

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук
(повна назва)
Кафедра _____ Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський)

Розробка рекомендаційної системи підбору вакансій на основі аналізу тексту
резюме та опису вакансій
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-3

_____ Дмитро ІВАЩЕНКО
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)

Тип програми _____ освітньо-професійна

Освітня програма _____ Штучний інтелект

(повна назва освітньої програми)

Керівник _____ ст. викл. Ігор БІБІЧКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

(підпис)

_____ Лариса ЧАЛА
(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
Кафедра _____ Штучного інтелекту _____
Рівень вищої освіти _____ перший (бакалаврський) _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)
Тип програми _____ освітньо-професійна _____
Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«_____» _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Іващенко Дмитру Костянтиновичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка рекомендаційної системи підбору вакансій на основі аналізу тексту резюме та опису вакансій

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 7 липня 2026 р.

3. Вихідні дані до роботи текстові резюме у форматах PDF, DOCX, TXT; описи вакансій із вимогами та стеком технологій; програмний репозиторій ml-resume-parser; методичні вказівки до виконання кваліфікаційної роботи; скріншоти роботи вебзастосунку; науково-технічні джерела з інформаційного пошуку, обробки природної мови та рекомендаційних систем

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та постановка задачі

2) Математичне та алгоритмічне забезпечення

3) Проектування програмної системи

4) Реалізація, тестування та аналіз результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі та літературних джерел	09.06.2026–14.06.2026	виконано
3	Формулювання постановки задачі, мети, об'єкта і предмета дослідження	15.06.2026–17.06.2026	виконано
4	Дослідження методів обробки тексту, TF-IDF, cosine similarity і embedding-моделей	18.06.2026–21.06.2026	виконано
5	Проектування архітектури програмної системи та структури бази даних	22.06.2026–25.06.2026	виконано
6	Реалізація backend-модулів авторизації, резюме, вакансій і рекомендацій	26.06.2026–28.06.2026	виконано
7	Реалізація frontend-інтерфейсу завантаження резюме та перегляду рекомендацій	27.06.2026–30.06.2026	виконано
8	Тестування роботи системи та аналіз результатів	01.07.2026–05.07.2026	виконано
9	Оформлення пояснювальної записки та додатків	06.07.2026–09.07.2026	виконано
10	Підготовка презентаційних матеріалів та доповіді	10.07.2026–12.07.2026	виконано
11	Захист перед ЕК	14.07.2026	

Дата видачі завдання 8 червня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. каф. ІІІ Ігор БІБІЧКОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 57 с., 4 рис., 5 табл., 1 дод., 21 джерело.

ВАКАНСІЯ, ВЕКТОРНЕ ПРЕДСТАВЛЕННЯ, КОСИНУСНА ПОДІБНІСТЬ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, РЕКОМЕНДАЦІЙНА СИСТЕМА, РЕЗЮМЕ, СЕМАНТИЧНИЙ АНАЛІЗ, SENTENCE TRANSFORMERS, TF-IDF.

Об'єкт дослідження – процес автоматизованого підбору вакансій для здобувачів роботи на основі текстової інформації про кандидата та вимоги роботодавця.

Предмет дослідження – методи обробки тексту, виділення навичок, обчислення подібності резюме й вакансій та їх ранжування в рекомендаційній системі.

Мета роботи – розробити рекомендаційну систему, яка аналізує резюме та описи вакансій, визначає рівень відповідності кандидата й формує пояснюваний ранжований список рекомендацій.

Методи дослідження – аналіз літературних джерел, обробка природної мови, словникове виділення навичок, TF-IDF, косинусна подібність, sentence embeddings, role-aware scoring, проектування вебзастосунків і функціональне тестування.

У роботі проаналізовано предметну область, обґрунтовано вибір методів зіставлення текстів, спроектовано full-stack архітектуру, реалізовано backend на FastAPI з асинхронним доступом до PostgreSQL, frontend на Next.js і модуль рекомендацій із трьома стратегіями: keyword, TF-IDF та embeddings. Практичним результатом є вебзастосунок, який приймає резюме у форматах PDF, DOCX або TXT, витягує текст, зіставляє його з активними вакансіями, зберігає рекомендації та показує відсоток відповідності, знайдені й відсутні навички та пояснення результату.

ABSTRACT

Bachelor's thesis contains: 76 pp., 4 fig., 5 tabl., 1 ann., 22 references.

COSINE SIMILARITY, JOB VACANCY, MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, RECOMMENDER SYSTEM, RESUME, SEMANTIC ANALYSIS, SENTENCE TRANSFORMERS, TF-IDF, VECTOR REPRESENTATION.

The object of research is the automated job recommendation process based on textual information about a candidate and employer requirements.

The subject of research is a set of text preprocessing, skill extraction, resume-vacancy similarity estimation and job ranking methods used in a recommendation system.

The purpose of the work is to develop and investigate a recommendation system that analyses resume text and job descriptions, estimates the degree of candidate-job relevance and generates an explainable ranked list of job recommendations.

The research methods include literature analysis, natural language processing, dictionary-based skill extraction, TF-IDF, cosine similarity, sentence embeddings, role-aware scoring, web application design and functional testing.

The thesis analyses the problem domain, justifies the selected matching methods, designs a full-stack architecture and implements a web application with a FastAPI backend, PostgreSQL database, Next.js frontend and a recommendation module supporting three strategies: keyword, TF-IDF and embeddings. The resulting system accepts resumes in PDF, DOCX or TXT format, extracts text, compares it with active vacancies, stores recommendations and presents the matching percentage, matched skills, missing skills and textual explanation to the user.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	8
Вступ.....	9
1 Аналіз предметної галузі та постановка задачі	11
1.1 Актуальність автоматизованого підбору вакансій.....	11
1.2 Особливості текстових даних резюме та вакансій.....	12
1.3 Огляд рекомендаційних систем і методів NLP.....	21
1.4 Аналіз аналогів та вимог до системи.....	22
1.5 Постановка задачі дослідження	27
2 Математичне та алгоритмічне забезпечення	28
2.1 Попередня обробка текстових документів.....	28
2.2 Виділення навичок на основі словника	28
2.3 Метод TF-IDF та косинусна подібність	30
2.4 Семантичні векторні представлення речень	31
2.5 Рольове коригування оцінки відповідності	32
2.6 Формування пояснюваних рекомендацій	33
3 Проєктування програмної системи.....	35
3.1 Вимоги до програмного забезпечення.....	35
3.2 Архітектура вебзастосунку.....	36
3.3 Структура бази даних.....	37
3.4 Проєктування backend-частини.....	38
3.5 Проєктування frontend-частини	39
3.6 Безпека та розгортання.....	40
4 Реалізація, тестування та аналіз результатів.....	46
4.1 Середовище реалізації.....	46
4.2 Реалізація завантаження та аналізу резюме	46
4.3 Реалізація генерації рекомендацій	47
4.4 Інтерфейс користувача.....	48
4.5 Функціональне тестування системи	50

4.6 Аналіз отриманих результатів.....	51
Висновки.....	52
Перелік джерел посилання	54
Додаток А Відомість кваліфікаційної роботи.....	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

API – Application Programming Interface – прикладний програмний інтерфейс;

CRUD – Create, Read, Update, Delete – базові операції над даними;

CV – Curriculum Vitae – резюме кандидата;

JWT – JSON Web Token – формат токена для автентифікації;

ML – Machine Learning – машинне навчання;

NLP – Natural Language Processing – обробка природної мови;

ORM – Object-Relational Mapping – об’єктно-реляційне відображення;

REST – Representational State Transfer – архітектурний стиль веб API;

SBERT – Sentence-BERT – модель отримання семантичних векторів речень;

TF-IDF – Term Frequency – Inverse Document Frequency – статистичне зважування термінів;

UI – User Interface – інтерфейс користувача;

UX – User Experience – користувацький досвід.

ВСТУП

Сучасний ринок праці характеризується швидкою зміною вимог до кандидатів, збільшенням кількості онлайн-вакансій і зростанням конкуренції між фахівцями. Кандидат часто має переглядати десятки або сотні оголошень, порівнюючи свій досвід із вимогами роботодавців. Такий процес є трудомістким, суб'єктивним і не завжди приводить до оптимального вибору, оскільки одна й та сама компетенція може бути описана різними словами, а релевантність вакансії не обмежується простим збігом назв технологій.

Особливо актуальною ця проблема є для ІТ-сфери, де вакансії містять велику кількість технічних термінів, назв фреймворків, мов програмування, баз даних, хмарних сервісів та вимог до досвіду. Резюме кандидата також має слабо структурований характер: навички можуть бути розташовані у різних секціях, досвід описується природною мовою, а частина важливих ознак подається у вигляді непрямих формулювань. Тому для якісного підбору вакансій доцільно використовувати методи обробки природної мови та машинного навчання.

Рекомендаційна система підбору вакансій має не тільки впорядкувати позиції за рівнем відповідності, а й пояснити користувачеві причини рекомендації: які навички вже збігаються з вимогами, які компетенції відсутні, чи відповідає рівень досвіду заявленому в вакансії, чи є ця вакансія суміжною з основною спеціалізацією кандидата. Наявність такого пояснення підвищує довіру до системи та перетворює її з простого фільтра на інструмент підтримки прийняття рішень.

Метою кваліфікаційної роботи є розроблення рекомендаційної системи, яка виконує аналіз тексту резюме та описів вакансій, оцінює відповідність кандидата вакансіям за кількома ознаками та формує ранжований список вакансій із поясненням результатів.

Для досягнення мети необхідно розв'язати такі задачі: проаналізувати предметну галузь автоматизованого підбору вакансій; розглянути методи представлення тексту та обчислення подібності документів; обґрунтувати вибір алгоритмів keyword matching, TF-IDF і embeddings; спроектувати архітектуру вебзастосунку; реалізувати модулі завантаження резюме, керування вакансіями та генерації рекомендацій; провести функціональне тестування розробленої системи.

Об'єктом дослідження є процес автоматизованого підбору вакансій для користувача за текстом його резюме. Предметом дослідження є методи аналізу текстових документів, виділення навичок, обчислення семантичної подібності та ранжування вакансій у рекомендаційній системі.

Практична значущість роботи полягає у створенні прототипу full-stack вебзастосунку, який може бути адаптований для навчальних кар'єрних порталів, внутрішніх HR-сервісів, платформ пошуку роботи та систем підтримки професійного розвитку. Система розроблена як розширювана архітектура, тому в майбутньому до неї можна додати навчання на реальних відгуках користувачів, інтеграцію з зовнішніми джерелами вакансій, багатомовну обробку текстів та оцінювання якості рекомендацій на розмічених наборах даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність автоматизованого підбору вакансій

Процес пошуку роботи в цифровому середовищі значною мірою перетворився на задачу інформаційного пошуку та ранжування. Кандидат формує резюме, у якому описує освіту, досвід, проєкти, навички та інструменти, а роботодавець створює вакансію з описом завдань, вимог, бажаних компетенцій і умов праці. Обидва документи є текстовими й містять частково структуровану інформацію. Задача системи полягає у визначенні того, наскільки зміст резюме відповідає змісту вакансії [3], [4].

Традиційні підходи до пошуку вакансій зазвичай використовують фільтри за посадою, містом, рівнем досвіду або ключовими словами. Такі фільтри корисні для первинного звуження вибірки, проте вони погано працюють у випадках, коли релевантність залежить від контексту. Наприклад, кандидат може мати досвід розробки REST API у Spring Boot, а вакансія може описувати вимогу як «server-side API development». За простим ключовим словом ці фрагменти не обов'язково збігатимуться, хоча за змістом вони близькі [4].

У задачі підбору вакансій важливо враховувати декілька рівнів відповідності. Перший рівень – явний збіг навичок, коли назви технологій присутні і в резюме, і у вакансії. Другий рівень – семантична близькість формулювань, коли різні слова описують один і той самий професійний досвід. Третій рівень – відповідність професійної ролі, оскільки backend-розробник може мати частину frontend-навичок, але основна спеціалізація кандидата все одно має впливати на ранжування вакансій.

Для кандидата рекомендаційна система має виконувати не лише функцію пошуку, а й функцію професійної навігації. Якщо система показує, що вакансія має високий загальний бал, але містить відсутні навички, користувач може підготуватися до подачі, оновити резюме або обрати план

навчання. Таким чином рекомендація стає інструментом розвитку, а не тільки способом знайти оголошення.

Для роботодавців та кар'єрних сервісів така система також є корисною, оскільки вона зменшує кількість нерелевантних відгуків, допомагає кандидатам краще розуміти вимоги та підвищує якість взаємодії між сторонами ринку праці. Пояснюваність результатів є особливо важливою, адже рішення, пов'язані з працевлаштуванням, мають бути прозорими та контрольованими людиною. Узагальнене порівняння наведено в таблиці 1.1 [3], [8].

Таблиця 1.1 – Порівняння традиційного пошуку та рекомендаційного підходу

Критерій	Традиційний пошук за ключовими словами	Рекомендаційна система з NLP
Форма запиту	Користувач сам вводить ключові слова	Система аналізує повний текст резюме
Облік контексту	Обмежений, переважно точні збіги	Враховує семантичну подібність і роль кандидата
Пояснення результату	Зазвичай відсутнє	Показує збіги, прогалини та текстове пояснення
Користь для розвитку	Низька	Дає сигнал, які навички потрібно підсилити

Отже, автоматизація підбору вакансій потребує комбінування методів інформаційного пошуку, обробки природної мови та прикладного проєктування вебсистем. У межах роботи було обрано підхід, який поєднує швидкий словниковий аналіз, статистичне представлення TF-IDF та семантичні embedding-моделі [4], [5], [6].

1.2 Особливості текстових даних резюме та вакансій

Резюме та описи вакансій належать до слабкоструктурованих текстових даних. Вони описують пов'язані сутності ринку праці, однак

створюються з різною метою та мають неоднакову внутрішню організацію. Резюме орієнтоване на представлення професійного профілю кандидата, його досвіду, освіти, компетентностей і досягнень. Опис вакансії, навпаки, відображає потреби роботодавця та містить вимоги до майбутнього працівника, перелік завдань, умови праці й очікуваний рівень кваліфікації [8], [10].

Типове резюме може містити такі секції: особисті та контактні дані, короткий професійний профіль, досвід роботи, освіта, технічні та особистісні навички, реалізовані проєкти, знання іноземних мов, професійні сертифікати й додаткові відомості. Водночас єдиного обов'язкового стандарту структури резюме не існує. Кандидати можуть змінювати порядок секцій, використовувати власні заголовки, поєднувати декілька блоків або взагалі подавати інформацію у вигляді суцільного тексту.

Опис вакансії зазвичай містить назву посади, інформацію про компанію, перелік посадових обов'язків, обов'язкові та бажані вимоги, необхідний досвід роботи, технічні й комунікативні навички, умови працевлаштування, формат роботи, рівень заробітної плати та місце розташування компанії. Окремі роботодавці також зазначають інформацію про корпоративну культуру, соціальні переваги, можливості професійного розвитку та етапи відбору кандидатів.

Відмінність у призначенні документів формує інформаційну асиметрію між їхніми текстами. У резюме кандидат переважно описує вже отриманий досвід, використовуючи конструкції на зразок «розробляв», «реалізував», «використовував», «брав участь». У вакансії роботодавець формулює очікування за допомогою слів «необхідно», «вимагається», «буде перевагою», «кандидат повинен знати». Отже, навіть за наявності однакової компетенції відповідні фрагменти можуть мати різну граматичну та лексичну форму.

Наприклад, у резюме може бути зазначено: «Розробляв серверні застосунки за допомогою Spring Boot та PostgreSQL». У вакансії аналогічна

вимога може бути сформульована як «Досвід створення REST-сервісів мовою Java з використанням реляційних баз даних». Попри відсутність повного збігу слів, обидва речення описують близький професійний контекст. Тому пряме порівняння документів лише за точним збігом термінів не завжди дає коректний результат.

Текстові дані резюме та вакансій характеризуються значною неоднорідністю. Документи можуть відрізнятися за обсягом, мовою, стилем викладення, способом форматування та рівнем деталізації. Одне резюме може містити декілька сторінок із докладним описом кожного проєкту, тоді як інше складається лише з короткого переліку технологій. Аналогічно одна вакансія може містити чітко структурований список вимог, а інша – загальний рекламний опис компанії з мінімальною кількістю технічної інформації.

Додатковою проблемою є наявність текстового шуму. Під час вилучення тексту з файлів PDF або DOCX можуть зберігатися службові символи, номери сторінок, повторювані заголовки, розриви рядків, зайві пробіли та елементи колонтитулів. Текст також може містити різні маркери списків, нестандартні лапки, дефіси, спеціальні символи та неоднорідну пунктуацію. Такі елементи не несуть корисного змісту для рекомендаційної системи, але можуть впливати на результати автоматичного порівняння [4].

Важливою особливістю є варіативність написання назв технологій, мов програмування, платформ та інструментів. Наприклад, записи «PostgreSQL», «Postgres» і «Postgre SQL» можуть позначати одну систему керування базами даних. Варіанти «JavaScript», «Java Script» і «JS» також належать до однієї технології. Поняття «REST API», «RESTful API», «REST services» та «розробка вебсервісів» можуть описувати близьку компетенцію, хоча їхня текстова форма суттєво відрізняється [10], [11].

Подібна проблема виникає через використання аббревіатур і повних назв. У резюме може бути вказано «ML», тоді як у вакансії використовується словосполучення «machine learning». Скорочення «NLP»

може відповідати термінам «natural language processing» або «оброблення природної мови». Без приведення таких варіантів до спільного представлення система може помилково вважати, що відповідна навичка в резюме відсутня [10], [11].

Резюме та вакансії також можуть містити змішаний багатомовний текст. Наприклад, основна частина документа може бути написана українською мовою, а назви технологій, посад і сертифікатів – англійською. Крім того, назва однієї посади може бути сформульована як «розробник програмного забезпечення», «software developer», «software engineer» або «програміст». Тому автоматизована система повинна враховувати не лише точне написання термінів, а й можливі мовні відповідники.

Ще однією проблемою є неоднозначність окремих термінів. Слово «Java» може позначати мову програмування, однак збіг коротких назв або аббревіатур іноді виникає у нерелевантному контексті. Термін «React» може бути назвою бібліотеки, але в іншому контексті це звичайне англійське дієслово. Для зменшення кількості хибних збігів бажано враховувати сусідні слова, структуру речення та професійний контекст документа.

Під час аналізу резюме важливо розрізняти фактичне володіння технологією та її випадкову згадку. Наприклад, фраза «не маю досвіду роботи з Docker» містить назву технології, але не підтверджує відповідну компетенцію. Аналогічно кандидат може зазначити, що планує вивчати певну мову програмування або лише ознайомився з нею. Простий словниковий пошук може зарахувати такі згадки як наявні навички, тому перспективним напрямом удосконалення є аналіз заперечень і контексту використання термінів.

Значення має також актуальність професійного досвіду. Технологія може згадуватися в резюме в описі роботи, яка виконувалася багато років тому, або в короткому навчальному проєкті. Водночас інша технологія може регулярно використовуватися кандидатом на поточному місці роботи. Звичайна модель зіставлення за ключовими словами оцінює обидві згадки

однаково, хоча фактичний рівень компетентності може відрізнятися. Для точнішого аналізу можна враховувати тривалість досвіду, дату останнього використання навички, кількість її згадок та контекст проєктів.

Схожа проблема виникає під час оброблення вимог вакансій. Не всі зазначені технології мають однакову важливість. Частина навичок є обов'язковою, інша – бажаною, а деякі згадуються лише в описі продукту компанії. Наприклад, фраза «досвід роботи з Python є обов'язковим» повинна мати більшу вагу, ніж «знання Docker буде перевагою». Тому для подальшого розвитку системи доцільно розділяти обов'язкові та додаткові вимоги й застосовувати для них різні вагові коефіцієнти.

Перед використанням текстів у рекомендаційних алгоритмах виконується їх попереднє оброблення. Основними етапами є вилучення тексту з файлу, приведення символів до єдиного кодування, перетворення тексту до нижнього регістру, видалення зайвих пробілів і службових символів, нормалізація пунктуації та поділ тексту на окремі токени. Така обробка зменшує вплив форматування документа і забезпечує більш стабільне порівняння текстових даних [4].

Токенізація передбачає розбиття тексту на окремі слова, словосполучення або інші елементарні одиниці. Після токенізації можуть вилучатися стоп-слова, тобто поширені службові слова, які мають низьку інформативність для визначення професійної відповідності. До таких слів належать прийменники, сполучники, займенники та інші частотні елементи мови. Водночас вилучення стоп-слів необхідно виконувати обережно, оскільки окремі службові конструкції можуть впливати на зміст вимоги або вказувати на заперечення.

Для зменшення кількості різних граматичних форм одного слова може використовуватися стемінг або лематизація. Лематизація приводить слово до його початкової форми. Наприклад, слова «розробляв», «розробка» та «розробляти» можуть бути пов'язані з однією лемою. Це особливо важливо для української мови, яка характеризується значною кількістю

словозмінних форм. Проте назви технологій і торгових марок доцільно зберігати без зміни, щоб не спотворювати їх написання.

У розробленій системі для виявлення професійних компетенцій використовується словник навичок. Для кожної канонічної назви навички визначено набір альтернативних варіантів написання, аббревіатур і синонімів. Наприклад, канонічній навичці «PostgreSQL» можуть відповідати варіанти «postgres», «postgre sql» і «postgresql». Під час оброблення тексту знайдені варіанти приводяться до єдиної назви, що дає можливість порівнювати резюме та вакансії незалежно від особливостей написання [10], [11].

Перевагами словникового підходу є простота реалізації, висока швидкість роботи та пояснюваність отриманих результатів. Система може чітко визначити, яка саме навичка була знайдена в тексті резюме або вакансії. Словник також легко розширювати шляхом додавання нових технологій і варіантів їх написання. Такий підхід добре підходить для навчального прототипу, в якому важливо продемонструвати логіку формування рекомендацій.

Недоліком словникового підходу є його залежність від повноти сформованого словника. Якщо термін або його варіант відсутній у словнику, система може не розпізнати відповідну компетенцію. Крім того, словниковий аналіз переважно фіксує сам факт наявності терміна, але не визначає рівень володіння навичкою, тривалість досвіду та характер її використання. Він також обмежено враховує зміст речень і взаємозв'язки між різними компетенціями.

Для подолання зазначених обмежень у системі застосовуються методи статистичного та семантичного аналізу тексту. Одним із них є TF-IDF – статистична міра, яка оцінює важливість терміна в конкретному документі з урахуванням частоти його появи в усьому корпусі. Слова, які часто трапляються в одному документі, але рідко зустрічаються в інших,

отримують більшу вагу. Загальновживані слова, присутні у значній кількості документів, мають меншу вагу [4], [5].

Після перетворення текстів резюме та вакансій у TF-IDF-вектори їхню подібність можна визначати за допомогою косинусної міри. Вона оцінює кут між векторами документів і дозволяє встановити, наскільки близьким є їхній термінологічний склад. Перевагою TF-IDF є відносна простота, невисока обчислювальна складність і можливість визначити слова, які найбільше вплинули на результат зіставлення.

Водночас TF-IDF працює переважно на рівні окремих слів і словосполучень та не повною мірою враховує семантичну близькість різних формулювань. Наприклад, словосполучення «створення серверної частини вебзастосунків» та «backend development» можуть мати близький зміст, але слабкий лексичний збіг. Для виявлення таких залежностей у системі використовуються векторні представлення текстів, або embeddings.

Метод embeddings перетворює текстовий документ на числовий вектор фіксованої розмірності. Координати такого вектора формуються моделлю машинного навчання на основі змісту тексту. Документи зі схожим змістом розташовуються близько один до одного у векторному просторі навіть тоді, коли в них використано різні слова. Семантична подібність резюме та вакансії також може обчислюватися за допомогою косинусної міри між відповідними векторами.

Застосування embeddings дозволяє враховувати загальний професійний контекст документа, взаємозв'язки між технологіями та змістовну близькість речень. Наприклад, модель може встановити зв'язок між досвідом розроблення вебсервісів і вимогою створювати REST API, навіть якщо точні формулювання не збігаються. Це підвищує якість рекомендацій порівняно з використанням лише словникового пошуку.

Разом із тим семантичні моделі мають певні обмеження. Результат їх роботи складніше пояснити користувачеві, ніж простий збіг навичок. Крім того, якість векторного представлення залежить від використаної моделі,

мови тексту, довжини документа та специфіки предметної області. Модель загального призначення може недостатньо точно враховувати вузькоспеціалізовану технічну термінологію [6], [7].

Для отримання збалансованого результату в розробленій системі поєднуються декілька підходів. Словниковий аналіз забезпечує точне виявлення конкретних професійних навичок, TF-IDF оцінює лексичну подібність документів, а embeddings визначають їхню семантичну близькість. Загальна оцінка відповідності формується з урахуванням результатів цих методів.

Комбінований підхід знижує залежність рекомендацій від одного алгоритму. Якщо в резюме й описі вакансії використовуються однакові назви технологій, це відображається в оцінці ключових слів. Якщо формулювання відрізняються, але документи мають близький зміст, відповідність може бути виявлена за допомогою семантичного аналізу. TF-IDF при цьому виступає проміжним методом, який враховує важливість термінів і загальну схожість словникового складу.

Окремою вимогою до рекомендаційної системи є пояснюваність отриманих результатів. У деяких рекомендаційних задачах користувачеві достатньо отримати впорядкований список об'єктів. Однак у сфері працевлаштування кандидату важливо розуміти, чому певна вакансія була рекомендована та які фактори вплинули на її позицію в рейтингу.

Для цього кожна рекомендація в системі містить загальний бал відповідності, бал за ключовими словами, семантичний бал, перелік знайдених навичок, перелік відсутніх навичок і текстове пояснення. Загальний бал відображає інтегральну оцінку відповідності резюме вимогам вакансії. Окремі складові дають змогу зрозуміти, чи рекомендація сформована переважно через точний збіг професійних компетенцій або через загальну змістову близькість документів.

Перелік *matched skills* містить навички, які одночасно виявлено в резюме кандидата та вимогах вакансії. Цей список демонструє сильні

сторони відповідності. Перелік missing skills містить компетенції, зазначені у вакансії, але не знайдені в резюме. Така інформація може бути використана кандидатом для визначення напрямів подальшого професійного розвитку або для уточнення змісту резюме, якщо відповідна навичка фактично є, але не була в ньому зазначена.

Текстове пояснення узагальнює основні фактори рекомендації зрозумілою для користувача мовою. Наприклад, система може повідомити, що вакансія добре відповідає основному профілю кандидата, але потребує додаткових знань певної технології. Такий підхід підвищує прозорість рекомендацій і сприяє формуванню довіри користувача до результатів автоматизованого аналізу.

Якість рекомендацій безпосередньо залежить від якості початкових текстових даних. Неповне або надто коротке резюме може не містити достатньої інформації для коректного зіставлення. Аналогічно нечіткий опис вакансії без конкретного переліку вимог ускладнює формування об'єктивної оцінки. Тому отриманий бал відповідності слід розглядати як допоміжний показник, а не як остаточне рішення щодо професійної придатності кандидата.

Рекомендаційна система не оцінює особисті якості кандидата, його фактичну продуктивність, мотивацію або здатність адаптуватися до колективу. Вона аналізує лише інформацію, наявну в тексті резюме та описі вакансії. Через це відсутність певної навички в резюме не завжди означає, що кандидат нею не володіє. Так само згадування технології не гарантує високого рівня практичної компетентності.

Отже, тексти резюме та вакансій характеризуються слабкою структурованістю, значною варіативністю формулювань, багатомовністю, неоднорідністю форматування та наявністю контекстно залежних термінів. Для їх коректного автоматизованого аналізу необхідно поєднувати попереднє оброблення тексту, нормалізацію назв навичок, статистичні методи та семантичні моделі. Використання словникового пошуку, TF-IDF

та embeddings дозволяє врахувати як точні термінологічні збіги, так і загальну змістову подібність документів. Додавання механізмів пояснюваності забезпечує прозорість отриманих рекомендацій та підвищує практичну цінність розробленої системи [4], [6], [10].

1.3 Огляд рекомендаційних систем і методів NLP

Рекомендаційні системи зазвичай поділяють на колаборативні, контентні та гібридні. Колаборативні методи використовують історію взаємодій користувачів з об'єктами, наприклад кліки або відгуки. У задачі підбору вакансій такий підхід потребує великої кількості історичних даних і не завжди доступний для нового користувача або нового проєкту. Тому в розробленій роботі основний акцент зроблено на контентному підході, який аналізує зміст резюме і вакансій [3].

Контентна рекомендація в цій задачі фактично зводиться до порівняння двох текстових документів. Для цього можна застосовувати класичні методи інформаційного пошуку, зокрема векторну модель документа та TF-IDF. Кожен документ подається у вигляді вектора, де координати відповідають словам або словосполученням. Подібність документів можна обчислювати за допомогою косинусної міри.

Перевага TF-IDF полягає у простоті, швидкості та прозорості. Недоліком є залежність від поверхневого збігу лексики. Якщо резюме і вакансія використовують різні формулювання, TF-IDF може недооцінити відповідність. Для подолання цієї проблеми застосовуються embedding-моделі, які перетворюють речення або документи у щільні вектори. Семантично близькі тексти мають близькі вектори навіть без повного збігу слів [5].

У сучасних системах для побудови sentence embeddings широко застосовуються трансформерні моделі. У роботі використано підхід Sentence Transformers, зокрема модель all-MiniLM-L6-v2. Вона формує 384-

вимірні вектори для речень і абзаців, що робить її придатною для семантичного пошуку та порівняння текстів [14], [15].

Однак для задачі підбору вакансій недостатньо тільки семантичної подібності. Наприклад, опис frontend- і backend-вакансій може містити багато спільних слів про веброзробку, але професійна роль кандидата має відрізняти ці напрями. Тому в роботі використано role-aware scoring: система визначає домен резюме і домен вакансії, після чого застосовує бонус або штраф залежно від їх відповідності [8], [9].

1.4 Аналіз аналогів та вимог до системи

Рекомендаційні системи призначені для автоматизованого відбору та ранжування об'єктів відповідно до потреб користувача. У задачі працевлаштування такими об'єктами є вакансії, а основою для формування рекомендацій виступають професійний досвід, освіта, навички та інші відомості, зазначені в резюме кандидата.

Рекомендаційні системи зазвичай поділяють на колаборативні, контентні та гібридні. Колаборативні методи використовують історію взаємодії користувачів з об'єктами. Для сервісу пошуку роботи такими взаємодіями можуть бути перегляди вакансій, додавання їх до обраного, надсилання резюме та відгуки на пропозиції роботодавців. На основі цієї інформації система визначає користувачів зі схожою поведінкою та рекомендує їм подібні вакансії [3].

Основним недоліком колаборативного підходу є необхідність накопичення значної кількості історичних даних. Для нового користувача, який ще не переглядав вакансії, або для нової вакансії, яка не має взаємодій, система не може сформувати достатньо точні рекомендації. Це явище називають проблемою «холодного старту». Для навчального прототипу та нового програмного продукту такий підхід є складним у застосуванні через відсутність великої активної аудиторії.

Контентні рекомендаційні системи використовують характеристики користувача й об'єктів. У задачі підбору вакансій профіль кандидата представлений текстом резюме, а характеристики вакансії – її назвою, описом, обов'язками та вимогами. Система аналізує зміст цих документів, визначає їхню відповідність і формує впорядкований список вакансій.

Перевагою контентного підходу є можливість отримувати рекомендації одразу після завантаження резюме. Для цього не потрібно накопичувати історію переглядів або відгуків користувача. Крім того, контентна система дозволяє пояснити, які навички та фрагменти тексту вплинули на отриманий результат.

Недоліком такого підходу є залежність від якості текстових даних. Якщо резюме містить мало інформації або опис вакансії сформульований надто загально, система може неточно визначити рівень відповідності. Також контентні рекомендації можуть бути надмірно вузькими, оскільки система переважно пропонує вакансії, схожі на вже зазначений у резюме професійний профіль.

Гібридні рекомендаційні системи поєднують контентні та колаборативні методи. Наприклад, аналіз резюме може доповнюватися історією переглядів вакансій і діями користувача. Такий підхід потенційно підвищує точність рекомендацій, однак потребує складнішої архітектури та більшого обсягу даних.

У розробленій системі основний акцент зроблено на контентному підході. Такий вибір пояснюється тим, що система повинна формувати рекомендації одразу після завантаження резюме та не залежати від наявності історії дій користувача.

Контентна рекомендація в цій задачі зводиться до автоматизованого порівняння тексту резюме з текстами вакансій. Для цього застосовуються методи оброблення природної мови, або Natural Language Processing. NLP охоплює методи вилучення, очищення, нормалізації, аналізу та числового представлення тексту [4].

Перед порівнянням документів виконується попереднє оброблення тексту. Воно включає приведення символів до нижнього регістру, видалення зайвих пробілів і службових знаків, розбиття тексту на окремі елементи та нормалізацію назв технологій. Також виконується пошук професійних навичок за словником, який містить канонічні назви й альтернативні варіанти їх написання [4].

Найпростішим способом аналізу є пошук збігів ключових слів і навичок. Система визначає, які технології присутні одночасно в резюме та описі вакансії. Наприклад, якщо в обох документах зазначено Python, PostgreSQL і Docker, це підвищує оцінку відповідності. Перевагами такого підходу є простота, швидкість і зрозумілість результату.

Водночас пошук ключових слів не враховує загальний зміст тексту. Одна й та сама навичка може бути описана різними словами. Наприклад, «створення серверної частини застосунку» та «backend development» мають близьке значення, але не містять однакових формулювань. Через це застосування лише словникового підходу може призвести до пропуску релевантних вакансій.

Для оцінювання лексичної подібності документів використовується метод TF-IDF. Він надає більшу вагу словам, які є важливими для конкретного документа, але нечасто зустрічаються в інших текстах. Загальні слова, характерні для більшості вакансій, мають менший вплив на результат, тоді як назви мов програмування, бібліотек і технологій отримують більшу вагу [4], [5], [12].

Після перетворення резюме та вакансій у числові вектори визначається ступінь їхньої подібності. Чим більше важливих термінів збігається в документах, тим вищою є оцінка відповідності. TF-IDF характеризується високою швидкістю роботи та відносною прозорістю, оскільки можна визначити, які слова вплинули на результат.

Недоліком TF-IDF є орієнтація переважно на лексичні збіги. Метод не розуміє значення речень і може надати низьку оцінку текстам, які мають

близький зміст, але використовують різні слова. Для розв'язання цієї проблеми застосовуються семантичні векторні представлення тексту, або embeddings.

Embedding-моделі перетворюють речення, абзаци або повні документи на щільні числові вектори. Семантично близькі тексти отримують близькі векторні представлення навіть за відсутності точного збігу слів. Це дозволяє системі знаходити вакансії, зміст яких відповідає професійному досвіду кандидата, хоча формулювання в документах відрізняються.

У сучасних системах для побудови таких представлень широко використовуються трансформерні моделі. Вони враховують контекст слів і взаємозв'язки між частинами речення. Завдяки цьому модель може точніше визначати зміст професійного досвіду та вимог вакансії.

У роботі застосовано підхід Sentence Transformers і модель all-MiniLM-L6-v2. Вона формує 384-вимірні вектори для текстових фрагментів і забезпечує прийнятний баланс між швидкістю та якістю семантичного аналізу. Модель може використовуватися для порівняння резюме з великою кількістю вакансій і визначення найбільш змістовно близьких документів [6].

Перевагою Sentence Transformers є можливість заздалегідь обчислити векторні представлення вакансій і зберігати їх у системі. Після завантаження нового резюме необхідно сформувати вектор лише для нього, а потім порівняти з уже підготовленими векторами вакансій. Це зменшує час формування рекомендацій.

Однак використання тільки семантичної подібності не завжди забезпечує правильне визначення професійного напрямку. Наприклад, frontend- і backend-вакансії можуть містити спільні слова про веброзробку, API, командну роботу та програмне забезпечення. Семантична модель може оцінити їх як близькі, хоча вони потребують різного набору основних компетенцій.

Для врахування цього аспекту в системі реалізовано механізм *role-aware scoring*. Він визначає професійний домен резюме та вакансії, наприклад *frontend*, *backend*, *fullstack*, *machine learning*, *data analysis*, *DevOps* або тестування програмного забезпечення. Після цього початкова оцінка коригується залежно від відповідності професійних напрямів [8], [9].

Якщо домен резюме відповідає домену вакансії, система підвищує підсумковий бал. Для суміжних напрямів оцінка може змінюватися незначно. Якщо професійні ролі суттєво відрізняються, застосовується зменшення бала. Це дозволяє уникнути ситуацій, коли вакансія є схожою за загальною тематикою, але не відповідає спеціалізації кандидата.

Загальна оцінка формується на основі декількох складових: збігу ключових навичок, лексичної подібності, семантичної близькості та відповідності професійної ролі. Поєднання цих методів зменшує залежність результату від одного алгоритму та підвищує стійкість рекомендацій.

Важливою вимогою є пояснюваність результатів. Користувач повинен розуміти, чому система рекомендувала конкретну вакансію. Тому разом із загальним балом відображаються знайдені навички, відсутні компетенції, окремі складові оцінювання та коротке текстове пояснення.

Перелік знайдених навичок демонструє сильні сторони відповідності кандидата вакансії. Перелік відсутніх навичок показує вимоги, які були зазначені роботодавцем, але не знайдені в резюме. Така інформація допомагає користувачеві оцінити власну готовність до вакансії та визначити напрями подальшого навчання.

Отриманий бал не слід розглядати як остаточне рішення щодо працевлаштування кандидата. Система аналізує лише текстові дані та не може повністю оцінити мотивацію, особисті якості, реальний рівень практичного досвіду й здатність працювати в команді. Рекомендації мають допоміжний характер і призначені для скорочення часу пошуку релевантних вакансій.

Отже, для розробленої системи обрано контентний підхід, який не потребує значного обсягу історичних даних і дозволяє формувати рекомендації одразу після завантаження резюме. Для аналізу текстів поєднано пошук ключових навичок, TF-IDF, семантичні embeddings і механізм role-aware scoring. Така комбінація дозволяє враховувати точні збіги технологій, загальну змістову близькість документів і відповідність професійного напрямку кандидата вакансії [3].

1.5 Постановка задачі дослідження

Нехай задано текст резюме R та множину активних вакансій $V = \{v_1, v_2, \dots, v_n\}$. Кожна вакансія містить назву, компанію, опис, вимоги, рівень досвіду та додаткові ознаки. Необхідно для кожної вакансії v_i обчислити оцінку відповідності $\text{score}(R, v_i)$ у межах $[0, 1]$, відсортувати вакансії за спаданням цієї оцінки та подати користувачеві top- K рекомендацій.

Оцінка має враховувати: частку навичок вакансії, знайдених у резюме; семантичну подібність тексту резюме та опису вакансії; відповідність професійного домену кандидата домену вакансії; відповідність рівня досвіду; наявність критичних відсутніх навичок. Результат повинен бути не тільки числовим, а й інтерпретованим.

Вхідними даними є файл резюме, активні вакансії та параметри стратегії зіставлення. Вихідними даними є список рекомендацій, де кожний елемент містить ідентифікатор вакансії, загальний бал, компонентний бал за ключовими словами, семантичний бал, списки знайдених і відсутніх навичок, пояснення та метадані обраної стратегії.

Таким чином, задача кваліфікаційної роботи має як дослідницьку, так і інженерну складову.

2 МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Попередня обробка текстових документів

Попередня обробка тексту є першим етапом рекомендаційного конвеєра. Її мета – зменшити варіативність запису однакових понять і підготувати дані до виділення навичок та векторизації. У розробленій системі попередня обробка має просту послідовність: приведення тексту до нижнього регістру, видалення зайвої пунктуації, заміна маркерів списків і тире пробілами, згортання повторних пробілів і видалення пробілів на початку та в кінці рядка [12].

Під час проєктування preprocessing-модуля важливо було не зруйнувати назви технологій. Наприклад, символи «+» та крапка мають значення у назвах C++, Node.js або .NET. Тому вони не видаляються повністю, а обробляються обережно. Такий підхід зберігає корисну інформацію і водночас прибирає шум, що заважає порівнянню.

Токенізація виконується розбиттям попередньо очищеного тексту за пробілами. Для кожного токена видаляються крапки на початку та в кінці, але внутрішні крапки залишаються. Це дозволяє коректно обробляти речення і не втрачати назви технологій із внутрішньою пунктуацією.

Етап попередньої обробки є спільним для різних стратегій. Keyword matching використовує токени для пошуку навичок, TF-IDF отримує очищений текст для побудови матриці термінів, а embedding-стратегія передає нормалізований текст у модель кодування речень. Уніфікація preprocessing підвищує стабільність результатів і спрощує подальше тестування.

2.2 Виділення навичок на основі словника

Словникове виділення навичок використовується як базовий і пояснюваний компонент системи. Для кожної навички формується

канонічна назва та список синонімів або варіантів написання. Наприклад, для навички JavaScript можна враховувати варіанти «js», «ecmascript», а для PostgreSQL – «postgres». Після токенизації система перевіряє, чи входить кожний alias до послідовності токенів документа [11].

Для уникнення хибних спрацювань пошук виконується на рівні токенів, а не простим пошуком підрядка. Це важливо для коротких технологічних назв. Наприклад, слово «go» не повинно збігатися з частиною слова «ongoing». Для багатослівних назв використовується перевірка входження неперервної послідовності токенів.

Результатом виділення є множина канонічних навичок. Для резюме ця множина описує компетенції кандидата, а для вакансії – вимоги роботодавця. Перетин множин дає matched skills, різниця між вимогами вакансії та навичками резюме дає missing skills. Саме ці два списки є основою пояснення рекомендації.

Перевага словникового підходу полягає у простоті контролю та розширення. Нову технологію можна додати до словника без перенавчання моделі. Недолік полягає у залежності від повноти словника, тому в майбутньому доцільно інтегрувати автоматичне виділення навичок на основі NER або спеціалізованої таксономії навичок.

Нехай S_R – множина навичок, знайдених у резюме, а S_V – множина навичок, потрібних вакансії. Тоді множина збігів і множина прогалин визначаються за формулою (2.1): [10].

$$S_{\text{match}} = S_R \cap S_V; \quad S_{\text{missing}} = S_V \setminus S_R, \quad (2.1)$$

де S_R – множина навичок, знайдених у резюме;

S_V – множина навичок, потрібних вакансії;

S_{match} – множина збігів;

S_{missing} – множина відсутніх навичок.

Покриття вимог вакансії навичками кандидата визначається за формулою (2.2):

$$\text{coverage} = |S_{\text{match}}| / |S_V|, \text{ якщо } |S_V| > 0, \quad (2.2)$$

де coverage – частка покриття вимог вакансії навичками кандидата.

Цей показник інтерпретується як частка вимог вакансії, яку кандидат уже покриває своїми навичками.

2.3 Метод TF-IDF та косинусна подібність

TF-IDF є класичним методом зважування термінів у задачах інформаційного пошуку. Ідея методу полягає в тому, що слово є важливим для документа, якщо воно часто зустрічається в цьому документі, але рідко зустрічається в інших документах корпусу. Для задачі порівняння резюме і вакансій корпус формується з тексту резюме та текстів усіх активних вакансій [4], [5].

Компонента TF показує частоту терміна у документі, а компонента IDF зменшує вагу термінів, які трапляються у багатьох документах. Наприклад, загальні слова «development», «team», «experience» можуть бути менш інформативними, ніж конкретні технології «FastAPI», «PostgreSQL», «Spring Boot».

Після побудови TF-IDF матриці кожен документ подається як вектор. Для оцінки подібності між резюме і вакансією використовується косинусна подібність, яка вимірює кут між векторами. Значення близьке до 1 означає високу схожість, а значення близьке до 0 – низьку схожість.

У розробленій системі TF-IDF стратегія комбінується зі словниковим покриттям і рольовим вирівнюванням. Це дозволяє не покладатися лише на текстову схожість, а враховувати конкретні навички та професійний домен кандидата.

Загальна оцінка для TF-IDF-стратегії в роботі задається як зважена сума компонентів за формулою (2.3):

$$\text{score}_{\text{tfidf}} = 0,30 \cdot \text{coverage} + 0,25 \cdot \text{role}_{\text{alignment}} + 0,45 \cdot \text{tfidf}_{\text{cosine}} \quad (2.3)$$

де score_tfidf – підсумкова оцінка для TF-IDF-стратегії;

tfidf_cosine – косинусна подібність TF-IDF-векторів;

role_alignment – коефіцієнт відповідності професійних ролей.

Коефіцієнти було підібрано евристично так, щоб семантична близькість тексту мала найбільшу вагу, але не витісняла повністю інформацію про конкретні навички та рольову відповідність.

2.4 Семантичні векторні представлення речень

Embedding-підхід перетворює текстові фрагменти на щільні числові вектори. На відміну від TF-IDF, такі вектори можуть відображати семантичну близькість текстів навіть тоді, коли вони не мають великої кількості однакових слів. Наприклад, фрази «built REST services» і «API development» можуть отримати близькі векторні представлення.

У роботі використовується модель all-MiniLM-L6-v2 із бібліотеки sentence-transformers. Вона формує 384-вимірні вектори для речень і абзаців та призначена для задач семантичного пошуку, кластеризації і порівняння текстів. Для кожного резюме та кожної вакансії модель обчислює embedding, після чого подібність визначається косинусною мірою [6], [14], [15].

Embedding-стратегія має вищу обчислювальну вартість, ніж keyword або TF-IDF, оскільки потребує завантаження трансформерної моделі та виконання нейромережевого кодування. У системі модель завантажується ліниво при першому використанні та кешується, що зменшує затримку для наступних запитів.

Перевагою цього методу є можливість враховувати зміст опису досвіду, а не тільки перелік технологій. Недоліком є складність інтерпретації векторів. Тому embedding-стратегія в системі все одно поєднується зі списками matched/missing skills, які забезпечують зрозуміле пояснення користувачу.

$$\text{score}_{\text{embed}} = 0,20 \cdot \text{coverage} + 0,25 \cdot \text{role}_{\text{alignment}} + 0,55 \cdot \text{embedding}_{\text{cosine}} \quad (2.4)$$

де $\text{score}_{\text{embed}}$ – підсумкова оцінка для embedding-стратегії;

$\text{embedding}_{\text{cosine}}$ – косинусна подібність семантичних векторів резюме і вакансії.

У цій формулі найбільшу вагу має $\text{embedding}_{\text{cosine}}$, оскільки головна мета стратегії – оцінити змістову близькість резюме і вакансії. Водночас coverage і $\text{role}_{\text{alignment}}$ залишаються важливими обмежувачами, які підвищують прикладну релевантність результату.

2.5 Рольове коригування оцінки відповідності

У задачі підбору вакансій важливо розуміти не тільки перелік технологій, а й професійний домен кандидата. Резюме може містити навички з різних напрямів, але основна роль кандидата має визначати пріоритет рекомендацій. Наприклад, fullstack-розробник може бути релевантним і до backend, і до frontend вакансій, тоді як data engineer має отримувати нижчий бал для чисто frontend-позицій.

Для цього в системі використовується аналізатор профілю резюме. Він визначає домен кандидата за набором core skills і ключових фраз. Серед доменів передбачено backend, frontend, fullstack, data, devops, mobile, embedded, security і qa. Для кожного домену задано набір ключових навичок і фраз, які підвищують імовірність належності резюме або вакансії до цього напрямку.

Рольове вирівнювання має три рівні. Якщо домен резюме і домен вакансії збігаються, коефіцієнт дорівнює 1.0. Якщо домени суміжні, наприклад backend і fullstack або backend і devops, використовується частковий коефіцієнт 0.55. Якщо домени не пов'язані, застосовується низьке значення 0.2. Така схема є простою, але дозволяє зменшити кількість очевидно нерелевантних рекомендацій.

Додатково враховується рівень досвіду. Якщо кандидат має junior-рівень, вакансії senior-рівня штрафуються; якщо кандидат має senior-рівень, mid-вакансії можуть отримати невеликий бонус або незначне зниження залежно від логіки стратегії. Це робить оцінку більш наближеною до реальних умов ринку праці.

2.6 Формування пояснюваних рекомендацій

Пояснюваність є однією з ключових вимог до системи. Користувач має бачити не тільки відсоток відповідності, а й зрозумілу причину такого результату. У розробленій системі пояснення формується автоматично після обчислення компонентів оцінки.

Текст пояснення містить загальну характеристику якості збігу, інформацію про рольову відповідність, рівень досвіду, частку покриття навичок, перелік основних сильних сторін та рекомендації щодо навичок, які варто підсилити. Якщо вакансія містить обов'язкові навички, яких немає в резюме, система прямо вказує на критичні прогалини.

Пояснення створюється на основі правил, а не генерується великою мовною моделлю. Це має дві переваги: результат є стабільним і легко контролюється; система не вигадує навички, яких не було у вхідних документах. Для дипломного прототипу такий підхід є достатнім і безпечним.

У майбутньому цей модуль можна розширити, додавши персоналізовані поради з навчання, інтеграцію з каталогами курсів або

автоматичне оновлення резюме на основі знайдених прогалин. Однак базова версія вже виконує головну функцію – робить рекомендацію прозорою для користувача.

Крім того, пояснюваність рекомендацій підвищує практичну цінність системи для користувача, оскільки дає змогу не лише переглянути список найбільш релевантних вакансій, а й зрозуміти логіку їх ранжування. Наприклад, кандидат може побачити, що певна вакансія має високий бал через збіг основних технологій, але водночас потребує додаткового досвіду з окремим інструментом або фреймворком. Завдяки цьому користувач отримує не абстрактну оцінку, а конкретний орієнтир для подальшого професійного розвитку.

Важливою перевагою такого підходу є також можливість перевірки правильності роботи системи під час тестування. Оскільки кожне пояснення пов'язане з конкретними знайденими та відсутніми навичками, розробник може проаналізувати, чи коректно система інтерпретувала текст резюме та опис вакансії. Це спрощує виявлення помилок у словнику навичок, правилах оцінювання або механізмі рольового коригування та робить систему більш придатною до подальшого вдосконалення.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Вимоги до програмного забезпечення

Розроблена система має бути вебзастосунком, доступним через браузер. Основними користувачами є кандидати, які завантажують резюме та отримують рекомендації вакансій. Додатково система може мати адміністративні сценарії створення та редагування вакансій. У межах прототипу основний фокус зроблено на користувацькому сценарії: реєстрація, вхід, завантаження резюме, генерація рекомендацій і перегляд результатів.

Функціональні вимоги визначають, які дії має підтримувати система. Нефункціональні вимоги визначають обмеження до якості: швидкість, безпека, розширюваність, зрозумілість інтерфейсу, модульність коду і простота розгортання. Для навчального проєкту особливо важливо, щоб архітектура була зрозумілою та дозволяла показати повний цикл від обробки файлу до виведення рекомендацій.

Вимоги до даних включають збереження користувачів, резюме, вакансій і рекомендацій. Резюме зберігається як файл і як витягнутий текст. Вакансія містить загальний опис і структуровані поля, такі як `required_skills`, `nice_to_have_skills`, `industry_domain`, `experience_level`. Рекомендація зберігає не тільки бал, а й детальну структуру пояснення. Склад функціональних вимог подано в таблиці 3.1.

Таблиця 3.1 – Функціональні вимоги до системи

ID	Функціональна вимога	Пріоритет
FR-1	Реєстрація користувача та вхід у систему	Високий
FR-2	Завантаження резюме у форматах PDF, DOCX, TXT	Високий
FR-3	Автоматичне витягування тексту з файлу резюме	Високий
FR-4	Перегляд списку активних вакансій	Середній
FR-5	Генерація рекомендацій за обраною стратегією	Високий

ID	Функціональна вимога	Пріоритет
FR-6	Відображення загального балу, matched/missing skills і пояснення	Високий
FR-7	Збереження результатів рекомендацій у базі даних	Високий
FR-8	Можливість вибору top-K результатів	Середній

Функціональні вимоги визначають основні дії, які система повинна виконувати для користувача: реєстрацію, завантаження резюме, аналіз тексту, генерацію рекомендацій та відображення результатів. Однак для практичного використання програмного продукту важливими є не лише його функції, а й якісні характеристики реалізації. Тому додатково було сформовано нефункціональні вимоги, які описують модульність, розширюваність, безпеку, зручність запуску та придатність системи до подальшого тестування й розвитку. Перелік нефункціональних вимог наведено в таблиці 3.2.

Таблиця 3.2 – Нефункціональні вимоги до системи

ID	Нефункціональна вимога	Реалізаційний підхід
NFR-1	Модульність алгоритмів	Strategy pattern для matching-модулів
NFR-2	Розширюваність backend	Service layer і repository layer
NFR-3	Зручний запуск	Docker Compose для PostgreSQL, backend і frontend
NFR-4	Безпека доступу	JWT, перевірка власника резюме
NFR-5	Пояснюваність	Збереження matched_skills, missing_skills, explanation
NFR-6	Придатність до тестування	Відокремлення бізнес-логіки від API-роутів

3.2 Архітектура вебзастосунку

Архітектура системи побудована за принципом розділення клієнтської та серверної частин. Frontend реалізовано на Next.js і працює на порту 3000. Backend реалізовано на FastAPI і працює на порту 8000. Дані

зберігаються у PostgreSQL, доступ до якого виконується через SQLAlchemy ORM. Для локального запуску всі компоненти об'єднано у Docker Compose [20].

Backend має багат шарову структуру. API-шар відповідає за HTTP endpoints, перевірку параметрів і взаємодію з користувачем. Service layer містить бізнес-логіку: завантаження резюме, витягування тексту, генерацію рекомендацій. Repository layer інкапсулює доступ до бази даних. AI layer містить алгоритми зіставлення та допоміжні модулі обробки тексту [16], [18].

Такий поділ зменшує зв'язність коду. Наприклад, RecommendationService не залежить від конкретної реалізації TF-IDF або embedding-стратегії. Він викликає RecommendationEngine, передаючи текст резюме, вакансії та назву стратегії. Engine через factory отримує потрібний алгоритм і повертає уніфіковані MatchScore об'єкти.

Frontend відповідає за відображення сторінок: головної, входу, реєстрації, завантаження резюме, перегляду вакансій, dashboard і рекомендацій. Взаємодія з backend виконується через API service modules. Такий підхід дозволяє тримати UI-компоненти відокремленими від деталей HTTP-запитів.

3.3 Структура бази даних

База даних системи містить сутності користувачів, резюме, вакансій і рекомендацій. Користувач може мати декілька резюме, але під час генерації рекомендацій використовується останнє завантажене резюме. Вакансії зберігаються як активні або неактивні. Рекомендація пов'язує користувача з конкретною вакансією і містить результати обчислення.

Модель Resume зберігає original_filename, file_path, extracted_text і дату завантаження. Поле extracted_text є ключовим для AI-модуля, оскільки саме воно передається на аналіз. У майбутньому модель може бути

розширена структурованими полями `parsed_skills`, `experience_years` і `education`.

Модель `Vacancy` містить не тільки `title`, `company`, `description` і `requirements`, а й додаткові JSON-поля `required_skills`, `nice_to_have_skills`, `tools_stack`, `languages_stack`, `human_languages`. Це дозволяє поєднувати неструктурований текст з напівструктурованими даними.

Модель `Recommendation` зберігає `overall_score`, `keyword_score`, `semantic_score`, `matched_skills`, `missing_skills`, `explanation` і `strategy`. Збереження результатів дає можливість швидко показувати попередні рекомендації без повторного запуску моделі та аналізувати історію генерації. Основні сутності бази даних наведено в таблиці 3.3.

Таблиця 3.3 – Основні сутності бази даних

Сутність	Основні поля	Призначення
User	id, email, hashed_password, is_active	Облікові записи користувачів
Resume	user_id, original_filename, file_path, extracted_text, uploaded_at	Завантажені резюме і витягнутий текст
Vacancy	title, company, description, requirements, required_skills, experience_level	Опис вакансій і вимоги роботодавця
Recommendation	user_id, vacancy_id, overall_score, keyword_score, semantic_score, explanation	Збережені результати підбору

3.4 Проєктування backend-частини

Backend-частина реалізує REST API. Основні групи маршрутів відповідають автентифікації, роботі з резюме, вакансіями та рекомендаціями. Кожна група маршрутів винесена в окремий файл, після чого підключається до центрального router registry. Це спрощує навігацію у коді та полегшує розширення API [16], [21].

Сервіс `ResumeService` відповідає за перевірку типу файлу, обмеження розміру, збереження файлу, витягування тексту та створення запису в базі

даних. Підтримуються розширення PDF, DOCX і TXT, максимальний розмір файлу становить 10 МБ. Якщо витягування тексту завершується помилкою, збережений файл видаляється, щоб не залишати неконсистентні дані.

`RecommendationService` реалізує повний сценарій генерації рекомендацій: отримує останнє резюме користувача, завантажує активні вакансії, формує словники вакансій, запускає `RecommendationEngine`, видаляє попередні рекомендації користувача, зберігає нові результати та повертає top-K записів із підвантаженими вакансіями.

`RecommendationEngine` є точкою входу до AI-пайплайна. Він не містить конкретної логіки обчислення балів, а делегує її вибраній стратегії. Наявність такого шару робить систему гнучкою: для додавання нової стратегії достатньо реалізувати клас з методом `match` і зареєструвати його у `factory`. Основні API endpoints системи наведено в таблиці 3.4.

Таблиця 3.4 – Основні API endpoints системи

Endpoint	Метод	Призначення	Доступ
<code>/api/v1/health</code>	GET	Перевірка доступності backend	Публічний
<code>/api/v1/auth/register</code>	POST	Реєстрація користувача	Публічний
<code>/api/v1/auth/login</code>	POST	Вхід і отримання JWT	Публічний
<code>/api/v1/resumes/upload</code>	POST	Завантаження резюме	Авторизований
<code>/api/v1/resumes</code>	GET	Список резюме користувача	Авторизований
<code>/api/v1/vacancies</code>	GET	Список активних вакансій	Публічний
<code>/api/v1/recommendations/generate</code>	POST	Генерація рекомендацій	Авторизований
<code>/api/v1/recommendations/me</code>	GET	Перегляд рекомендацій користувача	Авторизований

3.5 Проектування frontend-частини

Frontend реалізовано як Next.js застосунок з App Router. Основними сторінками є головна сторінка, форма входу, форма реєстрації, сторінка завантаження резюме, сторінка вакансій, dashboard та сторінка

рекомендацій. UI має мінімалістичний стиль: світлий фон, центральна колонка контенту, сині акцентні кнопки, картки з тонкою рамкою і достатніми відступами.

Головна сторінка пояснює користувачу основний сценарій: завантажити резюме, переглянути вакансії, отримати рекомендації. Сторінка завантаження резюме містить dropzone з підтримкою drag-and-drop і кнопку відправки. Сторінка рекомендацій містить вибір стратегії, кнопку генерації та список карток вакансій із балами.

Картка рекомендації відображає загальний відсоток, позицію в рейтингу, назву вакансії, компанію, локацію, компоненти score у вигляді горизонтальних індикаторів, matched skills зеленими тегами, missing skills помаранчевими тегами і текстове пояснення. Це робить результат не тільки інформативним, а й зручним для швидкого перегляду.

Frontend взаємодіє з backend через service modules. Такий підхід дозволяє ізолювати HTTP-логіку від візуальних компонентів і спрощує подальшу заміну API клієнта, додавання обробки помилок або кешування результатів.

3.6 Безпека та розгортання

Безпека розробленої системи базується на автентифікації користувачів, авторизації запитів і контролі доступу до персональних ресурсів. Дії, пов'язані із завантаженням, збереженням і переглядом резюме, а також формуванням та отриманням персоналізованих рекомендацій, доступні лише авторизованому користувачу. Такий підхід є обов'язковим, оскільки резюме може містити персональні дані: ім'я та прізвище кандидата, контактну інформацію, відомості про освіту, місце роботи, професійний досвід, навички та іншу інформацію, яка не повинна бути доступною стороннім особам.

Для автентифікації користувачів у системі застосовується технологія JSON Web Token (JWT). Після успішного введення облікових даних сервер перевіряє їх правильність і формує токен доступу. Отриманий токен передається клієнтському застосунку та використовується під час виконання наступних захищених запитів до backend. Як правило, токен передається в заголовок HTTP-запиту Authorization із використанням схеми Bearer. Завдяки цьому сервер може ідентифікувати користувача без необхідності зберігати окрему серверну сесію для кожного клієнта [22].

JWT містить набір службових даних, необхідних для визначення користувача та перевірки терміну дії токена. Цілісність токена забезпечується цифровим підписом, тому зміна його вмісту без знання секретного ключа призводить до помилки перевірки. Термін дії токена повинен бути обмеженим, що зменшує ризики у випадку його перехоплення. Після завершення терміну дії користувач має повторно пройти автентифікацію або отримати новий токен за допомогою передбаченого механізму оновлення.

На стороні backend перевірка автентифікаційних даних виконується через залежність `get_current_active_user`. Вона отримує токен із запиту, перевіряє його підпис, термін дії та коректність структури, після чого визначає поточного користувача. Додатково перевіряється, чи існує відповідний обліковий запис у базі даних і чи має він активний статус. Якщо токен відсутній, пошкоджений або недійсний, сервер відхиляє запит і повертає відповідний код помилки автентифікації.

Окрім загальної перевірки авторизації, у системі реалізовано контроль належності ресурсів. Під час спроби отримання, оброблення або видалення конкретного резюме backend перевіряє, чи пов'язаний відповідний запис із поточним користувачем. Таким чином, знання ідентифікатора чужого резюме не надає можливості отримати доступ до його вмісту. Аналогічний принцип застосовується до сформованих рекомендацій: користувач може

переглядати лише ті результати, які були створені на основі його власного резюме.

Такий підхід забезпечує розмежування доступу на рівні окремих об'єктів системи. Перевірка повинна виконуватися не лише в інтерфейсі frontend, а безпосередньо на стороні backend, оскільки клієнтську частину можна змінити або обійти шляхом надсилання запитів до API за допомогою сторонніх програм. Отже, серверна перевірка прав доступу є основним механізмом захисту персональних ресурсів.

Важливим елементом безпеки є захист паролів користувачів. Паролі не повинні зберігатися в базі даних у відкритому вигляді. Для їх зберігання доцільно використовувати криптографічне хешування із застосуванням спеціалізованих алгоритмів, призначених для роботи з паролями. Під час входу в систему хеш введеного пароля порівнюється зі значенням, збереженим у базі даних. Такий механізм зменшує ризик розкриття паролів у разі несанкціонованого доступу до бази даних [22].

Окрему увагу приділено перевірці файлів, які завантажуються користувачами. Система приймає резюме у форматах PDF, DOCX і TXT та обмежує максимальний розмір файлу. Перевірка типу й розміру файлу дозволяє зменшити ймовірність завантаження некоректних або потенційно небезпечних даних. Назва файлу, його розширення та фактичний тип вмісту повинні перевірятися на стороні сервера, оскільки клієнтська перевірка не може вважатися достатньою. Після отримання файлу система вилучає текстовий вміст, необхідний для подальшого аналізу та зіставлення з описами вакансій.

Розгортання системи для локального середовища реалізовано за допомогою Docker Compose. Конфігурація містить три основні сервіси: PostgreSQL, FastAPI backend і Next.js frontend. Кожний компонент запускається в окремому контейнері, що забезпечує ізоляцію середовищ виконання та спрощує налаштування залежностей. Усі сервіси

об'єднуються у внутрішню мережу Docker, завдяки чому вони можуть взаємодіяти між собою за визначеними іменами сервісів [18; 19; 20].

Сервіс PostgreSQL використовується для зберігання облікових записів користувачів, інформації про завантажені резюме, вакансії та сформовані рекомендації. Для бази даних налаштовано постійний Docker volume, який забезпечує збереження інформації після перезапуску або повторного створення контейнера. Без використання постійного сховища дані могли б бути втрачені після видалення контейнера.

FastAPI backend відповідає за реалізацію API, автентифікацію користувачів, оброблення резюме, взаємодію з базою даних і виконання алгоритмів формування рекомендацій. У конфігурації Docker Compose backend залежить від готовності сервісу PostgreSQL. Для цього може застосовуватися механізм перевірки стану контейнера, який запобігає запуску операцій із базою даних до моменту, коли PostgreSQL буде готовий приймати підключення.

Next.js frontend забезпечує користувацький інтерфейс системи. Через нього користувач може зареєструватися, увійти до системи, завантажити резюме, переглянути список вакансій і отримати персоналізовані рекомендації. Frontend взаємодіє з backend через HTTP API та залежить від доступності серверної частини. Така контейнерна схема дозволяє розгорнути всі компоненти системи однією командою, забезпечити однакове середовище на різних комп'ютерах і зменшити кількість помилок, пов'язаних із відмінностями версій програмного забезпечення.

Використання Docker Compose також спрощує тестування та подальше розгортання системи. Конфігураційні параметри, такі як адреса бази даних, секретний ключ JWT, термін дії токена й адреса backend, можуть передаватися через змінні середовища. Це дає змогу змінювати параметри системи без редагування програмного коду. Водночас конфіденційні значення не повинні зберігатися безпосередньо у відкритому репозиторії.

Для промислового середовища необхідно застосувати додаткові заходи захисту. Насамперед слід використовувати механізми керування секретами замість відкритого зберігання паролів, ключів і токенів у файлі `docker-compose.yml`. Секретний ключ JWT має бути достатньо складним, випадковим і недоступним для сторонніх осіб. Для різних середовищ розгортання, наприклад локального, тестового та промислового, доцільно використовувати різні секрети й облікові дані.

Також необхідно обмежити політику Cross-Origin Resource Sharing. У локальному середовищі може бути дозволена взаємодія з тестовою адресою `frontend`, однак у промисловому середовищі список дозволених джерел повинен містити лише офіційні домени системи. Використання надто широких налаштувань CORS може створити умови для надсилання небажаних запитів із зовнішніх вебресурсів.

Передавання даних між клієнтом і сервером повинно здійснюватися через захищений протокол HTTPS. Шифрування мережевого з'єднання захищає облікові дані, JWT-токени, тексти резюме та результати рекомендацій від перехоплення під час передавання. Для організації HTTPS може використовуватися зворотний проксі-сервер, який приймає зовнішні запити, виконує перевірку TLS-сертифіката та передає запити до контейнера `backend`.

Оскільки система працює з персональними даними, необхідно визначити політику їх зберігання. Доцільно зберігати лише ті дані, які необхідні для виконання функцій рекомендаційної системи, та не допускати необґрунтованого накопичення інформації. Користувач повинен мати можливість видалити завантажене резюме, сформовані на його основі рекомендації та, за потреби, власний обліковий запис. Видалення має охоплювати не лише основні таблиці бази даних, а й пов'язані файли, кешовані результати та резервні копії відповідно до встановленої політики зберігання.

Для контролю роботи системи необхідно реалізувати журналювання подій і помилок. У журналах можуть фіксуватися факти входу до системи, невдалі спроби автентифікації, завантаження та видалення резюме, формування рекомендацій і помилки оброблення файлів. Водночас журнали не повинні містити відкриті паролі, повні JWT-токени або повний текст резюме. Для захисту від автоматизованого підбору паролів доцільно обмежувати кількість невдалих спроб входу та застосовувати тимчасове блокування або затримку повторних запитів.

Додатковими заходами підвищення надійності є регулярне створення резервних копій бази даних, перевірка можливості їх відновлення, своєчасне оновлення залежностей і контейнерних образів, а також сканування компонентів на наявність відомих вразливостей. Доступ до бази даних не повинен бути відкритим безпосередньо для зовнішньої мережі. Взаємодія з нею має здійснюватися лише через backend або через обмежений адміністративний канал.

Отже, реалізована архітектура забезпечує базовий рівень захисту за рахунок JWT-автентифікації, перевірки активності користувача, контролю належності ресурсів і контейнерної ізоляції компонентів. Запропоновані додаткові заходи дозволять підвищити рівень безпеки системи під час її промислового розгортання та забезпечити належний захист персональної інформації користувачів.

4 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Середовище реалізації

Програмну систему реалізовано як monorepo, що містить папки backend і frontend, а також docker-compose.yml для запуску інфраструктури. Backend написано мовою Python з використанням FastAPI, SQLAlchemy, Pydantic, Alembic, PyJWT, pdfplumber, python-docx, scikit-learn і sentence-transformers. Frontend написано на TypeScript з використанням Next.js і React.

Вибір Python для backend зумовлений наявністю широкої екосистеми бібліотек для обробки тексту, машинного навчання та веб API. FastAPI забезпечує швидке створення типізованих endpoints, автоматичну документацію OpenAPI і зручну інтеграцію з Pydantic. PostgreSQL обрано як надійну реляційну базу даних, що добре підходить для збереження структурованих і напівструктурованих даних.

Frontend на Next.js дозволяє швидко створити сторінки вебзастосунку, організувати маршрути та компоненти. Використання TypeScript підвищує надійність клієнтського коду, оскільки типи API-відповідей можна явно описувати та контролювати.

4.2 Реалізація завантаження та аналізу резюме

Сценарій завантаження резюме починається з вибору файлу користувачем. Frontend відправляє файл на endpoint /api/v1/resumes/upload у форматі multipart/form-data. Backend перевіряє наявність імені файлу, розширення та розмір. Дозволено PDF, DOCX і TXT, що покриває найпоширеніші формати резюме [16], [17].

Після перевірки файл зберігається у файловому сховищі з унікальною назвою. Далі викликається модуль витягування тексту. Для PDF

застосовується `pdfplumber`, для DOCX – `python-docx`, для TXT – пряме читання тексту. Витягнутий текст зберігається у полі `extracted_text` моделі `Resume`.

Якщо під час обробки виникає помилка, система видаляє файл і повертає користувачу повідомлення про неможливість обробки. Такий механізм запобігає появі записів у базі даних без коректного тексту резюме.

У подальшому `extracted_text` передається в `RecommendationService`. Це дозволяє відокремити етап завантаження документа від етапу генерації рекомендацій. Користувач може завантажити резюме один раз, а потім запускати різні стратегії зіставлення.

4.3 Реалізація генерації рекомендацій

Генерація рекомендацій запускається на сторінці рекомендацій натисканням кнопки `Generate Recommendations`. Користувач може обрати одну зі стратегій: `keyword`, `TF-IDF` або `embeddings`. Запит надходить на backend endpoint `/recommendations/generate` разом із параметрами `strategy` і `top_k` [6].

`RecommendationService` отримує останнє резюме користувача і список активних вакансій. Далі формується список словників, де кожна вакансія містить `id`, `title`, `company`, `description` і `requirements`. Цей список передається в `RecommendationEngine`. Engine через `factory` обирає відповідну стратегію і викликає метод `match`.

Кожна стратегія повертає список `MatchScore`, відсортований за `overall_score`. Після цього старі рекомендації користувача видаляються, а нові результати зберігаються в таблиці `recommendations`. Таке рішення гарантує, що користувач бачить актуальний набір рекомендацій для останнього резюме та вибраної стратегії.

Збереження strategy у кожному записі дозволяє відстежувати, яким методом було отримано результат. Це корисно для аналізу, порівняння стратегій і подальшого розвитку системи.

4.4 Інтерфейс користувача

Інтерфейс користувача реалізовано у мінімалістичному стилі, що відповідає призначенню системи: швидко завантажити резюме та отримати зрозумілі рекомендації. У верхній частині сторінок розташовано навігаційну панель з назвою ML Resume Parser і посиланнями на основні розділи.

На головній сторінці користувач бачить заголовок AI-Powered Job Recommendations, короткий опис системи та дві основні дії: Upload Resume і Browse Vacancies. Нижче розміщено три картки, які пояснюють процес роботи: завантаження резюме, перегляд вакансій і отримання рекомендацій. Загальний вигляд головної сторінки наведено на рисунку 4.1, а сторінку завантаження резюме – на рисунку 4.2.

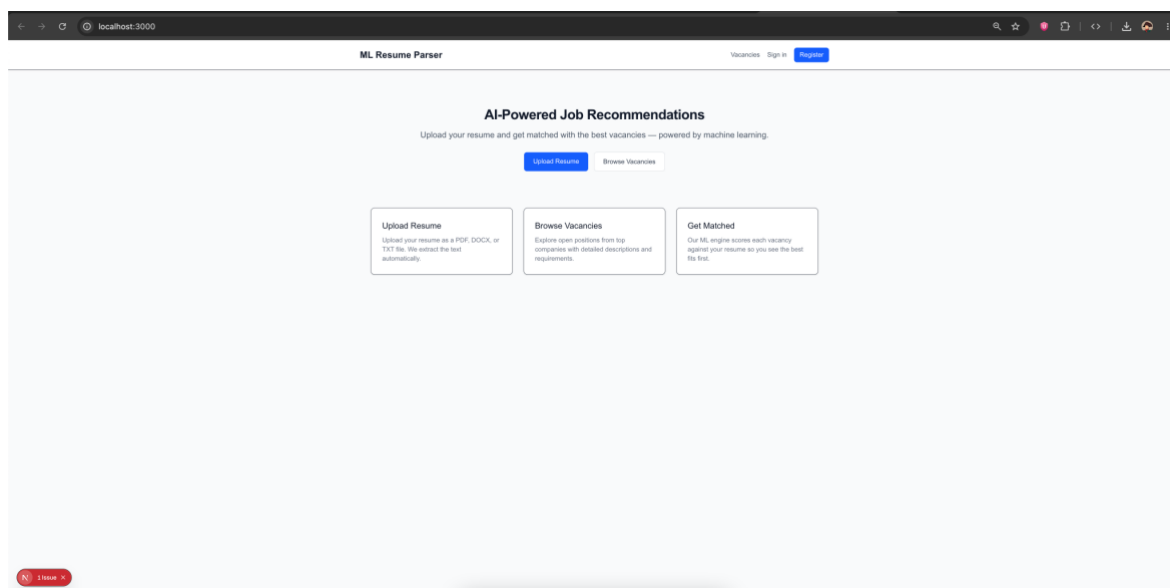


Рисунок 4.1 – Головна сторінка вебзастосунку ML Resume Parser

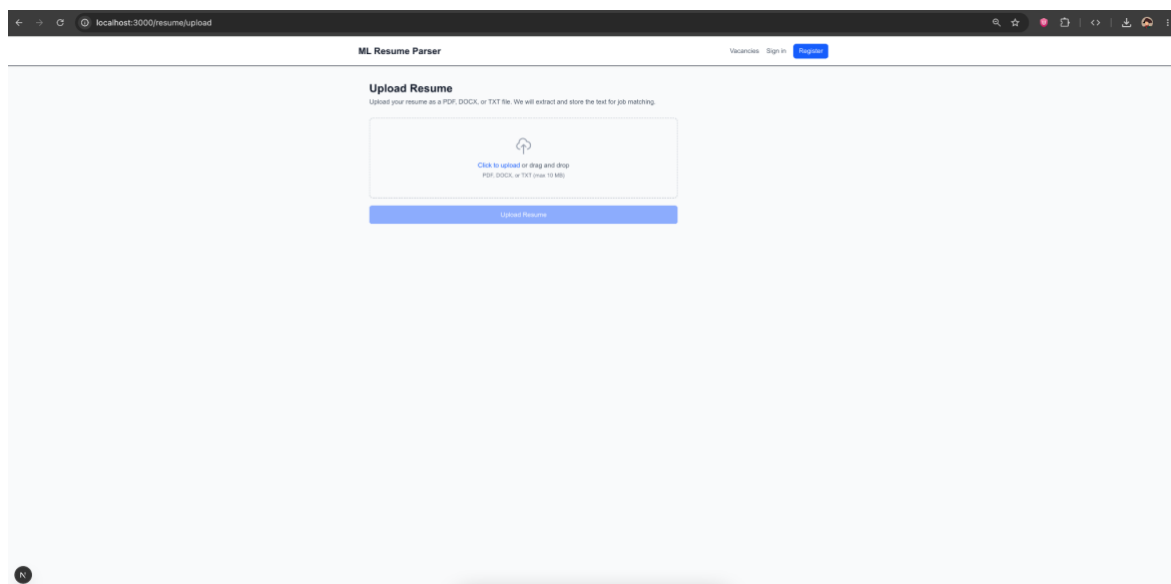


Рисунок 4.2 – Сторінка завантаження резюме

Сторінка завантаження містить галузь drag-and-drop, інформацію про допустимі формати PDF, DOCX і TXT, а також кнопку Upload Resume. Стан кнопки показує, що без обраного файлу дія недоступна, що є простим механізмом запобігання помилковому запиту. Сторінку входу користувача показано на рисунку 4.3.

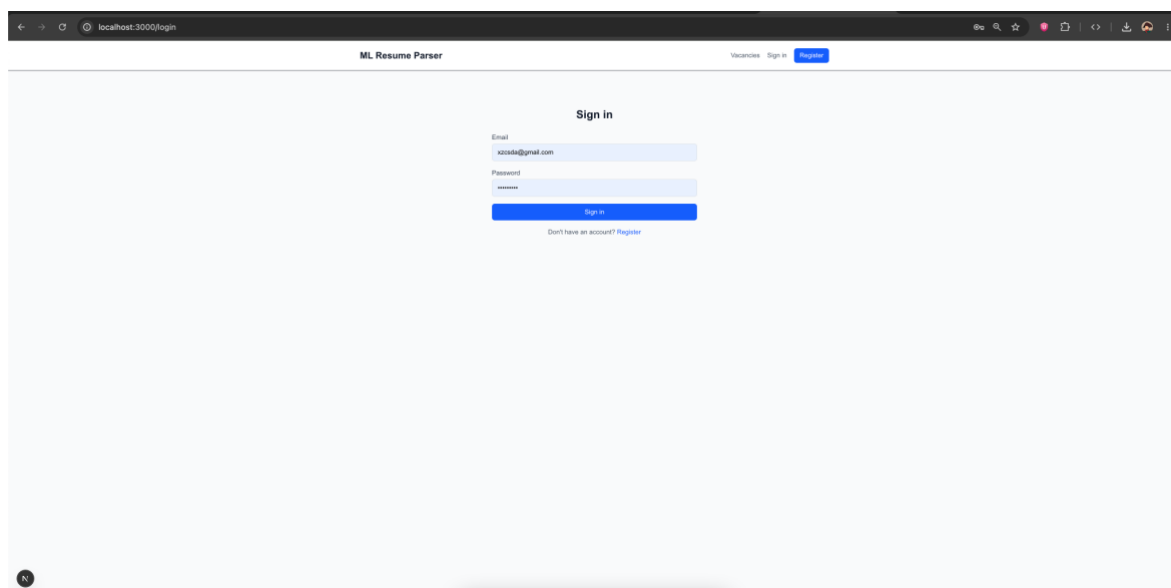


Рисунок 4.3 – Сторінка входу користувача

Сторінку рекомендацій із результатами matching наведено на рисунку 4.4.

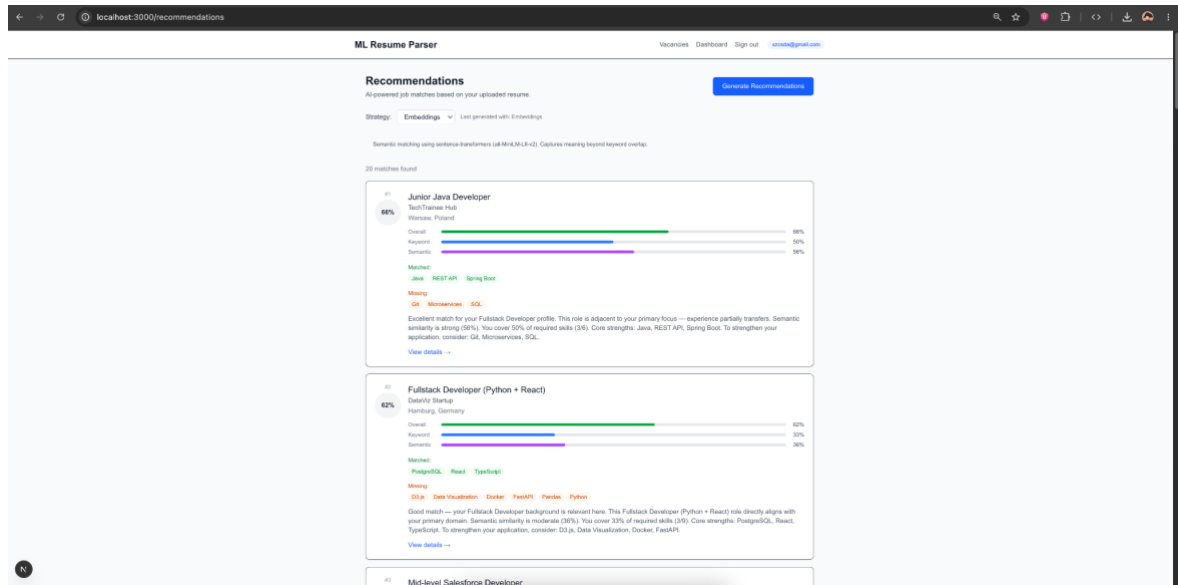


Рисунок 4.4 – Сторінка рекомендацій вакансій із результатами matching

На сторінці рекомендацій показано 20 знайдених збігів. Для першої вакансії Junior Java Developer відображено загальний бал 66%, keyword score 50% і semantic score 56%. Для другої вакансії Fullstack Developer (Python + React) загальний бал становить 62%, а система окремо показує matched skills і missing skills. Це підтверджує, що реалізований інтерфейс надає не тільки ранжування, а й пояснення результатів.

4.5 Функціональне тестування системи

Тестування виконувалося за основними користувацькими сценаріями. Метою тестування було переконатися, що система коректно обробляє реєстрацію, вхід, завантаження резюме, генерацію рекомендацій, вибір стратегії та відображення результатів. Окремо перевірялися негативні сценарії: неправильний формат файлу, відсутність резюме, відсутність активних вакансій і запит без авторизації [1], [2].

Функціональне тестування для дипломного прототипу має прикладний характер. Воно не вимірює точність моделі на великому розміченому датасеті, оскільки такий датасет не був наданий. Натомість перевіряється коректність реалізації пайплайна та відповідність результатів очікуваній логіці: вакансії з релевантними навичками мають отримувати вищі бали, а відсутні навички мають відображатися у missing skills.

4.6 Аналіз отриманих результатів

Результати роботи системи демонструють, що розроблений пайплайн виконує повний цикл рекомендації: від завантаження резюме до формування ранжованого списку вакансій. На сторінці рекомендацій відображено 20 знайдених збігів, що свідчить про коректне отримання активних вакансій і роботу top-K ранжування [3], [4], [6].

Перший результат – Junior Java Developer – має загальний бал 66%. Система показує matched skills Java, REST API і Spring Boot, а серед missing skills – Git, Microservices і SQL. Такий результат є логічним для резюме Fullstack Developer, яке має частину backend-компетенцій, але не покриває всі вимоги Java-вакансії.

Другий результат – Fullstack Developer (Python + React) – має загальний бал 62%. Matched skills включають PostgreSQL, React і TypeScript, а missing skills включають D3.js, Data Visualization, Docker, FastAPI, Pandas і Python. Це показує, що система вміє знаходити часткові збіги та надавати користувачу список компетенцій, які варто додати або розвинути.

Важливо, що система не обмежується одним числом. Користувач бачить компоненти оцінки, а також пояснення природною мовою. Це дозволяє інтерпретувати результат і приймати рішення: податися на вакансію, доопрацювати резюме, пройти навчання або обрати іншу позицію.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано задачу розробки рекомендаційної системи підбору вакансій на основі аналізу тексту резюме та описів вакансій. Було досліджено предметну галузь автоматизованого job matching, визначено особливості текстових даних резюме і вакансій, проаналізовано методи рекомендаційних систем, інформаційного пошуку та обробки природної мови.

Розроблено алгоритмічне забезпечення системи, яке поєднує словникове виділення навичок, TF-IDF, косинусну подібність, sentence embeddings і role-aware scoring. Такий підхід дозволяє враховувати як точні збіги технологій, так і семантичну близькість описів досвіду та вимог вакансій. Для підвищення практичної цінності результатів реалізовано пояснюваність рекомендацій через matched skills, missing skills і текстове пояснення.

Спроектовано архітектуру full-stack вебзастосунку з розділенням на frontend, backend, базу даних і AI layer. Backend реалізовано на FastAPI з використанням SQLAlchemy, Pydantic і PostgreSQL. Frontend реалізовано на Next.js і React. Для локального запуску підготовлено Docker Compose конфігурацію з сервісами PostgreSQL, backend і frontend.

Практичним результатом є вебзастосунок ML Resume Parser, який підтримує реєстрацію та вхід користувача, завантаження резюме у форматах PDF, DOCX і TXT, витягування тексту, перегляд вакансій, генерацію рекомендацій за різними стратегіями та відображення детального пояснення результатів. На демонстраційному сценарії система сформувала 20 рекомендацій, а найкращі результати отримали вакансії, що мають значні збіги з профілем кандидата.

Функціональне тестування підтвердило працездатність основних сценаріїв системи. Водночас встановлено напрями подальшого розвитку: розширення словника навичок, використання спеціалізованих таксономій

на зразок ESCO, навчання моделей ранжування на розмічених парах резюме-вакансія, підтримка багатомовних резюме, інтеграція з зовнішніми job boards, додавання метрик якості рекомендацій і покращення захисту персональних даних [9], [10].

Отже, мету роботи досягнуто, а поставлені задачі виконано повністю. Розроблена система може використовуватися як навчальний прототип інтелектуального HR-сервісу та як основа для подальших досліджень у сфері рекомендаційних систем, NLP і автоматизованого підбору вакансій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ: ДП «УкрНДНЦ», 2016. 31 с.
2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016. 20 с.
3. Ricci F., Rokach L., Shapira B. Recommender systems handbook. 3rd ed. New York: Springer, 2022. 1050 p.
4. Manning C. D., Raghavan P., Schütze H. Introduction to information retrieval. Cambridge: Cambridge University Press, 2008. 496 p.
5. Salton G., Buckley C. Term-weighting approaches in automatic text retrieval. *Information processing & management*. 1988. Vol. 24, no. 5. P. 513–523. URL: [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0) (date of access: 08.07.2026).
6. Reimers N., Gurevych I. Sentence-BERT: sentence embeddings using siamese bert-networks. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: <https://doi.org/10.18653/v1/d19-1410> (date of access: 08.07.2026).
7. BERT: pre-training of deep bidirectional transformers for language understanding / J. Devlin et al. *Proceedings of the 2019 conference of the north, Minneapolis, Minnesota*. Stroudsburg, PA, USA, 2019. URL: <https://doi.org/10.18653/v1/n19-1423> (date of access: 25.06.2026).
8. Lavi D., Medentsiy V., Graus D. ConSultantBERT: fine-tuned siamese sentence-bert for matching jobs and job seekers. *arXiv*. 2021. URL: <https://doi.org/10.48550/arXiv.2109.06501> (date of access: 08.07.2026).

9. CareerBERT: Matching resumes to ESCO jobs in a shared embedding space for generic job recommendations / J. Rosenberger et al. *Expert systems with applications*. 2025. Vol. 275. P. 127043. URL: <https://doi.org/10.1016/j.eswa.2025.127043> (date of access: 08.07.2026).
10. Skill extraction from job postings using weak supervision / M. Zhang et al. *arXiv*. 2022. URL: <https://doi.org/10.48550/arXiv.2209.08071> (date of access: 08.07.2026).
11. Kavas H., Serra-Vidal M., Wanner L. Multilingual Skill extraction for job vacancy – job seeker matching in knowledge graphs. In *Proceedings of the workshop on generative ai and knowledge graphs (GenAIK)*. 2025. P. 146–155.
12. TfidfVectorizer. *scikit-learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (date of access: 08.07.2026).
13. Cosine_similarity. *scikit-learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html (date of access: 08.07.2026).
14. Pretrained models. *SentenceTransformers Documentation*. URL: https://www.sbert.net/docs/sentence_transformer/pretrained_models.html (date of access: 08.07.2026).
15. Sentence-transformers/all-MiniLM-L6-v2. *Hugging Face*. URL: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2> (date of access: 08.07.2026).
16. FastAPI documentation. *FastAPI*. URL: <https://fastapi.tiangolo.com/> (date of access: 08.07.2026).
17. SQLAlchemy 2.0 documentation. *SQLAlchemy Documentation*. URL: <https://docs.sqlalchemy.org/> (date of access: 08.07.2026).
18. PostgreSQL global development group. *PostgreSQL 16 Documentation*. URL: <https://www.postgresql.org/docs/16/> (date of access: 04.07.2026).

19. Docker compose. *Docker Documentation*.

URL: <https://docs.docker.com/compose/> (date of access: 08.07.2026).

20. Vercel. Next.js docs: app router | next.js. *Next.js by Vercel*.

URL: <https://nextjs.org/docs/app> (date of access: 08.07.2026).

21. Fielding R. T. Architectural styles and the design of network-based software architectures: doctoral dissertation. Irvine: University of California, 2000.