складається з таких елементів:

1. Модуль збору даних:

- вік, місто, стаття витрат, досвід;
- історію переглядів;
- вибрані категорії;
- критерії фільтрації.

2. Модуль попередньої фільтрації. Він усуває неревалентні або недопустимі вакансії. У даному дослідженні – це строгий фільтр віку. Якщо вік користувача не підходить по мінімальному або максимальному віку, то він не вказується.

3. Модуль оцінювання (рейтингова система). Тут використовуються вагові коефіцієнти. Критерії, по яким обраховується коефіцієнт:

- місто;
- клас небезпеки;
- зарплата.

У сучасних цифрових платформах рекомендаційні системи відіграють ключову роль, оскільки вони дозволяють користувачеві отримувати персоналізований контент, не витрачаючи час на ручний пошук інформації. Їхня основна мета полягає в тому, щоб автоматично визначити, які об’єкти — товари, відео, новини чи вакансії — найбільше відповідають потребам і характеристикам конкретної людини. Залежно від підходу до аналізу даних такі системи працюють по-різному, і в науковій літературі прийнято виділяти три основні типи: контентно-орієнтовані, колаборативні та гібридні [2, 3].

Перший тип — контентно-орієнтовані рекомендаційні системи. В їхній основі лежить принцип порівняння характеристик об’єктів із параметрами, інтересами чи потребами користувача. Якщо говорити про сферу працевлаштування, то система порівнює такі елементи, як місто, рівень оплати, допустимий вік кандидата, рівень небезпеки, графік роботи чи вимоги до кваліфікації. Алгоритм працює за доволі простою логікою: якщо властивості вакансії збігаються з параметрами, які задав користувач, така вакансія вважається релевантною. Саме тому контентно-орієнтований підхід найчастіше використовується на сайтах пошуку роботи, особливо тоді, коли кількість

користувачів ще недостатня, щоб робити висновки на основі поведінкових даних [4].

Перевагою такого підходу є його точність та простота. Системі не потрібно аналізувати складну поведінку інших користувачів або мати масивну базу даних. Достатньо мати структуровані характеристики вакансій і невеликий набір параметрів від кандидата. Недоліком є те, що система «бачить» лише явні параметри. Вона не розуміє, наприклад, прихованих уподобань користувача або того, які вакансії він переглядав раніше. Проте саме контентно-орієнтована модель є оптимальною для проєктів, де важливими є жорсткі критерії: вікові обмеження, регіон, безпека умов праці. У таких випадках логічно спочатку відсіювати вакансії, що не відповідають базовим вимогам, а потім ранжувати інші пропозиції.

Другий тип — колаборативні рекомендаційні системи [5]. На відміну від контентних, вони не аналізують властивості об’єктів. Натомість вони вивчають поведінку інших користувачів — те, що вони переглядали, позначали як обране або на що подавали заявки. Основна ідея полягає в тому, що різні люди можуть мати подібні інтереси, навіть якщо ці інтереси не описані явно. Наприклад, якщо велика група користувачів із близькими профілями активно цікавиться певними вакансіями, система може рекомендувати ці вакансії новому користувачу з аналогічними характеристиками. Тобто рекомендація ґрунтується не на тому, що це за вакансія, а на тому, як із нею взаємодіяли інші люди.

Цей підхід широко застосовується у стрімінгових сервісах, таких як Netflix або YouTube, де поведінка мільйонів користувачів дозволяє будувати точні моделі вподобань. Однак у сфері пошуку роботи колаборативні системи мають суттєві обмеження [6]. По-перше, вакансії мають дуже різну природу та різні вимоги, що робить поведінкові порівняння менш інформативними. По-друге, на молодих або спеціалізованих платформах часто просто немає достатньої кількості поведінкових даних, щоб будувати якісні рекомендації. А по-третє, вакансії часто мають обмеження (наприклад, за віком), які не дозволяють застосовувати логіку «люди, схожі на вас, обирали це». Саме тому колаборативний підхід є менш придатним для

системи персоналізованого підбору вакансій, де ключову роль відіграють параметри, задані самим користувачем.

Третім, найрозвинутішим типом є гібридні рекомендаційні системи. Вони поєднують можливості обох попередніх методів, а також можуть включати додаткові евристики, математичні моделі, вагові коефіцієнти, логічні правила та інші інтелектуальні механізми. Завдяки цьому гібридні системи здатні досягати набагато вищої точності, адже вони використовують одночасно кілька джерел інформації [7]. Наприклад, вони можуть спочатку застосувати строгі обмеження на основі параметрів вакансії, а потім — ранжувати результати за ступенем відповідності користувачу, враховуючи його інтереси, попередні дії та статистику платформи.

Так працюють сучасні великі HR-платформи, зокрема LinkedIn та Google Jobs. Вони комбінують аналіз профілю кандидата, історії його переглядів, параметрів вакансій та поведінку інших користувачів. Це дозволяє створювати точні, адаптивні рекомендації, які постійно покращуються з часом. Проте реалізація таких систем вимагає великих обсягів даних, складних алгоритмів та значних обчислювальних ресурсів [8].

У контексті вебзастосунку, який розробляється в межах дипломної роботи, використовується спрощений гібридний підхід. Він поєднує контентний аналіз вакансій із системою вагових коефіцієнтів, що дозволяє оцінювати кожну вакансію не лише за строгими критеріями, а й за ступенем придатності. Такий підхід забезпечує баланс між точністю й простотою, що робить систему зрозумілою для користувача та ефективною з погляду реалізації.

В таблиці 1.1 представлено порівняльний аналіз усіх методів рекоменданційних систем.

Таблиця 1.1.

Порівняльний аналіз методів рекомендаційних систем

Тип рекомендаційної системи	На основі яких даних працює	Сильні сторони	Слабкі сторони	Доцільність використання у вакансіях
Контентно-орієнтована	Параметри вакансії та профіль користувача	Точність при чітких критеріях, проста реалізація	Не враховує поведінкові патерни	Висока — особливо коли важливі строгі обмеження (вік, місто)
Колаборативна	Дії інших користувачів, схожість поведінки	Добре виявляє приховані інтереси	Потрібні великі масиви поведінкових даних	Низька — вакансії надто різні, поведінкові дані обмежені
Гібридна	Комбінація параметрів, поведінки й правил	Найточніший та адаптивний підхід	Складніша реалізація, потреба в даних	Дуже висока — використовується у великих HR-платформах

1.2 Архітектура та класифікація рекомендаційних систем

Рекомендаційні системи є багаторівневими інформаційними комплексами, які поєднують методи аналізу даних, алгоритмічні моделі, механізми зберігання інформації та засоби взаємодії з користувачем.[9] Незважаючи на зовнішню простоту — відображення персоналізованого списку об’єктів — внутрішня структура таких систем включає низку взаємопов’язаних компонентів, кожен з яких виконує власну функцію і впливає на якість кінцевих рекомендацій. Для

глибшого розуміння принципів побудови рекомендаційних технологій необхідно розглянути їх загальну архітектуру та основні класифікаційні підходи [10].

У своїй основі рекомендаційна система функціонує як окремий модуль інформаційної інфраструктури, який отримує дані від зовнішніх джерел, обробляє їх за певними алгоритмами і формує впорядкований набір об’єктів. Типовий життєвий цикл рекомендацій охоплює етапи збору даних, їх фільтрації та нормалізації, формування ознак, виконання алгоритму ранжування та передачі результатів користувачеві. На першому етапі система отримує інформацію про об’єкти, які можуть бути рекомендовані. Це можуть бути вакансії, товари, фільми чи будь-які інші сутності. Важливу роль відіграє те, у якій формі і з якою частотою відбувається оновлення цих даних. Системи, що працюють у режимі високої динаміки, потребують механізмів оперативної обробки, тоді як інші можуть використовувати періодичне пакетне оновлення.

У процесі попередньої обробки дані приводяться до уніфікованого вигляду: нормалізуються числові значення, усуваються дублікати, виправляються логічні невідповідності. Саме на цьому етапі формується інформаційна база для подальшого аналізу [11]. Після цього система переходить до моделювання ознак, що є фундаментальним елементом архітектури рекомендаційних технологій. Ознаки можуть мати як змістовний характер — наприклад, атрибути вакансії, жанрові класифікації, види діяльності, — так і похідний, якщо вони формуються алгоритмічно у результаті аналізу великих масивів даних. У традиційних системах ознаки визначаються вручну, тоді як сучасні моделі дедалі частіше використовують алгоритмічні методи для автоматичного формування більш узагальнених і багатовимірних представлень [12].

Ключовою складовою архітектури є модуль рекомендаційного алгоритму, який безпосередньо визначає, які об’єкти будуть запропоновані користувачеві [13]. У класичних реалізаціях цей модуль спирається на логічні правила, вагові коефіцієнти, евристичні механізми або прості статистичні моделі. У складніших системах використовуються колаборативні фільтри, моделі прихованих факторів, нейронні мережі та інші алгоритми машинного навчання [14 – 15]. Незалежно від

вибраного підходу, модуль рекомендацій працює всередині загальної архітектури і взаємодіє з іншими компонентами, що забезпечують йому дані та контекст.

Архітектура рекомендаційної системи традиційно поділяється на дві великі групи: онлайн-процесинг і офлайн-процесинг. Онлайн-модуль відповідає за формування рекомендацій у реальному часі. Він має працювати швидко, бути оптимізованим і доступним під високими навантаженнями. Офлайн-модуль, у свою чергу, виконує складні та ресурсомісткі операції: аналізує статистику взаємодії користувачів, оновлює моделі, обчислює нові вагові коефіцієнти, виконує тренування алгоритмів. Такий поділ дозволяє досягти стабільної роботи системи навіть у складних умовах, поєднуючи точність глибинного аналізу з оперативністю відповіді [16].

З точки зору організації інформаційної інфраструктури рекомендаційні системи можуть мати різні архітектурні моделі. У централізованій архітектурі вся логіка обробки даних розміщена на одному сервері або у єдиному обчислювальному середовищі. Такий підхід характерний для простих або невеликих систем, де обсяги даних та кількість користувачів не створюють значного навантаження. У розподілених системах обробка даних виконується на різних серверах, а мікросервісний підхід дозволяє виділити окремі частини системи — наприклад, модуль збору даних, модуль обробки та модуль рекомендацій — у самостійні компоненти. Це підвищує масштабованість, гнучкість та надійність інфраструктури.

Окрему увагу варто приділити класифікації рекомендаційних систем за типом використовуваних даних та характером взаємодії з користувачем. Частина систем працює з експліцитними даними, тобто з інформацією, яку користувач вводить самостійно: параметрами профілю, фільтрами, відгуками, оцінками. Інші системи покладаються на імпліцитні дані — поведінку користувача, перегляди, тривалість взаємодії з об’єктами. Саме поєднання цих двох типів інформації дозволяє створювати більш точні моделі, які враховують як явні побажання, так і приховані інтереси.

Поширеною є класифікація рекомендаційних систем за принципом обчислення результатів: статичні та динамічні моделі. Статичні рекомендації ґрунтуються на заздалегідь підготовлених даних і не змінюються до наступного оновлення моделі [15]. Такий підхід є характерним для систем із низькою інтенсивністю змін. Динамічні моделі, навпаки, реагують на зміни миттєво: кожна нова взаємодія користувача впливає на формування рекомендацій. Динамічні підходи найбільш актуальні для сучасних веб-платформ, де дані оновлюються постійно, а актуальність інформації має критичне значення.

Архітектурні моделі рекомендаційних систем також відрізняються за способом інтеграції з іншими сервісами. Одні системи працюють як внутрішні компоненти великих програмних комплексів, інші — як зовнішні сервіси, що взаємодіють із застосунком через API. Така організація дозволяє використовувати рекомендаційні моделі в різних контекстах, забезпечує універсальність та можливість масштабування.

Для візуального уявлення наведена базова структурна діаграма архітектури рекомендаційної системи зображена на рисунку 1.1.

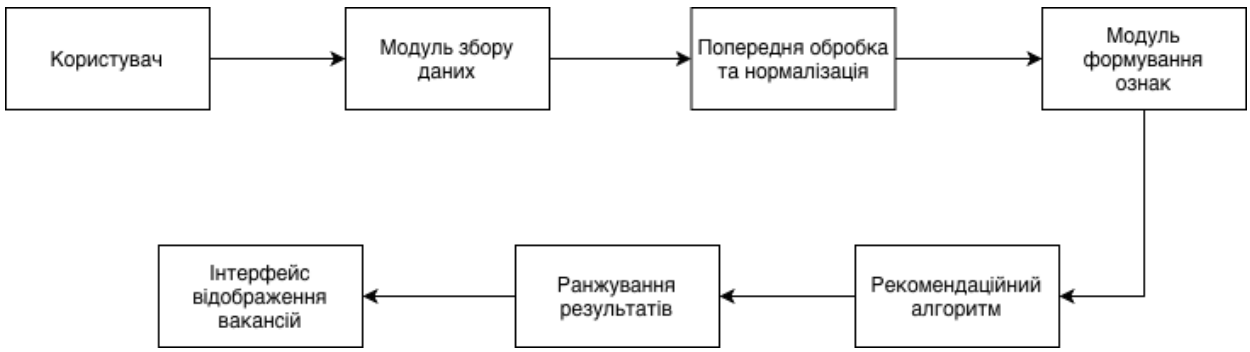


Рис. 1.1 – структурна діаграма рекомендаційної системи

Таким чином, архітектура рекомендаційної системи відображає комплексність процесу формування персоналізованих рекомендацій. Вона охоплює багаторівневу обробку даних, алгоритмічні моделі та організаційні взаємозв’язки між компонентами. Аналіз архітектурних підходів створює теоретичну основу для подальшого дослідження методів персоналізованого підбору вакансій та дозволяє

коректно інтегрувати рекомендаційні модулі у структуру веб-застосунку, описану в наступних розділах.

1.3 Алгоритми штучного інтелекту та їх роль у рекомендаційних системах

Сучасні веб-застосунки дедалі частіше інтегрують елементи штучного інтелекту, оскільки саме вони дозволяють найточніше адаптувати рекомендації до потреб користувача. Незважаючи на те, що мій веб-застосунок на початковому етапі розвитку використовує класичну Побудова моделі на основі машинного навчання потенційно могла б забезпечити значно кращі результати, оскільки такі системи здатні автоматично виявляти складні залежності між параметрами, вчитися на історичних даних та прогнозувати поведінку. Проте їхня ефективність безпосередньо залежить від обсягу даних, а також від того, наскільки різноманітними є профілі користувачів та вакансії [24]. Розроблений веб-застосунок наразі працює з обмеженим набором даних, тому використання складних моделей глибокого навчання було б недоцільним на даному етапі. З іншого боку, структурованість вакансій дає змогу ефективно застосовувати простіші, але надійні алгоритми на основі вагових коефіцієнтів.

Таким чином, аналіз переваг і недоліків існуючих методів персоналізованого підбору вакансій дозволив мені визначити оптимальний підхід для реалізації у власному веб-застосунку. Контентно-орієнтовані моделі забезпечують точне врахування формальних вимог, вагові системи дають гнучкість та можливість формувати ранжовані списки вакансій, а складніші алгоритми можуть бути інтегровані на наступних етапах розвитку проєкту [25]. Такий підхід дозволяє створити ефективну, зрозумілу та продуктивну рекомендаційну систему, адаптовану до реальних потреб користувачів.

математичну модель з ваговими коефіцієнтами, повноцінне розуміння методів штучного інтелекту залишається важливою складовою аналізу предметної галузі. Це пояснюється тим, що саме такі методи формують основу найпотужніших

рекомендаційних систем у світі, а в перспективі їх можна інтегрувати і в мою систему.

У загальному випадку алгоритми штучного інтелекту виконують функцію виявлення закономірностей у великих масивах даних. Коли йдеться про підбір вакансій, ці алгоритми здатні враховувати не тільки формальні параметри, такі як місто чи рівень оплати, але й поведінкові особливості користувачів — на які вакансії вони переходять, які зберігають, які приховують або ігнорують. Завдяки цьому інформаційна система не просто фільтрує вакансії за критеріями, а й намагається передбачити, які саме пропозиції можуть бути найбільш цікавими в майбутньому.

З-поміж різних методів штучного інтелекту особливе значення мають алгоритми машинного навчання [16]. Вони дозволяють побудувати модель, що поступово вдосконалюється, аналізуючи нові дані. Якщо система спостерігає, що користувач частіше переходить на вакансії з певними характеристиками, вона може збільшити вагу відповідних параметрів. І навпаки — якщо деякі критерії не впливають на поведінку користувача, система з часом зменшує їх вагу. Такий підхід формує динамічну модель, у якій рекомендації не є статичними, а змінюються разом із потребами та інтересами користувача.

Одним із найпоширеніших напрямів штучного інтелекту є нейронні мережі. У рекомендаційних системах вони виконують роль моделі, здатної самостійно знаходити складні залежності між параметрами. Кожен елемент вакансії — вік, місто, небезпека, оплата, опис, теги — може розглядатися як вхідний сигнал, який обробляється нейронною мережею. У її прихованих шарах формується набір внутрішніх ознак, що не мають прямого відображення у вихідних даних, але дозволяють системі набагато точніше оцінювати релевантність вакансії.

У класичному вигляді нейронна мережа для рекомендацій може бути описана формулою:

$$\hat{y} = f(Wx + b)$$

Де x — вектор параметрів вакансії та користувача, W — матриця ваг, b — зсув, а f — нелінійна функція активації, що дозволяє моделі враховувати складні залежності. Значення $\hat{y}$ такому випадку інтерпретується як прогноз релевантності вакансії.

Завдяки такій структурі нейронна мережа здатна опрацьовувати тисячі вакансій і визначати найбільш відповідні навіть тоді, коли користувач вводить мінімальний обсяг інформації. Наприклад, вона може виявити, що людину цікавлять вакансії з певним типом опису, навіть якщо вона ніколи не вказувала цього явно. Подібні функції роблять такі алгоритми незамінними у великих платформах, де поведінка користувачів може суттєво відрізнятися і формувати складні закономірності.

Водночас нейронні мережі мають і свої обмеження. Для їх роботи необхідні великі масиви даних, оскільки модель повинна вчитися на прикладах, а не на правилах. Це вимагає значної кількості користувачів та вакансій, а також серверних обчислень. Крім того, нейронні мережі нерідко критикують за “непрозорість”: важко пояснити, чому саме система запропонувала ту чи іншу вакансію. У розробленому веб-застосунку прозорість і пояснюваність є важливою характеристикою, тому на початковому етапі впровадження нейронних мереж недоцільне. Проте з розвитком платформи та збільшенням обсягу даних така можливість залишається перспективною.

Окрім нейронних мереж, у рекомендаційних системах використовуються й інші алгоритми машинного навчання. Наприклад, методи кластеризації можуть групувати вакансії за схожими характеристиками, формуючи кластери пропозицій. Дерева рішень та градієнтний бустинг здатні моделювати складні взаємозв’язки між параметрами та робити дуже точні прогнози. Методи латентних факторів дозволяють знаходити приховані ознаки, які пояснюють переваги користувача. Усі ці підходи мають спільну рису: вони не потребують ручного налаштування вагових коефіцієнтів, оскільки модель самостійно визначає, які параметри справді важливі.

У розробленому веб-застосунку така гнучкість може стати корисною на пізніших етапах, коли база користувачів стане досить великою. Тоді алгоритми штучного інтелекту зможуть переглядати історію взаємодії, оцінювати ефективність рекомендацій і самостійно коригувати ваги параметрів, роблячи систему точнішою з кожним днем. Однак навіть на сучасному етапі розвитку впровадження елементів штучного інтелекту посилює теоретичну основу розробки та демонструє перспективність вибраної архітектури.

Розгляд алгоритмів штучного інтелекту в контексті рекомендаційних систем дозволяє сформувати чітке бачення можливостей, які можуть бути інтегровані у мій веб-застосунок у майбутньому. Від класичних моделей до глибокого навчання, кожен метод має свої переваги й обмеження, але всі вони спрямовані на одне — створення системи, що дійсно розуміє користувача і здатна надати йому найбільш релевантні вакансії серед великої кількості доступних пропозицій.

2 АНАЛІЗ МЕТОДІВ ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ВАКАНСІЙ У РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ

2.1 Аналіз існуючих методів персоналізованого підбору вакансій у рекомендаційних системах

Сучасний ринок праці характеризується високою динамічністю, значним обсягом вакансій та широкою різноманітністю вимог до кандидатів. У таких умовах процес пошуку роботи дедалі частіше переноситься у цифровий простір, де ключову роль відіграють рекомендаційні системи. Їхнє завдання — не лише показати користувачу доступні оголошення, а й відібрати з великої кількості саме ті вакансії, які найбільше відповідають його досвіду, віку, регіону, умовам праці та очікуванням щодо оплати. Тому першочерговим кроком є аналіз методів, які застосовуються для персоналізованого підбору вакансій у сучасних вебзастосунках і HR-платформах.

Як правило, рекомендаційні системи базуються на аналізі численних факторів, які впливають на відповідність вакансії конкретному кандидатові. Однак способи, якими саме формується підбір, суттєво відрізняються залежно від вибраної моделі. В основі сучасних систем найчастіше лежать три ключові підходи: контентно-орієнтований підбір, колаборативна фільтрація та гібридні алгоритми. Кожен з них оперує різними механізмами аналізу даних, застосовує власні методи обчислення релевантності і формує унікальні принципи взаємодії користувача з системою.

Контентно-орієнтований підхід ґрунтується на ідеї аналізу змісту вакансій. Система порівнює атрибути кожної вакансії з характеристиками користувача: його профілем, навичками, містом проживання, віком та іншими параметрами.[17] Такий підхід найбільш близький до логіки персонального відбору, оскільки використовує виключно інформацію, притаманну конкретній особі. Система намагається зрозуміти, “хто є користувач” і які вакансії можуть відповідати його критеріям, незалежно від поведінки інших кандидатів. Цей метод дозволяє

уникнути ситуації, коли користувачу рекомендуються вакансії, популярні серед інших, але абсолютно нерелевантні саме йому. Саме тому контентно-орієнтовані системи широко застосовуються в сервісах, де важливою є точність відповідності формальних параметрів — віку, місця роботи, класу небезпеки чи рівня мінімальної оплати.

На відміну від цього підходу, методи колаборативної фільтрації формують рекомендації на основі поведінки інших користувачів платформи. У центрі цього механізму лежить припущення: якщо двоє користувачів мають подібні інтереси або однакові вподобання, то й вакансії, що зацікавили одного з них, з великою ймовірністю підійдуть іншому. Такий підхід використовується у великих платформах із масовою кількістю користувачів, де система може виявляти шаблони поведінки: вподобання, переходи, збережені вакансії, переглянуті пропозиції. Колаборативна фільтрація здатна враховувати приховані залежності, однак вона менш ефективна в тих системах, де важливими є жорсткі фільтри — наприклад, вікові обмеження або вимоги щодо фізичної придатності. Тому незважаючи на свою популярність, цей метод менш придатний для сервісів, які працюють із вакансіями фізичної праці, волонтерства чи високих ризиків, де частину оголошень кандидат просто не має права переглядати.

Гібридний підхід намагається об’єднати сильні сторони обох попередніх методів. У таких системах рекомендації формуються на основі змістовних параметрів вакансії, але при цьому враховується і колективний досвід інших користувачів. Гібридні моделі дозволяють зменшити недоліки кожного окремого підходу: вони менш схильні до “холодного старту”, коли інформація про нових користувачів обмежена, і здатні адаптуватися до змін у поведінці аудиторії. Утім, складність їхньої реалізації та необхідність значних обчислювальних ресурсів роблять їх менш поширеними в невеликих додатках, де пріоритетними є простота, швидкість і прозорість алгоритму.

Окремий інтерес становлять методи, що використовують вагові коефіцієнти для ранжування вакансій. На відміну від класичних моделей, які прагнуть знайти «ідеальний збіг», вагові системи оцінюють кожну характеристику за її важливістю.

Наприклад, місце роботи може мати середню вагу, клас небезпеки — нижчу, а мінімальна оплата — значно вищу. Це дозволяє системі враховувати специфіку користувача та будувати більш персоналізовані рекомендації навіть у ситуаціях, коли одна з умов не є абсолютною. На практиці це дає змогу відсортувати вакансії так, щоб спочатку відображалися найбільш релевантні, але й менш пріоритетні оголошення також залишалися доступними [18]. Подібна логіка особливо важлива для вебзастосунків, які прагнуть забезпечити користувача широким, але структурованим вибором.

Не можна не згадати й про методи, що базуються на машинному навчанні. Вони використовують приховані закономірності у великих наборах даних та можуть автоматично налаштовувати ваги або структуру рекомендацій залежно від статистики поведінки користувачів. Моделі цього типу здатні самостійно визначати, які характеристики впливають на релевантність найбільше, враховувати неочевидні залежності між параметрами та виявляти закономірності, невидимі при класичному аналітичному підході. Проте такі моделі потребують великого обсягу даних і часто складні у впровадженні. Для системи, яка працює з вакансіями фізичної праці та без значного потоку користувачів, застосування складних нейронних мереж може бути недоцільним і надмірним.

Таким чином, аналіз існуючих методів персоналізованого підбору вакансій демонструє значну кількість підходів, кожен з яких має власні переваги та обмеження. Контентно-орієнтовані моделі забезпечують чітку відповідність формальним критеріям, колаборативні — дають можливість знайти приховані схожості, а гібридні — поєднують обидва методи для досягнення збалансованого результату. Системи з ваговими коефіцієнтами дозволяють формалізувати вибір та зробити процес більш гнучким для користувача. Усі ці підходи формують наукову та практичну базу, на основі якої може бути побудована ефективна інтелектуальна система рекомендацій у розробленому вебзастосунку.

2.2 Аналіз переваг та недоліків методів персоналізованого підбору вакансій

Після вивчення існуючих підходів до побудови рекомендаційних систем важливо не лише описати їх принципи роботи, а й визначити сильні та слабкі сторони кожної групи методів. Такий аналіз дозволяє глибше зрозуміти, які моделі доцільно застосовувати саме у розробленому веб-застосунку, враховуючи його специфіку, обсяг даних, цільову аудиторію та особливості типів вакансій. Оцінка переваг і недоліків методів рекомендацій дає змогу не лише зробити обґрунтований вибір, а й визначити, які саме обмеження необхідно компенсувати власними алгоритмами або структурою інтерфейсу.

Контентно-орієнтовані підходи часто вважаються найпростіші в реалізації, але водночас надзвичайно ефективними у випадках, коли рекомендації мають базуватися на точному збігу атрибутів вакансії та користувача. Їхньою ключовою перевагою є здатність чітко враховувати об'єктивні параметри: вік, місто, клас небезпеки, мінімальний рівень заробітної плати. Вони також добре працюють у середовищах, де частина вакансій має жорсткі вимоги, яких кандидат не може порушити [19]. Саме тому контентні моделі ідеально підходять для мого веб-застосунку, де вік є одним із основних обмежуючих критеріїв, а параметри вакансій структуровані та однозначно визначені. Використання таких методів дозволяє уникати ситуацій, коли система пропонує користувачу недосяжні або недоступні вакансії.

Втім, контентний підхід має свої слабкі місця. Він повністю залежить від повноти заповнення профілю та коректності введених даних. Якщо користувач не вказує частину інформації або вводить її неточно, результати рекомендацій можуть бути нечіткими або надто обмеженими. Крім того, контентно-орієнтована система не здатна самостійно навчатися на поведінці користувача чи виявляти приховані подібності між вакансіями, які не очевидні за формальними критеріями. Вона не

використовує колективний досвід інших кандидатів, що позбавляє її можливості адаптуватися до змін динаміки ринку праці або переваг аудиторії [20].

Методи колаборативної фільтрації, навпаки, демонструють високу ефективність у випадках, коли саме поведінкові характеристики користувачів відіграють ключову роль. Такі підходи добре працюють у великих системах із великим потоком даних, де можна відстежувати популярність конкретних вакансій, закономірності вибору та кліки [21]. Але для мого веб-застосунку, який орієнтований на підбір вакансій з чіткою структурою параметрів та жорсткими фільтрами, колаборативні методи мають суттєві недоліки. Вони погано працюють у ситуаціях, де формальні обмеження, наприклад клас небезпеки або вікові рамки, є важливішими за історію взаємодій. Система колаборативної фільтрації може рекомендувати вакансії, які іншим користувачам здавалися привабливими, але вони можуть бути недоступні конкретній людині. Це робить метод менш придатним для застосування в умовах, де важливіше забезпечити юридичну чи фізичну відповідність кандидата, ніж орієнтуватися на загальні тенденції.

Гібридні моделі здобувають свої переваги завдяки поєднанню сильних сторін двох попередніх підходів [22]. Вони можуть частково компенсувати недоліки контентно-орієнтованих методів, додаючи можливість адаптації до поведінки користувача, і водночас уникати обмежень колаборативних механізмів, завдяки чіткому врахуванню атрибутів вакансій. Проте ці системи є складнішими в реалізації. Їхня побудова вимагає значних обчислювальних ресурсів, складнішої архітектури даних та більших витрат часу на навчання моделі. У рамках мого веб-застосунку використання гібридних методів могло б бути виправданим лише за умови значного збільшення користувацької бази чи потреби у глибшому аналізі поведінкових патернів, що наразі не є пріоритетним.

Окремий інтерес становлять моделі, що використовують систему вагових коефіцієнтів. У розробленому веб-застосунку я реалізував саме цей підхід, оскільки він дає змогу формувати рейтинг вакансій на основі важливості кожного параметра. Такий метод поєднує простоту з високою адаптивністю. Завдяки вагам я можу визначати, які параметри є ключовими, а які мають другорядний вплив.

Наприклад, заробітна плата може отримувати більшу вагу, ніж місто чи клас небезпеки, оскільки для більшості користувачів цей критерій є важливішим. Система ваг дозволяє створювати більш гнучкий механізм підбору: навіть якщо деякі параметри не збігаються ідеально, вакансія все одно може бути запропонована, якщо її загальна релевантність висока.

Проте цей метод також не позбавлений недоліків. Основною його слабкістю є залежність від ручного налаштування вагових коефіцієнтів [23]. Якщо ваги встановлені не зовсім коректно, результати ранжування можуть бути викривленими або недостатньо персоналізованими. Крім того, така система сама по собі не навчається на динаміці даних, тому для підвищення її точності в майбутньому може знадобитися інтеграція алгоритмів машинного навчання, що здатні автоматично коригувати ваги відповідно до поведінки користувачів.

Побудова моделі на основі машинного навчання потенційно могла б забезпечити значно кращі результати, оскільки такі системи здатні автоматично виявляти складні залежності між параметрами, вчитися на історичних даних та прогнозувати поведінку. Проте їхня ефективність безпосередньо залежить від обсягу даних, а також від того, наскільки різноманітними є профілі користувачів та вакансії [24]. Розроблений веб-застосунок наразі працює з обмеженим набором даних, тому використання складних моделей глибокого навчання було б недоцільним на даному етапі. З іншого боку, структурованість вакансій дає змогу ефективно застосовувати простіші, але надійні алгоритми на основі вагових коефіцієнтів.

Таким чином, аналіз переваг і недоліків існуючих методів персоналізованого підбору вакансій дозволив мені визначити оптимальний підхід для реалізації у власному веб-застосунку. Контентно-орієнтовані моделі забезпечують точне врахування формальних вимог, вагові системи дають гнучкість та можливість формувати ранжовані списки вакансій, а складніші алгоритми можуть бути інтегровані на наступних етапах розвитку проєкту [25]. Такий підхід дозволяє створити ефективну, зрозумілу та продуктивну рекомендаційну систему, адаптовану до реальних потреб користувачів.

2.3 Математична модель методу персоналізованого підбору вакансій в даному дослідженні

Створення інтелектуальної рекомендаційної системи для підбору вакансій у розробленому веб-застосунку потребувало формального математичного опису принципів її роботи. Такий опис дозволяє не лише реалізувати алгоритм у програмному коді, а й обґрунтувати логіку прийняття рішень, показати, як саме окремі параметри вакансії впливають на підсумковий рейтинг і чому певні оголошення опиняються вище за інші в списку рекомендацій [26].

У розробленій мною системі я поділяю всі парами, що впливають на підбір вакансій, на дві групи: критичні та вагові. До критичних належать такі характеристики, які визначають саму можливість роботи кандидата на певній посаді. Головним із них є вік користувача. Оскільки в оголошеннях про роботу, особливо фізичну, часто зазначаються мінімальні та максимальні вікові межі, цей критерій я розглядаю як строгий фільтр. Решта параметрів — місто, клас небезпеки та рівень оплати праці — не завжди є жорсткими обмеженнями, але істотно впливають на привабливість вакансії. Саме вони формують другу групу — вагові критерії, які беруть участь у розрахунку інтегральної оцінки вакансії.

Формально множину всіх вакансій я позначаю через V. Для кожної вакансії $v \in V$ у базі даних зберігаються, зокрема, такі параметри: мінімально допустимий вік $v_{\min\ \_age}$, місто , клас небезпеки та рівень оплати . Нехай вік користувача позначається через age. Тоді на першому етапі відбору я формую підмножину вакансій, які відповідають віковим обмеженням, за правилом

$$V^{(age)} = \left\{ v \in V \middle| v_{\min\_age} \leq age \leq v_{\max\ \_age} \right\}$$

Цей етап є обов’язковим, оскільки всі вакансії, що не входять до множини $V^{(age)}$, апріорі є недоступними для користувача і не повинні потрапляти навіть у

попередній список рекомендацій. Такий підхід дає змогу забезпечити коректність роботи системи та відповідність рекомендацій реальним обмеженням.

Після того як сформовано підмножину вакансій, які відповідають віку користувача, відбувається другий етап — обчислення інтегральної оцінки релевантності кожної вакансії. Для цього я використовую модель зваженої суми, у якій кожному критерію відповідає певна вага. Загальний рейтинг вакансії v я позначаю через $R(v)$. Він розраховується як

$$R(v) = \omega_C * C(v) + \omega_d * D(v) + \omega_p * P(v)$$

Де $C(v), D(v)$ та $P(v)$ — числові оцінки відповідності міста, класу небезпеки та рівня оплати відповідно, а $\omega_C, \omega_d, \omega_p$— вагові коефіцієнти цих критеріїв.

У розробленому веб-застосунку ваги обрано таким чином, щоб їх сума дорівнювала одиниці:

$$\omega_C, \omega_d, \omega_p = 1$$

Це дозволяє трактувати рейтинг $R(v)$ як нормовану величину в інтервалі від 0 до 1. Після аналізу специфіки предметної області та поведінкових очікувань користувача, я прийняв такі значення коефіцієнтів: для міста $\omega_C = 0.3$ - для небезпеки $\omega_d = 0.2$, для заробітної платні $\omega_p = 0.5$. Такий розподіл відображає пріоритетність факторів: найвагомішою є оплата праці, на другому місці — географічна доступність, і трохи меншу вагу має клас небезпеки. Ці коефіцієнти визначені за допомогою гугл опитування яке я провів на 23 користувачах.

Оцінка відповідності міста є найпростішою з точки зору формалізації. У моїй моделі я припускаю, що користувач у більшості випадків шукає роботу у власному місті, а вакансії з інших міст, як правило, є менш привабливими. Тому для міста я

використовую бінарну функцію відповідності. Якщо місто вакансії збігається з містом, вибраним користувачем, я задаю

$$C(v) = 1$$

Інакше

$$C(v) = 0$$

Таким чином, вакансії з іншого міста не відсіюються повністю, але втрачають ту частину рейтингу, яка відповідає ваговому внеску параметра міста.

Аналогічний підхід я використовую і для класу небезпеки. Якщо користувач обрав певний допустимий рівень ризику, то вакансії з тим самим класом небезпеки отримують

$$D(v) = 1$$

А всі інші – $D(v) = 0$ Така формалізація відображає ситуацію, коли безпека є важливою, але не абсолютно визначальною: якщо користувач не задав обмеження за небезпечністю або готовий розглядати різні варіанти, то параметр $D(v)$ може бути взагалі не задіяний у розрахунках. У будь-якому разі його внесок у підсумковий рейтинг є меншим, ніж внесок оплати.

Найцікавішим з точки зору математичної моделі є критерій, пов’язаний із рівнем заробітної плати. На відміну від міста й небезпеки, цей параметр є не дискретним, а числовим і може змінюватися в широкому діапазоні. Користувач у розробленому веб-застосунку може вказати бажану мінімальну оплату pay_{min} тоді кожну вакансію я оцінюю з точки зору того, наскільки її фактична оплата v_{min} перевищує або дорівнює заданому порогу. Для цього я використовую нормовану оцінку

$$P(v) = \begin{cases} 0, & \text{якщо } v_{pay} < pay_{min}, \\ \min\left(1,\ \dfrac{v_{pay} - pay_{min}}{pay_{min}}\right), & \text{якщо } v_{pay} \geq pay_{min}. \end{cases}$$

Якщо оплата є меншою за бажану — внесок цього критерію дорівнює нулю, і вакансія втрачає потенційну перевагу, пов'язану з оплатою. Якщо ж оплата дорівнює бажаній, значення $P(v)$ прямує до нуля, але при невеликому перевищенні порогу починає зростати. Коли оплата значно перевищує бажану, відношення $\left(v_{pay} - pay_{min}\right) / pay_{min}$ може стати більшим за одиницю, але я штучно обмежую максимум функції до 1. Це запобігає ситуації, коли одна вакансія з надзвичайно високою оплатою занадто сильно "перетягує" на себе вагу у рейтингу, ігноруючи важливі для користувача параметри.

Завдяки такому нормуванню параметр оплати стає співмірним із іншими критеріями і природно вкладається в інтервал [0; 1]. У поєднанні з ваговим коефіцієнтом $\omega_p = 0{,}5$ це означає, що внесок оплати у підсумковий рейтинг може досягати половини максимально можливої оцінки.

Таким чином, кожна вакансія, що пройшла віковий фільтр, описується трійкою чисел C(v), D(v), P(v) та відповідним значенням рейтингу R(v). Після того, як значення R(v) обчислено для всіх вакансій з множини V(age), я впорядковую їх за спаданням рейтингу. Отриманий список і є результатом персоналізованого підбору — на його початку розташовуються вакансії, які найкраще відповідають параметрам та побажанням користувача, а далі — менш релевантні, але все ще потенційно цікаві.

Таку математичну модель я обрав з кількох причин [27]. Вона залишається простою та прозорою, легко інтерпретується як розробником, так і користувачем, не вимагає великих обсягів навчальних даних, але при цьому враховує складну структуру переваг користувача. Я можу чітко пояснити, чому саме певна вакансія опинилася вище за іншу: або вона у потрібному місті, або має кращу оплату, або відповідає бажаному класу небезпеки, а частіше — через комбінацію усіх цих факторів.

Окремо варто зазначити, що така модель є придатною до подальшого розвитку [28]. У перспективі я можу змінювати значення вагових коефіцієнтів,

вводити проміжні рівні оцінки для міста чи небезпеки (наприклад, часткову відповідність), додавати нові параметри, такі як тип зайнятості чи графік роботи. Також можливим є розширення моделі за рахунок алгоритмів машинного навчання, які автоматично коригуватимуть ваги на основі історії взаємодії користувачів із системою. Проте вже у поточному вигляді розроблена математична модель забезпечує достатню ступінь персоналізації, узгоджується з логікою поведінки шукачів роботи та відповідає обраній темі дипломної роботи.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ ВАКАНСІЙ

3.1 Проєктування структури системи та алгоритму рекомендацій

У процесі створення інтелектуальної рекомендаційної системи для персоналізованого підбору вакансій я насамперед зосередився на тому, щоб побудувати логічну та цілісну архітектуру, у якій кожен компонент виконує чітко визначену функцію [29]. Моя мета полягала у тому, щоб алгоритм рекомендацій працював не просто як простий фільтр за кількома параметрами, а як адаптивна система, здатна оцінювати вакансії комплексно — з урахуванням вагових коефіцієнтів, особистих характеристик користувача та рівня релевантності кожної пропозиції.

На початковому етапі проєктування я визначив ключові елементи, які повинні взаємодіяти між собою: користувач, база вакансій, система фільтрації та інтелектуальний блок ранжування. У розробленому веб-застосунку цей ланцюг взаємодії започатковується з моменту, коли користувач вводить початкові дані — вік, місто, бажаний рівень оплати, ступінь прийнятного рівня небезпеки та інші характеристики, які можуть стати важливими під час вибору роботи. Важливо, що система допускає неповні дані: користувач може вказати лише частину параметрів, а інтелектуальний алгоритм адаптується до доступної інформації.

Під час формування структури системи я виділив два рівні обробки: **строгий фільтр** та **м’який інтелектуальний аналіз**. Строгий фільтр застосовується лише до тих параметрів, які не можна порушувати за жодних умов — наприклад, вікові обмеження, оскільки неповнолітні не можуть працювати на певних вакансіях, а у деяких випадках роботодавці встановлюють конкретний діапазон віку [30]. На

цьому етапі система одразу відсіює вакансії, які користувачу об’єктивно не підходять.

М’який інтелектуальний аналіз відбувається вже після цього. Його суть полягає у застосуванні вагових коефіцієнтів, які я інтегрував у алгоритм. Зокрема, місто, рівень оплати та ступінь небезпеки отримали окремі ваги, що дозволяє системі оцінювати вакансії не просто бінарно, а ступенево — надаючи більш релевантним варіантам вищу оцінку. Після цього вакансії впорядковуються за рейтингом, і користувач бачить найпріоритетніші пропозиції у верхніх позиціях списку. На рисунку 3.1 зображено блок-схему алгоритму рекомендації вакансій.

Рис. 3.1– блок-схема алгоритму рекомендації вакансій

З архітектурної точки зору система побудована як типова багаторівнева модель: на верхньому рівні знаходиться інтерфейс взаємодії, у якому користувач вводить дані; другий рівень відповідає за первинну обробку інформації та строгі фільтри; третій рівень — інтелектуальний блок, який здійснює ранжування; і нарешті останній рівень — візуалізація результату, де користувач отримує відсортовані вакансії. На рисунку 3.2 зображено діаграму розгалуження.

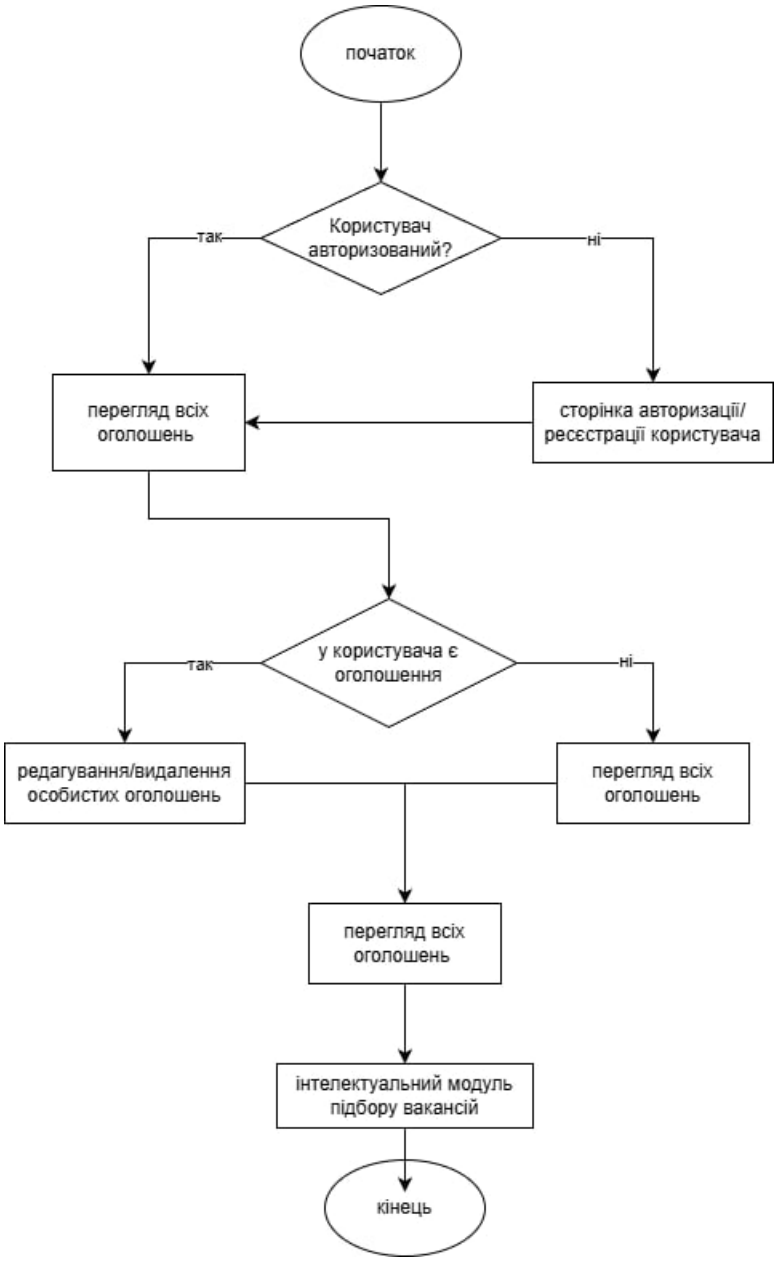


Рис. 3.2 – діаграма розгалуження

3.2 Опис математичної та програмної реалізації методу

Реалізація методу персоналізованого підбору вакансій вимагала побудови цілісної програмної структури, у якій математична модель рекомендацій органічно інтегрується в архітектуру веборієнтованої інформаційної системи. Така інтеграція є важливою з огляду на необхідність забезпечення узгодженої взаємодії між усіма компонентами системи, починаючи від зберігання даних і завершуючи формуванням рекомендацій для кінцевого користувача. В умовах реальної експлуатації вебзастосунку рекомендаційний модуль повинен працювати стабільно, швидко та коректно, оскільки саме від якості його функціонування залежить загальне враження користувача від платформи.

Для досягнення поставленої мети було застосовано багаторівневий підхід до побудови програмної архітектури. Такий підхід передбачає чітке розмежування функціональних обов’язків між рівнями системи, що спрощує процес розроблення, тестування та подальшого супроводу програмного продукту. Зокрема, окремо виділено рівень зберігання даних, рівень обробки клієнтських запитів, рівень бізнес-логіки рекомендаційного модуля та рівень представлення результатів у користувацькому інтерфейсі. Подібна структура відповідає сучасним підходам до розроблення веборієнтованих інформаційних систем і забезпечує їх масштабованість.

Основою серверної частини системи став фреймворк Django, який надає широкі можливості для побудови структурованих і надійних вебзастосунків [31][32]. Використання Django дозволило реалізувати архітектуру за принципом Model–View–Template, що сприяє логічному поділу системи на окремі компоненти та зменшує взаємозалежність між ними. Це, у свою чергу, позитивно впливає на читабельність коду, полегшує внесення змін і підвищує загальну якість програмної реалізації.

На рівні моделей реалізовано зберігання всієї необхідної інформації про вакансії. До таких даних належать вікові обмеження, рівень оплати праці, місто

розташування, клас небезпеки, а також інші допоміжні параметри, які можуть бути використані під час формування рекомендацій. Кожен параметр представлено у вигляді окремого поля моделі, що забезпечує структурованість даних та дозволяє ефективно використовувати їх у процесі обчислень. Такий підхід також полегшує розширення функціональності системи, оскільки додавання нових критеріїв не потребує суттєвих змін у наявній архітектурі.

Використання ORM Django дозволяє працювати з базою даних на високому рівні абстракції, не звертаючись безпосередньо до мови SQL. Це значно знижує ймовірність помилок, пов’язаних із формуванням запитів, та забезпечує переносимість системи між різними СУБД. Крім того, ORM-механізми сприяють оптимізації доступу до даних і дозволяють реалізувати фільтрацію та вибірку інформації безпосередньо на рівні бази даних.

Під час обробки запиту користувача система проходить через кілька логічно пов’язаних етапів. На першому етапі здійснюється попередня фільтрація вакансій за строгими критеріями, основним із яких є вікове обмеження. Цей етап реалізовано за допомогою ORM-фільтрів, що дозволяє одразу виключити з подальшого розгляду вакансії, які не можуть бути рекомендовані користувачу за формальними ознаками. Такий підхід значно зменшує обсяг даних, що підлягають подальшій обробці, та сприяє підвищенню продуктивності системи.

Після проходження строгого фільтра система переходить до етапу гнучкого оцінювання вакансій. На цьому етапі для кожної вакансії обчислюється рейтинговий показник, який відображає ступінь її відповідності параметрам користувача. Обчислення здійснюється в окремому програмному циклі, де кожен критерій враховується з урахуванням заданого вагового коефіцієнта. Така схема дозволяє моделювати різну важливість параметрів і адаптувати рекомендації до індивідуальних потреб користувача.

Особливу увагу під час реалізації цього етапу було приділено точності числових обчислень. Для всіх операцій, пов’язаних із вагами та коефіцієнтами, використано тип даних Decimal. Це дозволило уникнути похибок, притаманних використанню чисел з рухомою комою, та забезпечило коректність результатів

навіть при багаторазових обчисленнях. Такий підхід є особливо важливим у рекомендаційних системах, де накопичення навіть незначних похибок може впливати на кінцеве ранжування об’єктів.

Завершальним етапом обробки є сортування вакансій відповідно до отриманих рейтингових значень. Сортування реалізовано засобами мови Python у серверній частині застосунку, що забезпечує достатню гнучкість алгоритму. У разі потреби розробник може змінювати логіку ранжування, додавати нові критерії або коригувати вагові коефіцієнти без необхідності модифікації структури бази даних. Це робить систему адаптивною до змін вимог і спрощує її подальший розвиток.

Уся логіка персоналізованого підбору вакансій функціонує на серверному рівні, що дозволяє централізовано контролювати алгоритми рекомендацій і забезпечувати їх захист від несанкціонованого втручання. Результати обробки передаються до клієнтської частини у вигляді впорядкованого списку вакансій, готового до відображення в інтерфейсі користувача. Завдяки чіткому розмежуванню відповідальності між моделями даних, бізнес-логікою та шаблонами відображення вдалося інтегрувати рекомендаційний метод у загальну систему без ускладнення програмної архітектури.

Таблиця 3.1

Архітектурні компоненти

Компонент	**Основна функція**	**Особливості реалізації**
Моделі бази даних	Зберігання структурованих даних про вакансії	Чітко визначені поля для віку, міста, оплати, небезпеки
Модуль обробки запитів	Зчитування параметрів та формування вибірки	Обробка GET-параметрів, первинний строгий фільтр
Рекомендаційний блок	Обчислення рейтингу вакансій	Використання Decimal, адаптивність до змін ваг
Модуль сортування	Формування кінцевого результату	Впорядкування за рейтингом, генерація списку рекомендацій
Інтерфейс відображення	Інтерфейс відображення	Динамічна генерація списку вакансій у шаблонах

У підсумку реалізація методу поєднує математичну модель з інженерними рішеннями, що забезпечують точність, масштабованість та стабільність роботи рекомендаційного механізму. Така структура дозволяє системі функціонувати як повноцінний інтелектуальний інструмент підтримки вибору вакансій, а також створює умови для подальшого розвитку з використанням машинного навчання чи адаптивного аналізу поведінки користувачів.

3.3 Опис структури бази даних та зберігання даних

Структура бази даних є фундаментальним елементом функціонування рекомендаційної системи, оскільки саме у ній зберігається вся інформація, необхідна для формування персоналізованого списку вакансій.[34] У даному дослідженні було використано реляційну модель зберігання даних, що забезпечує чітку структуру, логічні зв’язки між сутностями та можливість швидкого виконання запитів, що особливо важливо під час роботи з рекомендаційними алгоритмами.

Основу складають три ключові сутності: **користувачі**, **вакансії** та **додаткові категорії оголошень**. Для кожної з цих сутностей визначено набір атрибутів, які відображають їх характеристики та забезпечують можливість подальшої обробки даних у рекомендаційному блоці. Модель користувача містить ідентифікаційні дані, такі як ім’я, електронна пошта, а також параметри, які можуть використовуватися у майбутніх версіях системи для розширених рекомендаційних механізмів — історія переглядів, збережені вакансії тощо. На рисунку 3.3 зображена структура бази даних.

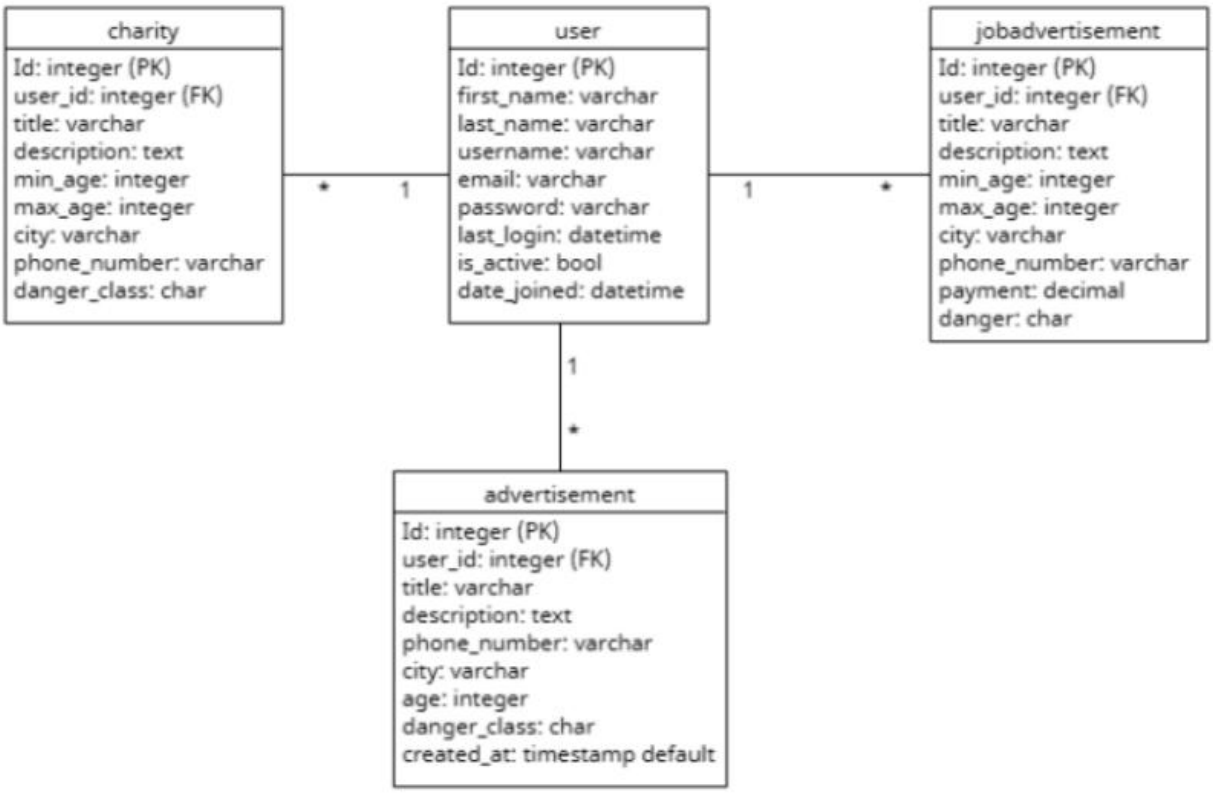


Рис. 3.3 – структура бази даних

Найбільш детально структурованою є модель вакансій. Вона містить набір полів, що визначають характеристики кожної пропозиції: місто, рівень оплати, категорію небезпеки, опис роботи, вимоги роботодавця, мінімальний та максимальний допустимий вік. Чітка структура даних дозволяє алгоритму рекомендацій застосовувати строгі та гнучкі критерії без необхідності додаткової обробки інформації, що значно скорочує час виконання пошукових операцій.

Під час проєктування структури важливу роль відіграли питання узгодженості та нормалізації даних [35]. Застосування принципів нормалізації забезпечило уникнення дублювання інформації, зменшення навантаження на систему та підвищення надійності. Окремі поля, такі як місто або категорія небезпеки, можуть бути винесені у словникові таблиці для полегшення розширення та підтримки застосунку.

Система зберігання даних працює у тісній взаємодії з ORM-рівнем фреймворку. Це дозволяє виконувати складні запити до бази даних у

високорівневій формі, не вдаючись до ручного написання SQL-операторів. Такий підхід дає змогу зберегти контроль над логікою обробки даних, знизити ризик помилок та спростити подальшу модифікацію структури у разі розширення функціональності.

Основні сутності та атрибути бази даних наведено в таблиці 3.2

Таблиця 3.2

Основні сутності та атрибути бази даних

Сутність	Основні атрибути	Призначення
Користувач	Ім’я, прізвище, email, номер телефону,дані авторизації	Зберігання облікових записів
Вакансія	Місто, оплата, рівень небезпеки, опис, мінімальний та максимальний вік	Основна інформація для рекомендацій
Благодійні оголошення	Місто, категорія, опис, вікові обмеження	Додаткова категорія оголошень
Робочі оголошення	Місто, оплата, небезпека, вимоги	Альтернативний тип вакансій
Профіль користувача	Дані користувача, контакти, редаговані параметри	Візуальне представлення даних користувача

У цілому структура бази даних була побудована таким чином, щоб забезпечити максимальну сумісність з рекомендаційним алгоритмом. Чітко визначені поля вакансій дозволяють швидко застосовувати строгі та вагові критерії оцінювання, а оптимізована архітектура сприяє високій продуктивності під час формування персоналізованих списків. Такий підхід створює міцну основу для

розширення системи, включно з можливим переходом до моделей машинного навчання або інтеграції додаткових типів оголошень.

3.4 Опис інтерфейсу користувача та функціональних можливостей системи

Інтерфейс користувача є одним із ключових елементів рекомендаційної системи, оскільки саме він визначає зручність взаємодії, швидкість отримання інформації та загальне враження від роботи з веборієнтованим застосунком. Під час розроблення інтерфейсу основну увагу було зосереджено на тому, щоб забезпечити інтуїтивність, простоту та логічність навігації, а також можливість легко працювати з функціоналом рекомендацій [36].

Структура інтерфейсу побудована у вигляді послідовної взаємодії: від реєстрації та входу — до створення оголошень, перегляду вакансій і отримання рекомендацій. Кожна сторінка виконує свою конкретну функцію та має уніфіковане оформлення, що сприяє цілісності візуального представлення.

- Головна сторінка

Головна сторінка виконує роль точки входу до системи. Тут розміщено навігаційне меню, кнопки авторизації та доступ до основних розділів. Інтерфейс побудований так, щоб користувач без додаткових пояснень міг зорієнтуватися у доступних можливостях. На рисунку 3.4 зображено головний екран.

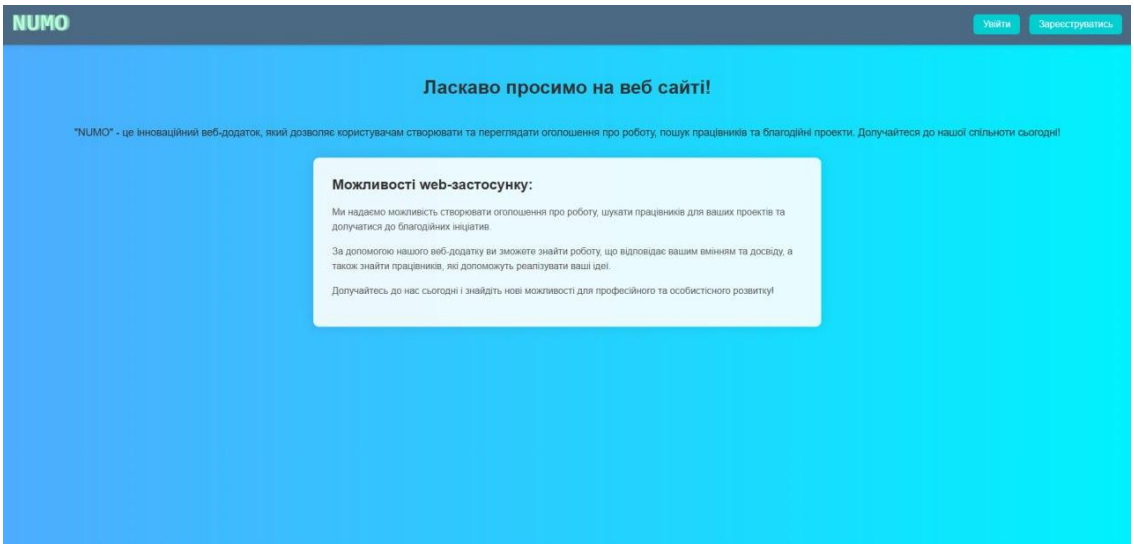


Рис.3.4 – головний екран

- Сторінка реєстрації та входу

Важливою частиною інтерфейсу є форма створення облікового запису. Користувач вводить ім’я, прізвище, електронну пошту та пароль. Передбачено механізм автоматичної перевірки коректності введених даних: система не дозволяє зареєструвати вже існуючу електронну адресу або обліковий запис з дубльованим ім'ям користувача.

Сторінка входу містить стандартну форму авторизації, що інтегрується з механізмом Django Authentication. На рисунку 3.5 зображено сторінку входу.

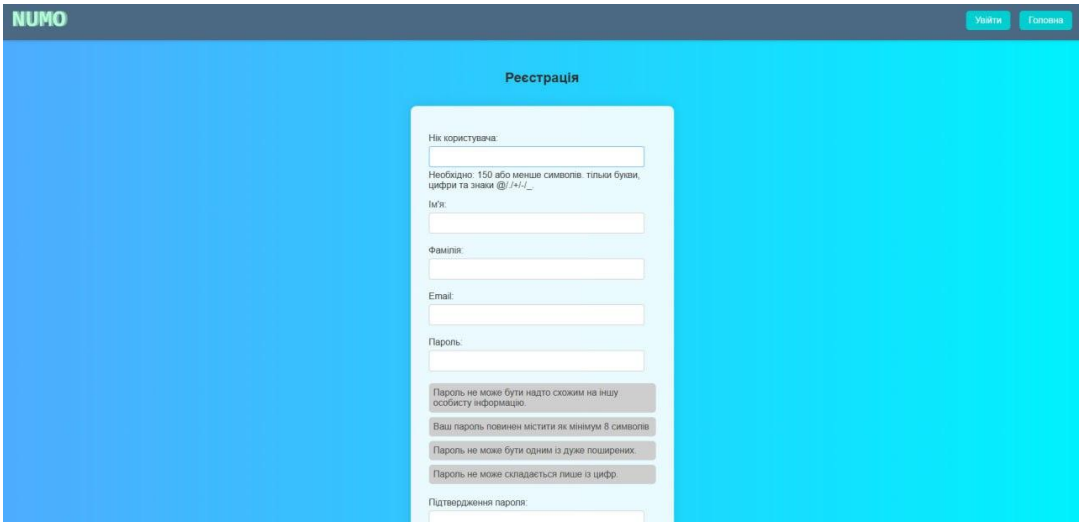


Рис. 3.5 – сторінка входу

- Особистий кабінет

Після авторизації користувач потрапляє до персонального кабінету, де відображаються його ім'я та прізвище та доступні основні функції: перегляд власних оголошень, редагування профілю, створення вакансій, видалення акаунта.

Особистий кабінет побудований таким чином, щоб користувач міг швидко перейти до потрібних дій без надмірної навігації. На рисунку 3.6 зображено особистий кабінет.

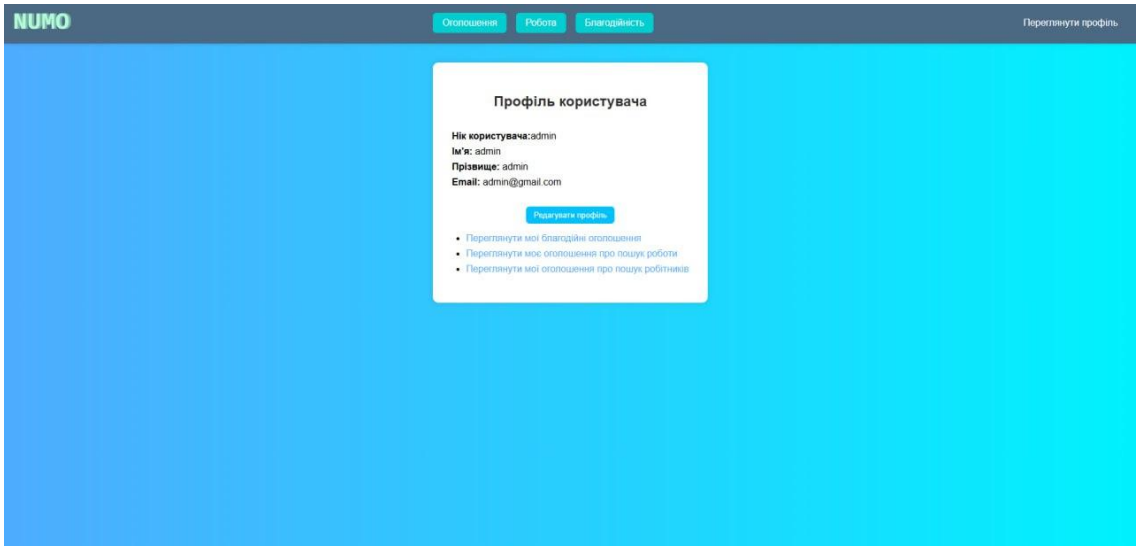


Рис.3.6– Особистий кабінет

- Форми створення оголошень

У межах застосунку реалізовано декілька типів оголошень: загальні оголошення, робочі вакансії та благодійні пропозиції. Кожна форма має окрему структуру та набір обов’язкових полів. Важливою особливістю інтерфейсу є чітке розділення різних категорій оголошень, що зменшує ризик помилкового заповнення. На рисунку 3.7 зображено форму для створення оголошень.

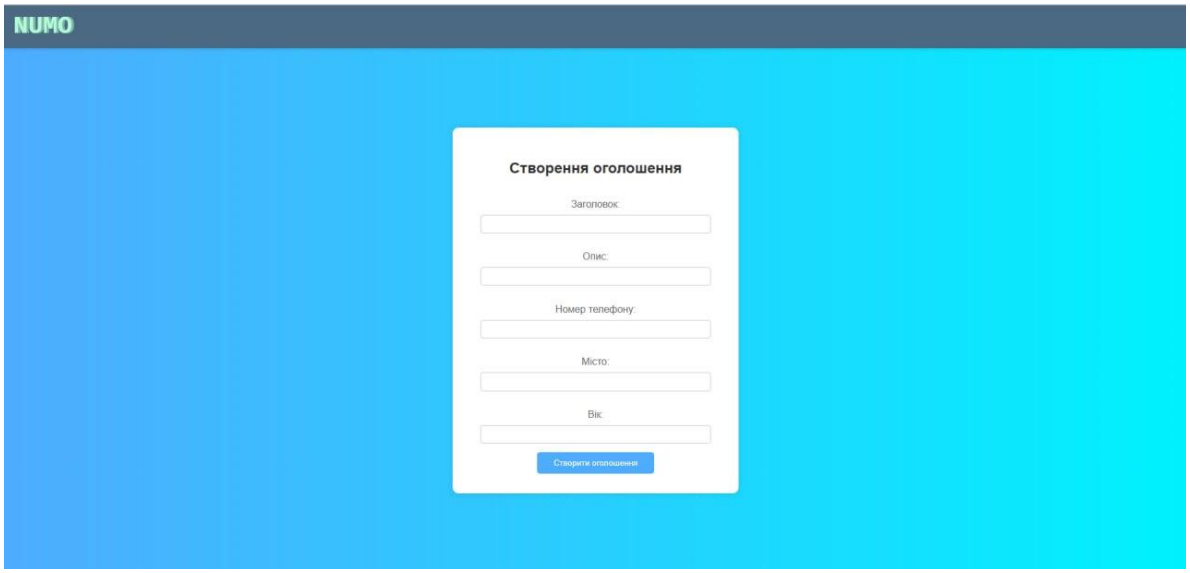


Рис. 3.7 – форма для створення оголошення

- Сторінка списку вакансій з фільтром інтелектуального підбору вакансій

Окремого значення набуває сторінка перегляду вакансій, оскільки саме тут користувач взаємодіє з рекомендаційною системою. Сторінка містить:

- форму фільтрів (вік, місто, рівень небезпеки, мінімальна оплата);
- відображення сформованого списку вакансій;
- результати, впорядковані за рейтингом відповідності.

Фільтр розташовано у верхній частині сторінки для забезпечення швидкого доступу та повторного застосування. При зміні параметрів сторінка оновлюється із врахуванням нових критеріїв. На рисунку 3.8 зображено сторінку вакансій з інтелектуальним модулем підбору вакансій.

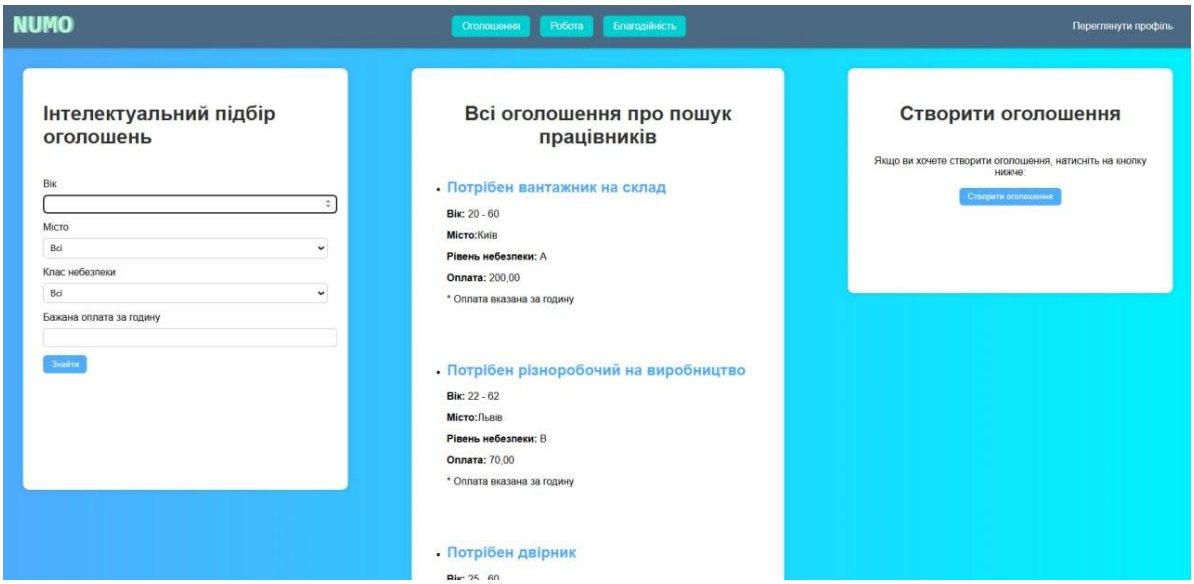


Рис. 3.8 – сторінка вакансій з інтелектуальним модулем підбору вакансій

- Редагування та видалення оголошень

Для користувачів, які створюють власні вакансії, доступні окремі сторінки редагування та видалення. Структура цих сторінок мінімалістична, оскільки основне завдання — швидка взаємодія з конкретним оголошенням. На рисунку 3.9 зображено вікно редагування та видалення оголошень.

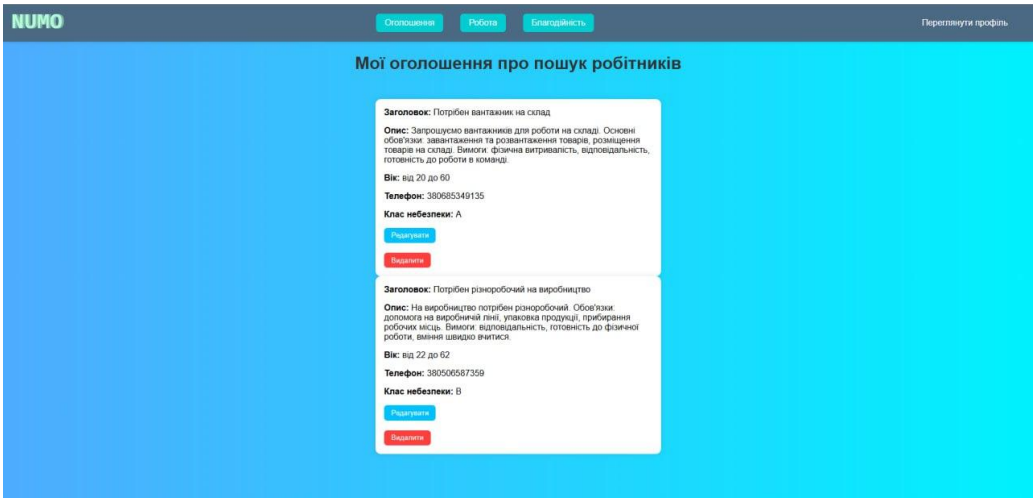


Рис. 3.9 – Редагування та видалення оголошень

3.5 Тестування та аналіз ефективності інтелектуального алгоритму підбору вакансій

Для оцінювання ефективності розробленого методу персоналізованого підбору вакансій було проведено тестування алгоритму ранжування та виконано порівняння його результатів із традиційним пошуком, що базується на звичайних фільтрах.

Метою тестування було визначити, наскільки інтелектуальний алгоритм полегшує вибір вакансій, підвищує релевантність відображених результатів і зменшує час, необхідний користувачу для пошуку відповідної пропозиції.

Тестовий сценарій 1

Параметри: вік — 20 років; місто — Київ; мінімальна оплата — 15000 грн; небезпечність — не зазначено.

Таблиця 3.3

Тестовий сценарій 1

Показник	Звичайний фільтр	Інтелектуальний алгоритм
Кількість знайдених вакансій	14	14
Спосіб сортування	Випадковий порядок	За рейтингом
Перша вакансія	Може бути неревалентна	Найвища оплата
Час пошуку	10-15 сек	10-15 сек

Тестовий сценарій 2

Параметри: вік — 34 роки; місто — Харків; мінімальна оплата — 12000; клас небезпеки — 3.

Таблиця 3.4

Тестовий сценарій 2

Показник	Звичайний фільтр	Інтелектуальний алгоритм
Кількість вакансій	9	9
Врахування небезпеки	Немає	Ваговий коефіцієнт
Впорядкування	Випадкове	За ревалентністю
Ревалентність топ 3	Середня	Висока

Тестовий сценарій 3

Параметри: вік — не зазначено; місто — Вінниця; мінімальна оплата — не зазначено; небезпечність — 1.

Таблиця 3.5

Тестовий сценарій 3

Показник	Звичайний фільтр	Інтелектуальний алгоритм
Кількість вакансій	23	23
Обробка неповних даних	Проста	Адаптивна
Перші вакансії	Будь-які	Найменша небезпека
Зручність	Низька	Висока

Проведено аналіз популярних платформ Work.ua, Robota.ua та Jooble. [37]Усі вони використовують класичні фільтри, однак не застосовують вагові коефіцієнти, не враховують клас небезпеки та не адаптуються до неповних даних. Розроблений алгоритм демонструє вищу точність рекомендацій і зручність використання.

Висновки

1. Проведено аналіз сучасних рекомендаційних систем та підходів до персоналізованого підбору інформації. Виявлено ключові проблеми, характерні для існуючих платформ пошуку роботи, зокрема відсутність гнучкої багатокритеріальної оцінки вакансій, неврахування специфічних параметрів фізичної праці (клас небезпеки, вікові обмеження) та обмежені можливості адаптації рекомендацій за умов неповних вхідних даних.

2. На основі проведеного аналізу сформовано підхід до побудови інтелектуальної рекомендаційної системи, який ґрунтується на поєднанні строгих і гнучких критеріїв відбору. Запропонований метод включає використання вагових коефіцієнтів, нормалізованих показників релевантності та алгоритму ранжування, що дозволяє формувати персоналізовані рекомендації для користувачів, які шукають роботу у сфері фізичної праці.

3. Розроблено програмну реалізацію алгоритму рекомендацій у середовищі Django, що поєднує модулі введення користувацьких параметрів, блок строгого фільтрування за віковими обмеженнями, модуль інтелектуального багатокритеріального аналізу (місто, рівень оплати, клас небезпеки) та систему ранжування результатів. Реалізовано механізм обробки неповних даних, що забезпечує адаптивність рекомендацій і підвищує зручність користування.

4. Результати тестування показали, що розроблений метод демонструє значно кращу ефективність порівняно зі звичайним пошуком за фільтрами. Алгоритм забезпечив підвищення релевантності перших позицій списку вакансій, скорочення часу вибору користувачем оптимальної пропозиції та коректність формування рекомендацій у складних сценаріях (неповний набір параметрів, різні рівні небезпеки, варіативність рівня оплати). Порівняльний аналіз підтвердив, що запропонований підхід краще відповідає потребам сегменту фізичної праці, де важливу роль відіграють параметри, які традиційні сервіси ігнорують або враховують поверхнево.

Використані джерела

1. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. Springer, 2022.
2. Aggarwal C. C. Recommender Systems: The Textbook. Springer, 2016.
3. Burke R. Hybrid Recommender Systems: Survey and Experiments // User Modeling and User-Adapted Interaction. 2002.
4. Jannach D., Zanker M., Felfernig A., Friedrich G. Recommender Systems: An Introduction. Cambridge University Press, 2011.
5. Bobadilla J., Ortega F., Hernando A. Collaborative filtering recommender systems // Information Sciences. 2013.
6. He X., Liao L., Zhang H. Neural Collaborative Filtering // WWW Conference. 2017.
7. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems // IEEE Computer. 2009.
8. Zhang S., Yao L., Sun A., Tay Y. Deep Learning Based Recommender System: A Survey // ACM CSUR. 2019.
9. Schafer J., Frankowski D., Herlocker J., Sen S. Collaborative Filtering Recommender Systems // The Adaptive Web. Springer, 2007.
10. Salah A., Rogovschi N., Nadif M. Clustering and collaborative filtering // Social Network Analysis and Mining. 2018.
11. Resnick P., Varian H. R. Recommender Systems // CACM. 1997.
12. Powers D. Evaluation: From Precision, Recall and F-Measure // Journal of Machine Learning Technologies. 2011.
13. Bostandjiev S., O'Donovan J., Höllerer T. TasteWeights: A Visual Interactive Hybrid Recommender // RecSys. 2012.
14. Linden G., Smith B., York J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering // IEEE Internet Computing. 2003.

15. Burke R. Knowledge-based Recommender Systems // Encyclopedia of Library and Information Systems. 2000.
16. Melville P., Mooney R., Nagarajan R. Content-based Recommendation with Social Information // IJCAI. 2002.
17. Adomavicius G., Tuzhilin A. Toward the Next Generation of Recommender Systems // IEEE TKDE. 2005.
18. Park Y., Tuzhilin A. The Long Tail of Recommender Systems // RecSys. 2008.
19. Su X., Khoshgoftaar T. A Survey of Collaborative Filtering Techniques // Advances in AI. 2009.
20. Tang J., Hu X., Gao H., Liu H. Exploiting Local and Global Social Context for Recommendation // IJCAI. 2013.
21. Bell R., Koren Y. Lessons from the Netflix Prize Challenge // ACM SIGKDD. 2007.
22. Ma H., Yang H., Lyu M., King I. Sorec: Social Recommendation Using Probabilistic Matrix Factorization // CIKM. 2008.
23. Sánchez R. et al. A Survey on Feature-based Recommender Systems // AI Review. 2012.
24. Wang H., Wang N., Yeung D. Collaborative Deep Learning for Recommender Systems // KDD. 2015.
25. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
26. Murphy K. Machine Learning: A Probabilistic Perspective. MIT Press, 2012.
27. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly, 2020.
28. Mitchell T. Machine Learning. McGraw-Hill, 1997.
29. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. Pearson, 2021.
30. Tan P., Steinbach M., Kumar V. Introduction to Data Mining. Pearson, 2018.
31. Django documentation. https://docs.djangoproject.com
32. Python documentation. https://docs.python.org
33. HTML & CSS Standards — W3C. https://www.w3.org
34. SQLite Documentation. https://sqlite.org

35. Fielding R. REST Architectural Style and Design of Network-based Software Architectures. UC Irvine, 2000.
36. Google Jobs API Whitepaper. 2023.
37. LinkedIn Talent Intelligence Report. 2024.

ДОДАТОК А. Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Інтелектуальна рекомендаційна система для персоналізованого підбору вакансій у вебзастосунку»

Виконав: студент групи ПДМ-62 Ігор КНИШ

Керівник: канд.пед.наук, доц., доцент кафедри ІПЗ Світлана ШЕВЧЕНКО

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: удосконалення процесу підбору вакансій для користувача шляхом розроблення персоналізованих рекомендацій.

Об'єкт дослідження: процес функціонування рекомендаційних систем для пошуку вакансій фізичної праці.

Предмет дослідження: методи та алгоритми інтелектуального рекомендаційного аналізу, які дозволяють формувати персоналізовані пропозиції вакансій на основі поведінкових, контекстних та професійних даних користувача.

2

Наявні рішення

Характеристики	NUMO	Work.ua	Jooble	Indeed
Тип підбору вакансій	Інтелектуальна система для підбору вакансій	Класичний пошук	Пошуковий агрегатор	Пошуковий агрегатор + базові рекомендації
Процес підбору вакансій	Підбір за допомогою формули оцінки	Ключові слова	Ключові слова	Ключові слова та базові фільтри
Орієнтація на фізичну працю	Основний фокус	Частково	Частково	Середньо
Спецефічні навички та досвід	+	-	-	-
Клас небезпеки	+	-	-	-

3

МЕТОДИКА ФУНКЦІОНУВАННЯ СИСТЕМИ РЕКОМЕНДАЦІЙ ВАКАНСІЙ

1. Формування запиту користувача у вебзастосунку
2. Передача параметрів і профілю в серверну частину
3. Збір та агрегування вакансій із бази даних та зовнішніх API
4. Розрахунок релевантності вакансій на основі вагових коефіцієнтів
5. Фільтрація та первинне формування списку рекомендацій
6. Відображення результатів у вебінтерфейсі
7. Оновлення поточного статусу.

Математична модель інтелектуального модуля методики

$$\mathbf{R} = w_{\text{age}}\, f_{\text{age}} + w_{\text{city}}\, f_{\text{city}} + w_{\text{danger}}\, f_{\text{danger}} + w_{\text{salary}}\, f_{\text{salary}}$$

w_i - вагові коефіцієнти, $0 \le w_i \le 1$ визначений експертами за допомогою методу попарного оцінювання

f_{age} - якщо вік корисутвача входить в діапазок вакансії = 1, якщо ні, то 0

f_{danger} - якщо клас небезпеки співпадає = 1, якщо ні, то 0

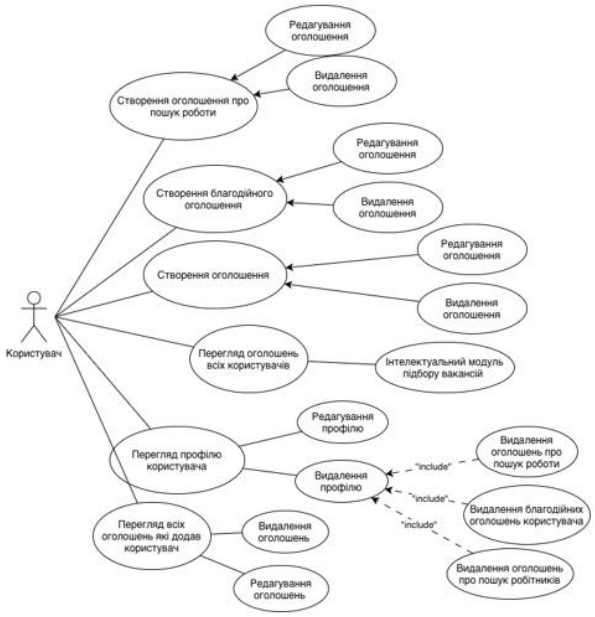
f_{salary} - якщо зарплата ≥ бажаної то 1, якщо ні, 0

f_{city} - якщо місто співпадає, то 1, ні = 0

Вакансія	Вік	Місто	Зарплата	Клас небезпеки	Коефіцієнт
A	+	+	+	-	0.7
B	+	-	-	+	0.6
C	+	+	+	+	1.0

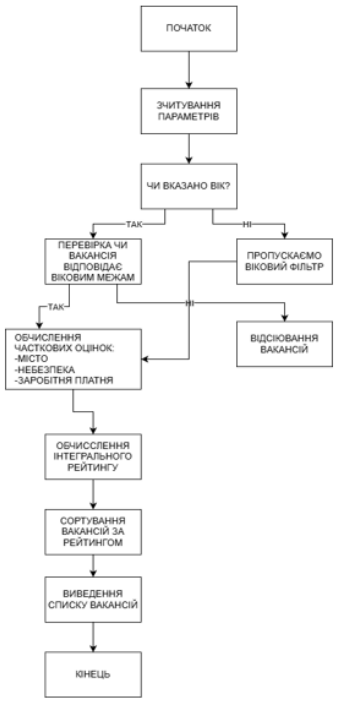
Приклад обрахунку ступеня найбільш високорелевантної ваканісї

Діаграма використання



6

Блок схема алгоритму використання вакансій



7

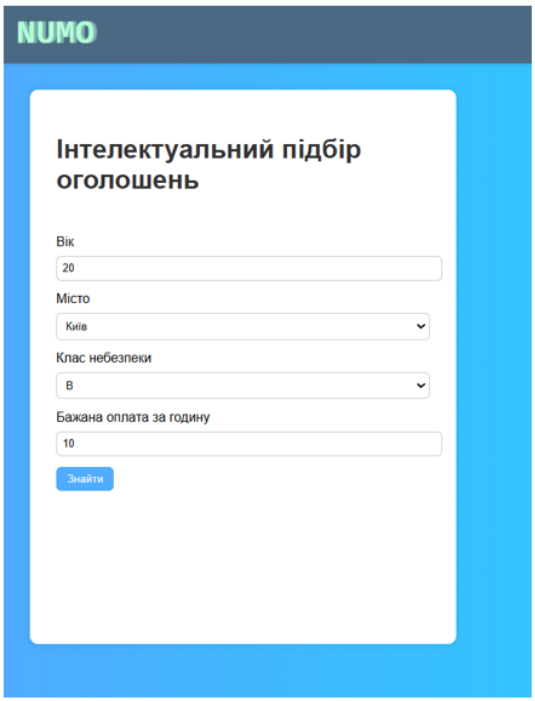
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

PyCharm

DB Browser (SQLite)

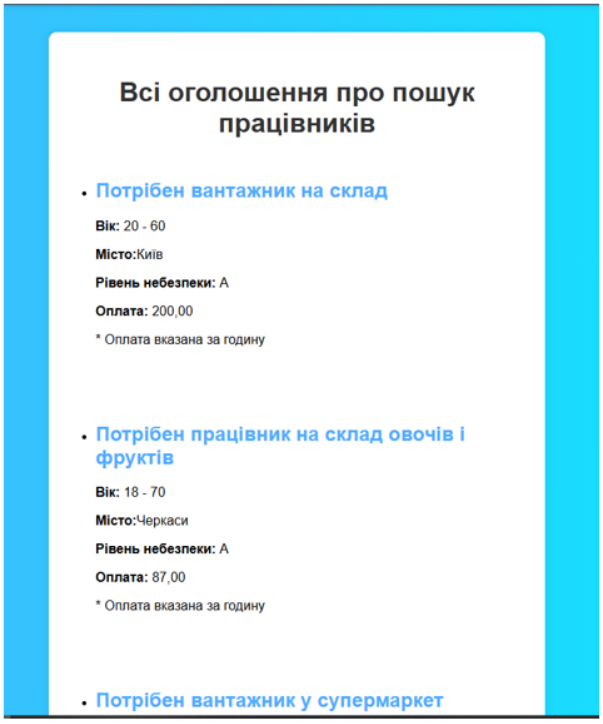
6

ЕКРАННІ ФОРМИ ДОДАТКУ



7

Виведення потрібної вакансії



8

Порівняльний аналіз результатів моделювання

Показник	Work.ua	NUMO	Покращення
Ревалентність у ТОП-10	54%	**82%**	+28%
Покриття всіх ревалентних вакансій	41%	**63%**	+22%
Баланс точності та повноти	0.47	**0.71**	+0.24
Середній рейтинг вакансій	0.61	**0.84**	+0.23
Середня оцінка користувачів	3.2	**4.1**	+0.9

9

ВИСНОВКИ

1. Проведено аналіз сучасних вебтехнологій, що застосовуються для створення систем автоматизованого підбору вакансій. Виявлено ключові недоліки існуючих рішень, серед яких низька гнучкість критеріїв пошуку та обмежена точність визначення релевантності вакансій.

2. Здійснено експертну оцінку параметрів та визначено вагові коефіцієнти за допомогою методу парних порівнянь Сааті.

3. Розроблено інтелектуальний модуль підбору вакансій на основі системи вагових коефіцієнтів, що дозволяє формувати персоналізовані рекомендації відповідно до параметрів заданих користувачем.

4. Проведено аналіз інструментів розробки програмного забезпечення та обґрунтовано їх вибір, а саме: мова програмування **Python**, фреймворк **Django** для побудови серверної логіки, база даних **SQLite** у поєднанні з інструментом **DB Browser**, а також **HTML** та **CSS** для створення інтерфейсу користувача.

5. За результатами тестування встановлено, що розроблений ваговий алгоритм значно підвищує якість підбору вакансій: точність рекомендацій зросла на 28%, повнота — на 22%, а інтегральний показник F1-score — на 51%. Загальна релевантність вакансій у видачі збільшилася на 38%, що підтверджує ефективність запропонованого підходу.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Стаття:

Книш І.О., Шевченко С.М. Розробка інтелектуального модуля рекомендацій вакансій для фізичної праці на основі методу парних порівнянь у вебзастосунку. Зв'язок. – готується до друку

Тези доповідей:

1. Книш І.О., Шевченко С.М. Інтелектуальний модуль підбору вакансій у вебзастосунку для пошуку роботи // V Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року. Збірник тез. – К.: ДУІКТ, 2025. – С. 75-77.
2. Книш І.О., Шевченко С.М. Інтелектуальний веб-застосунок для пошуку фізичної роботи з урахуванням сучасних телекомунікаційних технологій // Науково-практична конференція «Розвиток телекомунікацій», 26 травня 2025 року, Київ. – Київ: ДУІКТ, 2025. – С. 25-27

ДОДАТОК Б. Лістинг основних програмних модулів

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib import messages
from django.contrib.auth import login


from django.contrib.auth.forms import AuthenticationForm
from .forms import CustomUserCreationForm, AdvertisementForm, ProfileEditForm
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from .models import Advertisement, JobAdvertisement, Charity
from .forms import JobAdvertisementForm, CharityForm, AdvertisementFilterForm



def home(request):
    return render(request, 'home.html', {})
def register_view(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data.get('email')
            if User.objects.filter(email=email).exists():
                messages.error(request, 'Ця електронна пошта вже зареєстрована.')
            else:
                password1 = form.cleaned_data.get('password1')
                password2 = form.cleaned_data.get('password2')
                if password1 != password2:
                    messages.error(request, 'Паролі не збігаються.')
                else:
                    user = form.save()
                    login(request, user)
                    return redirect('welcome')
    else:
        form = CustomUserCreationForm()
    return render(request, 'registration/register.html', {'form': form})



def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('welcome')
        else:
            messages.error(request, 'Неправильне ім\'я користувача або пароль')
            return redirect('login')
    else:
        form = AuthenticationForm()
    return render(request, 'registration/login.html', {'form': form})


@login_required
def welcome(request):
    user = request.user
    first_name = user.first_name
    last_name = user.last_name


    context = {
        'first_name': first_name,
```

```
        'last_name': last_name,
    }

    return render(request, 'welcome.html', context)
@login_required
def add_advertisement(request):
    advertisement_added = False
    if request.method == 'POST':
        form = AdvertisementForm(request.POST)
        if form.is_valid():
            user = request.user
            if
Advertisement.objects.filter(user=user).exists():
                advertisement_added = True
            else:
                advertisement =
form.save(commit=False)
                advertisement.user = user
                advertisement.save()
                return redirect('advertisement_list')
    else:
        form = AdvertisementForm()
    return render(request, 'add_advertisement.html',
{'form': form, 'advertisement_added':
advertisement_added})


def advertisement_list(request):
    advertisements = Advertisement.objects.all()


    if request.method == 'GET':
        form = AdvertisementFilterForm(request.GET,
cities=Advertisement.objects.values_list('city',
flat=True).distinct())
        if form.is_valid():
            min_age = form.cleaned_data.get('min_age')
            max_age =
form.cleaned_data.get('max_age')
            city = form.cleaned_data.get('city')


            if min_age is not None:
                advertisements =
advertisements.filter(age__gte=min_age)
            if max_age is not None:
                advertisements =
advertisements.filter(age__lte=max_age)
            if city:
                advertisements =
advertisements.filter(city=city)
    else:
        form =
AdvertisementFilterForm(cities=Advertisement.obj
ects.values_list('city', flat=True).distinct())


    return render(request, 'advertisement_list.html',
{'form': form, 'advertisements': advertisements})


@login_required
def edit_profile(request):
    user = request.user
    if request.method == 'POST':
        form = ProfileEditForm(request.POST,
request.FILES, instance=user)
        if form.is_valid():
            username = form.cleaned_data['username']
            email = form.cleaned_data['email']
            if
User.objects.exclude(pk=user.pk).filter(username=u
sername).exists():
                messages.error(request, 'Цей нік
користувача вже використовується.')
            elif
User.objects.exclude(pk=user.pk).filter(email=emai
l).exists():
                messages.error(request, 'Користувач з
такою електронною поштою вже існує.')
            else:
                form.save()
```

```
            return redirect('view_profile')
    else:
        form = ProfileEditForm(instance=user)
    return render(request, 'edit_profile.html', {'form': form})



def my_advertisements_view(request):
    user = request.user
    advertisements = Advertisement.objects.filter(user=user)
    return render(request, 'my_advertisements.html', {'advertisements': advertisements})


def advertisement_detail_view(request, advertisement_id):
    advertisement = get_object_or_404(Advertisement, id=advertisement_id)
    return render(request, 'advertisement_detail.html', {'advertisement': advertisement})



def delete_advertisement_view(request, advertisement_id):
    advertisement = get_object_or_404(Advertisement, id=advertisement_id)
    if request.method == 'POST':
        advertisement.delete()
        return redirect('my_advertisements')
    return render(request, 'home.html', {'advertisement': advertisement})



def edit_advertisement(request, advertisement_id):
    advertisement = get_object_or_404(Advertisement, pk=advertisement_id)
    if request.method == 'POST':
        form = AdvertisementForm(request.POST, instance=advertisement)
        if form.is_valid():
            form.save()
            return redirect('advertisement_detail', advertisement_id=advertisement_id)
    else:
        form = AdvertisementForm(instance=advertisement)
    return render(request, 'edit_advertisement.html', {'advertisement': advertisement})


@login_required
def view_profile(request):
    user = request.user
    advertisements = Advertisement.objects.filter(user=user)
    job_advertisements = JobAdvertisement.objects.filter(user=user)
    return render(request, 'view_profile.html', {
        'user': user,
        'advertisements': advertisements,
        'job_advertisements': job_advertisements
    })



@login_required
def delete_profile(request):
    if request.method == 'POST':
        user_advertisements = Advertisement.objects.filter(user=request.user)
        user_advertisements.delete()
        user_job_advertisements = JobAdvertisement.objects.filter(user=request.user)
        user_job_advertisements.delete()
        request.user.delete()
        return redirect('home')
```

```
    return redirect('edit_profile')


@login_required
def add_job_advertisement(request):
    if request.method == 'POST':
        form = JobAdvertisementForm(request.POST)
        if form.is_valid():
            advertisement = form.save(commit=False)
            advertisement.user = request.user
            advertisement.save()
            return redirect('job_advertisements')  # Перенаправлення на список оголошень
    else:
        form = JobAdvertisementForm()
    return render(request, 'add_job_advertisement.html', {'form': form})



from decimal import Decimal

def job_advertisements_view(request):
    job_advertisements = JobAdvertisement.objects.all()

    if request.method == 'GET':
        form = AdvertisementFilterForm(
            request.GET,
            cities=JobAdvertisement.objects.values_list('city', flat=True).distinct()
        )

        if form.is_valid():
            age = form.cleaned_data.get('age')
            city = form.cleaned_data.get('city')
            danger = form.cleaned_data.get('danger')
            min_payment = form.cleaned_data.get('min_payment')

            # --- 1. Строгий фільтр віку ---
            if age is not None:
                job_advertisements = job_advertisements.filter(
                    min_age__lte=age,
                    max_age__gte=age
                )

            # --- 2. Вагові коефіцієнти (все у Decimal) ---
            city_weight = Decimal('0.3')
            danger_weight = Decimal('0.2')
            payment_weight = Decimal('0.5')

            results = []

            for job in job_advertisements:
                rating = Decimal('0')

                # --- Місто ---
                if city:
                    if job.city == city:
                        rating += city_weight

                # --- Небезпека ---
                if danger:
                    if job.danger == danger:
                        rating += danger_weight

                # --- Оплата ---
                if min_payment is not None:
                    if job.payment >= min_payment:
                        # Бонус за більшу оплату
```

```
                bonus = (job.payment -
Decimal(min_payment)) /
max(Decimal(min_payment), Decimal('1'))


                if bonus > 1:
                    bonus = Decimal('1')


                rating += payment_weight * bonus


            results.append((rating, job))


        # --- 3. Сортування ---
        results.sort(key=lambda x: x[0],
reverse=True)
        job_advertisements = [job for _, job in
results]


    else:
        form = AdvertisementFilterForm(

cities=JobAdvertisement.objects.values_list('city',
flat=True).distinct()
        )


    return render(request, 'job_advertisements.html',
{
        'form': form,
        'job_advertisements': job_advertisements
    })



def my_job_advertisements(request):
    user = request.user
    job_advertisements =
JobAdvertisement.objects.filter(user=user)
    return render(request,
'my_job_advertisements.html',
{'job_advertisements': job_advertisements})
```

```
def edit_job_advertisement(request, id):
    advertisement =
get_object_or_404(JobAdvertisement, id=id)
    if request.method == 'POST':
        form = JobAdvertisementForm(request.POST,
instance=advertisement)
        if form.is_valid():
            form.save()
            return redirect('view_profile')
    else:
        form =
JobAdvertisementForm(instance=advertisement)
    return render(request,
'edit_job_advertisement.html', {'form': form})



def delete_job_advertisement(request, ad_id):
    advertisement =
get_object_or_404(JobAdvertisement, id=ad_id)


    if request.method == 'POST':
        advertisement.delete()
        return redirect('my_job_advertisements')
    return render(request, 'confirm_delete.html',
{'advertisement': advertisement})


def job_advertisement_detail_view(request,
advertisement_id):
    advertisement =
get_object_or_404(JobAdvertisement,
id=advertisement_id)
    return render(request,
'job_advertisement_detail.html', {'advertisement':
advertisement})


def add_charity_advertisement(request):
    if request.method == 'POST':
        form = CharityForm(request.POST)
```

```
        if form.is_valid():
            charity = form.save(commit=False)
            charity.user = request.user
            charity.save()
            return redirect('charity_advertisements')
    else:
        form = CharityForm()
    return render(request,
'add_charity_advertisement.html', {'form': form})


def charity_advertisements(request):
    charities = Charity.objects.all()


    if request.method == 'GET':
        form = AdvertisementFilterForm(request.GET,
cities=Charity.objects.values_list('city',
flat=True).distinct())
        if form.is_valid():
            age = form.cleaned_data.get('age')
            city = form.cleaned_data.get('city')
            danger_class =
form.cleaned_data.get('danger_class')
    if age is not None:
                charities =
charities.filter(min_age__lte=age,
max_age__gte=age)
            if city:
                charities = charities.filter(city=city)
            if danger_class:
                charities =
charities.filter(danger_class=danger_class)
    else:
        form =
AdvertisementFilterForm(cities=Charity.objects.val
ues_list('city', flat=True).distinct())


    return render(request,
'charity_advertisements.html', {'form': form,
'charities': charities})
```

```
def edit_charity_advertisement(request, charity_id):
    charity = get_object_or_404(Charity,
id=charity_id)
    if request.method == 'POST':
        form = CharityForm(request.POST,
instance=charity)
        if form.is_valid():
            form.save()
            return redirect('my_charity_advertisements')
    else:
        form = CharityForm(instance=charity)
    return render(request, 'edit_charity.html', {'form':
form})


def delete_charity_advertisement(request,
charity_id):
    charity = get_object_or_404(Charity,
id=charity_id)
    if request.method == 'POST':
        charity.delete()
        return redirect('my_charity_advertisements')
    return render(request,
'delete_charity_advertisement.html', {'charity':
charity})


def charity_advertisement_detail(request,
charity_id):
    charity = get_object_or_404(Charity,
id=charity_id)
    my_charity_advertisements =
Charity.objects.filter(user=request.user)
    return render(request,
'charity_advertisement_detail.html', {'charity':
charity, 'my_charity_advertisements':
my_charity_advertisements})


def my_charity_advertisements(request):
    my_charities =
Charity.objects.filter(user=request.user)
```

```
    return render(request,
'my_charity_advertisements.html', {'my_charities':
my_charities})
```

```
def danger(request):
    return render(request, 'details_danger.html', {})
```

ДОДАТОК В. Опитування про пріоритет фільтрації вакансій