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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система оптимізації процесу рекрутингу з використанням методів машинного навчання»

на здобуття освітнього ступеня бакалавра
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

_____ Марія ДУДНИК
(підпис)

Виконала: здобувачка вищої освіти групи ПД-42

Марія ДУДНИК

Керівник: Тимур ДОВЖЕНКО
канд.техн.наук

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дудник Марії Євгенівни

1. Тема кваліфікаційної роботи: «Система оптимізації процесу рекрутингу з використанням методів машинного навчання»
керівник кваліфікаційної роботи кандидат технічних наук Тимур ДОВЖЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи обробки природної мови (NLP) та машинного навчання, технічна документація до бібліотек scikit-learn, Flask, SQLite, наукові дослідження в галузі рекрутингової аналітики, метрики оцінювання моделей класифікації та архітектурні патерни проєктування інформаційних систем.

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

1. Огляд та аналіз існуючих методів, технологій та програмних рішень обробки природної мови і машинного навчання в контексті задачі оптимізації процесу рекрутингу.

2. Проєктування інтелектуальної системи для автоматизованого підбору, оцінювання та ранжування кандидатів на вакансії.
3. Програмна реалізація, структура бази даних та опис функціонування модулів системи оптимізації процесу рекрутингу з використанням методів машинного навчання.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма прецедентів.
5. Діаграма компонентів.
6. Блок-схема алгоритму автоматизованого оцінювання кандидата.
7. Блок-схема алгоритму ранжування кандидатів за вакансією.
8. Блок-схема алгоритму прийняття кадрового рішення.
9. Екранні форми.
10. Апробація результатів дослідження

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Огляд існуючих алгоритмів обробки тексту та методів і технологій оцінювання релевантності резюме вимогам вакансій	08.03 – 18.03.2026	
4	Проектування застосунку для автоматизованої оптимізації процесу рекрутингу з використанням методів машинного навчання	19.03 – 31.03.2026	
5	Програмна реалізація застосунку	01.04 – 24.04.2026	
6	Тестування застосунку	25.04 – 30.04.2026	
7	Оформлення роботи: вступ, висновки, реферат	01.05 – 09.05.2026	
8	Розробка демонстраційних матеріалів	10.05 – 18.05.2026	
9	Попередній захист роботи	12.05-22.05.2026	

Здобувачка вищої освіти _____ Марія ДУДНИК
(підпис)

Керівник
кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 67 стор., 11 табл., 34 рис., 20 джерел.

Мета роботи – покращення ефективності підбору персоналу шляхом впровадження інтелектуальної системи оптимізації процесу рекрутингу.

Об'єкт дослідження – розроблення інтелектуальної системи оптимізації процесу рекрутингу.

Предмет дослідження – методи та алгоритми машинного навчання, моделі оброблення резюме й вакансій, а також програмні засоби оптимізації рекрутингового процесу на основі аналітики даних.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для обробки природної мови та машинного навчання для оптимізації процесу рекрутингу. Проаналізовано інструменти автоматизації: LinkedIn Talent Insights, Workday Recruiting, Manatal. Розроблено та протестовано застосунок. Використано мову програмування Python, вебфреймворк Flask, базу даних SQLite, а також бібліотеки scikit-learn та pandas для обробки даних та реалізації ML-модуля. Програмно реалізовані ключові функціональні можливості, зокрема: автентифікація користувачів, створення вакансій та завантаження резюме кандидатів, автоматизований розрахунок відповідності Match Score на основі семантичного аналізу навичок та досвіду.

Сферою використання застосунку є автоматизація процесів підбору персоналу в компаніях для зниження часових витрат і мінімізації упередженості при відборі кандидатів.

КЛЮЧОВІ СЛОВА: ОПТИМІЗАЦІЯ РЕКРУТИНГУ, МАШИННЕ НАВЧАННЯ, NLP, MATCH SCORE, РАНЖУВАННЯ КАНДИДАТІВ, FLASK, SQLITE, PYTHON

ЗМІСТ

ВСТУП

Актуальність теми полягає у необхідності підвищення ефективності процесів підбору персоналу в умовах зростання обсягів даних про кандидатів, динамічних вимог ринку праці та обмеженого часу на ухвалення кадрових рішень. Традиційні підходи до рекрутингу, що ґрунтуються на ручному аналізі резюме та суб'єктивному оцінюванні компетентностей, не забезпечують достатньої швидкодії, об'єктивності й масштабованості. Сучасні дослідження доводять, що використання машинного навчання для структурованої обробки даних дозволяє суттєво підвищити точність оцінювання кандидатів і зменшити вплив людського фактора. У таких умовах виникає потреба у створенні інтелектуальної системи, здатної автоматизувати критичні етапи рекрутингового процесу та забезпечити обґрунтованість управлінських рішень.

Метою роботи є покращення ефективності підбору персоналу шляхом впровадження інтелектуальної системи оптимізації процесу рекрутингу.

Досягнення поставленої мети потребує вирішення таких завдань:

1. Провести аналіз предметної області, дослідити процеси підбору кандидатів та виявити ключові проблеми автоматизації обробки резюме.
2. Провести порівняльний аналіз існуючих програмних рішень на ринку, що дозволить визначити їхні недоліки та обґрунтувати доцільність створення нової системи.
3. Визначити та сформулювати функціональні та нефункціональні вимоги, що дозволить встановити технічні характеристики та критерії успішності готового продукту.
4. Провести аналіз та обґрунтувати вибір технологічного стеку та моделей машинного навчання, що забезпечить високу ефективність та швидкість обробки даних.

5. Спроєктувати та реалізувати архітектуру застосунку, структуру бази даних, інтелектуальні модулі оцінювання кандидатів та інтерфейс користувача з розмежуванням прав доступу.
6. Провести тестування розробленого програмного забезпечення для перевірки коректності роботи інтерфейсних модулів, точності алгоритмів.

Об'єктом дослідження є процес рекрутингу як цілісна інформаційно-керуюча система.

Предметом дослідження є методи та алгоритми машинного навчання, моделі оброблення резюме й вакансій, а також програмні засоби оптимізації рекрутингового процесу на основі аналітики даних.

Методи дослідження включають алгоритми машинного навчання, методи оброблення природної мови, статистичні методи аналізу даних, методи моделювання та експериментальної оцінки продуктивності.

Наукова новизна роботи полягає у розробленні комплексної методики оцінювання кандидатів, що об'єднує структуровані метрики HR-процесів з моделями аналізу текстових даних, а також у створенні адаптивної архітектури системи, здатної підлаштовуватися під різні професійні домени, мінімізуючи суб'єктивність рекрутингових рішень. Запропонований підхід забезпечує підвищення точності оцінювання, скорочення часу обробки кандидатських даних та покращення якості управлінських рішень у сфері підбору персоналу.

1 ПОТОЧНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМ ОПТИМІЗАЦІЇ РЕКРУТИНГУ ЗАСТОСУВАННЯМ МАШИННОГО НАВЧАННЯ

1.1 Огляд існуючих методів оптимізації рекрутингу

Сучасні підходи до автоматизації рекрутингу демонструють послідовну еволюцію від ручних експертних процедур до інтелектуальних систем, що здатні самонавчатися та адаптуватися до змін ринку праці. Початковий етап розвитку характеризувався статистичними та експертними методами, у межах яких використовувалися правила пошуку ключових слів, скорингові карти та прості лінійні моделі, такі як логістична регресія й кластеризація. Такі рішення відзначалися високою інтерпретованістю, однак залежали від ручної підтримки й не враховували контекстні фактори. Незважаючи на обмеження, ці системи заклали основу для подальшого переходу до алгоритмічних формалізацій процесу підбору персоналу [1].

Наступний рівень розвитку становлять алгоритмічні системи підтримки прийняття рішень (СППР), які впроваджують методи багатокритеріальної оптимізації (MCDM), зокрема аналітичний ієрархічний процес (AHP) і метод TOPSIS. Вони дозволяють враховувати вагові коефіцієнти для різних критеріїв кандидатів, реалізуючи прозору й відтворювану логіку прийняття рішень. У сучасних Applicant Tracking Systems (ATS) такі моделі поєднуються з формальними бізнес-правилами, що визначають послідовність етапів найму, і процедурами ILP/LP для оптимізації витрат часу рекрутера. Основними перевагами алгоритмічних СППР є контрольованість, масштабованість і можливість формалізованого аудиту, тоді як недоліком залишається низька адаптивність до нових типів даних без інтеграції навчальних моделей [2].

На найвищому рівні автоматизації перебувають методи машинного навчання та гібридні інтелектуальні системи, які поєднують класичні алгоритми (SVM,

Random Forest, Gradient Boosting) з моделями глибокого навчання, зокрема BERT і Sentence-BERT для семантичного зіставлення резюме та вакансій. У межах мультимодальних систем використовуються архітектури GPT-4V, що забезпечують обробку PDF-резюме, відео-інтерв'ю й соціальних профілів через поєднання NLP, CV та ML-модулів. Додатково впроваджуються інструменти пояснюваного штучного інтелекту (XAI) - SHAP, LIME - а також метрики оцінювання BLEU, ROUGE та F1, які дають змогу забезпечити прозорість, стабільність і контроль якості результатів. Еволюційний тренд галузі полягає у переході від жорстко формалізованих алгоритмів до контекстно-семантичних моделей з інтегрованим механізмом самонавчання [3].

На рисунку 1.1 подано класифікацію сучасних підходів до оптимізації рекрутингу, що охоплює три рівні автоматизації: статистичні та експертні методи, алгоритмічні СППР і гібридні інтелектуальні системи на основі машинного навчання.

Рис. 1.1 Класифікація сучасних підходів до автоматизації рекрутингу

У класичних системах автоматизації підбору персоналу ключову роль відіграють методи попередньої фільтрації кандидатів, які ґрунтуються на заздалегідь визначених правилах і ключових словах. Такі системи реалізують базову логіку пошуку за лексичними шаблонами, регулярними виразами або скоринговими картами. Вхідні дані - резюме у форматах PDF чи DOCX - надходять із черги ATS або через REST API, після чого проходять етапи парсингу, токенізації

й нормалізації [21]. Отриманий набір термінів співставляється з контрольними списками навичок і посадових назв, формуючи попередній скоринг. На основі порогових значень (thresholds) система визначає, чи підлягає резюме подальшій аналітиці, чи відхиляється. Такий rule-based підхід забезпечує передбачуваність рішень і високу швидкість обробки, проте він не враховує контекстні зв’язки та не здатний до самонавчання, що обмежує масштабованість [4].

На рисинку 1.2 наведено архітектурну схему rule-based фільтрації, у якій потоки даних з ATS і REST API надходять до модуля парсингу резюме, потім - до модуля нормалізації (токенізація, TF-IDF) і перевірки ключових слів або регулярних виразів. Після обчислення скорингової оцінки система порівнює результат із пороговим значенням у блоці Gate. Резюме, що пройшли перевірку, передаються на подальшу обробку в середовище машинного навчання (PASS -> Staging для ML), тоді як невідповідні автоматично відхиляються (FAIL -> Reject Queue).

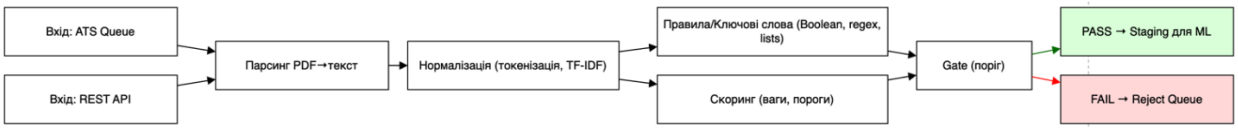


Рис. 1.2 Архітектурна схема rule-based фільтрації резюме у класичних HR-системах

Оцінювання ефективності таких систем здійснюється за допомогою базових метрик класифікації, наведених у таблиці 1.1. Високі значення precision та recall свідчать про збалансовану роботу системи, тоді як F1-scoreвикористовується для визначення загальної узгодженості між цими показниками.

Таблиця 1.1

Основні метрики ефективності rule-based і статистичних методів фільтрації

Метрика	Формула	Інтерпретація	Типова межа прийнятності
Precision	$\frac{TP}{TP + FP}$. (1.1)	Частка коректно відібраних резюме серед усіх відібраних	≥ 0.8
Recall	$\frac{TP}{TP + FN}$. (1.2)	Частка правильно відібраних резюме серед усіх релевантних	≥ 0.7
F1-score	$\frac{2 \cdot (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$. (1.3)	Узагальнений баланс точності та повноти	≥ 0.75
Accuracy	$\frac{TP + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$. (1.4)	Загальна частка правильних рішень системи	≥ 0.85

Для зменшення обмежень rule-based фільтрів сучасні HR-платформи інтегрують регресійні, кластеризаційні та ранжувальні моделі, які використовують статистичні залежності між атрибутами кандидатів. Наприклад, логістична регресія дозволяє прогнозувати ймовірність відповідності вакансії, а алгоритми кластеризації (K-means, DBSCAN) застосовуються для виявлення груп подібних профілів. Хоча такі системи покращують якість відбору, їхня ефективність суттєво знижується зі зростанням обсягу даних і варіативності текстів, що зумовлює необхідність переходу до адаптивних методів машинного навчання та семантичного аналізу [5].

У межах сучасних систем рекрутингу методи машинного навчання (ML) є ключовим інструментом підвищення точності та швидкодії відбору кандидатів. Найпоширенішим є підхід supervised learning, у якому моделі навчаються на попередньо маркованих даних - резюме, що мають історію прийняття або

відхилення. Такі моделі застосовуються для класифікації резюме, прогнозування ймовірності успішності кандидата та визначення релевантності профілю до вакансії. На практиці використовуються алгоритми SVM, Random Forest, Gradient Boosting та нейронні мережі, що дозволяють автоматизувати процес оцінки компетенцій і досвіду [6].

На рисинку 1.3 наведено типовий supervised пайплайн для аналізу резюме, що включає етапи конвертації PDF у HTML-структуру (pdf2htmlEX), сегментації секцій (освіта, досвід, навички), векторизації тексту та подальшої класифікації за допомогою моделей scikit-learn або Weka [17]. Результатом є структуровані дані про кандидата, які можуть використовуватись для побудови моделей прогнозування успішності найму.

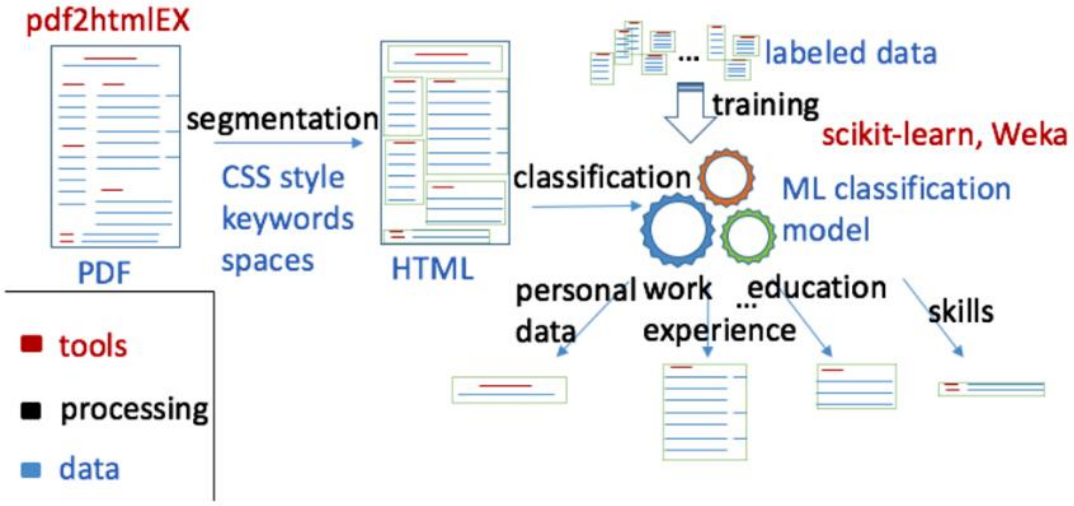


Рис. 1.3 Пайплайн класифікації резюме на основі машинного навчання [6]

На основі результатів класифікації система виконує ранжування кандидатів. Інтерфейс прикладу наведено на рисинку 1.4, де для кожного резюме відображається відсоток відповідності, перелік навичок і сертифікацій, що автоматично виявляються в тексті. Це дозволяє HR-аналітику переглядати лише найрелевантніших кандидатів, зменшуючи навантаження на первинний етап відбору.

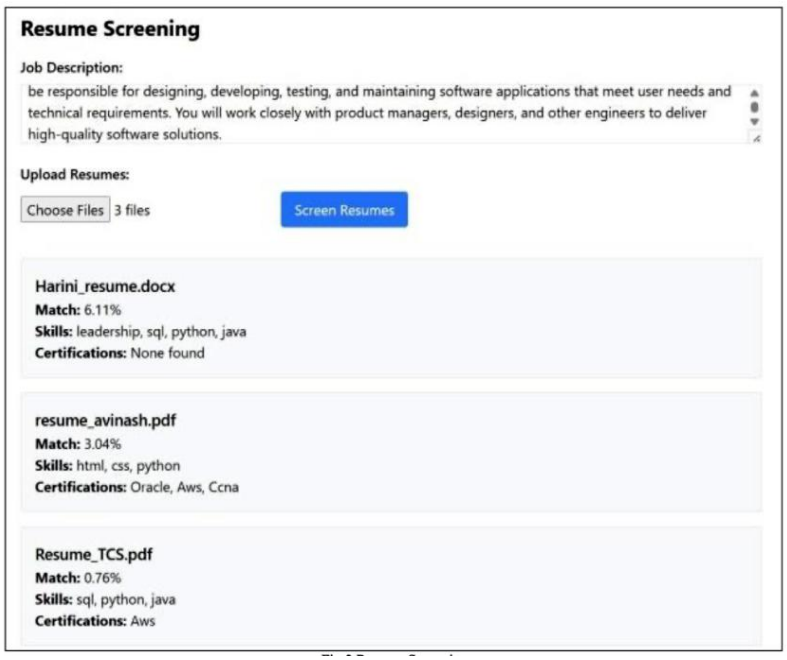


Рис. 1.4 Інтерфейс ранжування резюме у системі автоматизованого скринінгу [7]

У дослідженні Zimmermann et al. (2016) [7] наведено приклад інтеграції supervised та unsupervised методів. Автори запропонували архітектуру, у якій supervised-моделі здійснюють початкову класифікацію, а unsupervised-підходи (кластеризація на основі word embeddings + t-SNE) забезпечують сегментацію та виявлення латентних закономірностей між навичками, професійними групами та типами посад. Приклад візуалізації кластерів професій за векторними представленнями показано на рисинку 1.5.

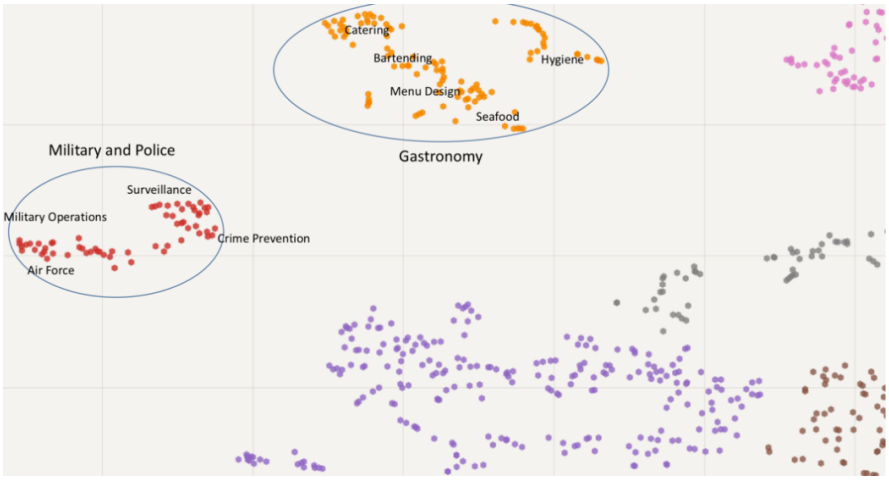


Рис. 1.5 Візуалізація кластеризації навичок та професійних груп на основі word embeddings [7]

У більш сучасних розробках (Kumar et al., 2020; IJISRT, 2025) [8] застосовується підхід, у якому векторизація тексту та косинусна подібність (cosine similarity) використовуються для оцінки семантичної відповідності резюме до опису вакансії. Це дозволяє враховувати контекстну схожість навіть при відсутності прямих збігів ключових слів. Ефективність таких моделей оцінюється за стандартними метриками precision, recall, F1-score і ROC-AUC, які характеризують збалансованість і стабільність класифікації.

На рисинку 1.6 подано приклад ROC-кривої для бінарної класифікації (оцінка «релевантний/нерелевантний»)

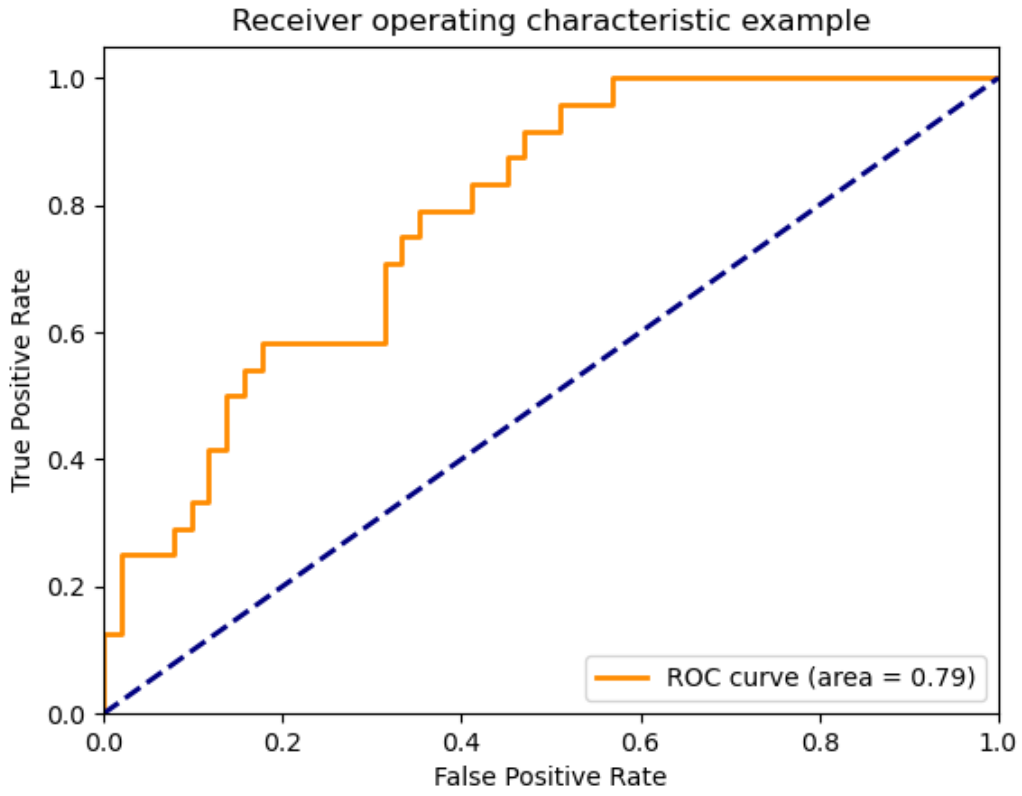


Рис. 1.6 ROC-крива для оцінки ефективності моделі класифікації резюме [9]

Тоді як рисунок 1.7 ілюструє багатокласовий варіант ROC-AUC для систем, що прогнозують рівень відповідності кандидата кільком категоріям вакансій.

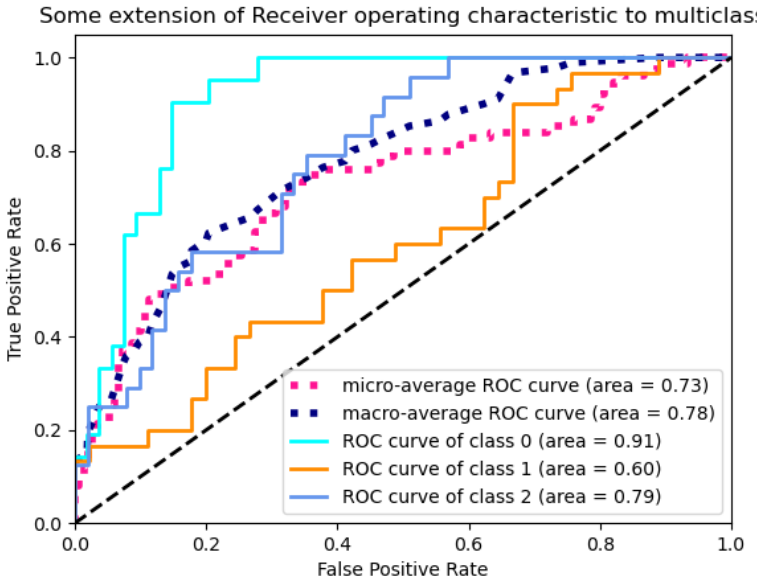


Рис. 1.7 Багатокласовий варіант ROC-AUC для моделей прогнозування відповідності кандидатів [9]

Методи машинного навчання забезпечують якісно новий рівень аналітики у процесах рекрутингу, дозволяючи здійснювати адаптивне навчання моделей на основі зворотного зв'язку від рекрутерів, автоматичне оновлення вагових коефіцієнтів і підвищення точності прогнозів за рахунок контекстно-семантичного аналізу текстових документів.

У сучасних інтелектуальних HR-системах ключовим напрямом розвитку є методи обробки природної мови (NLP), що забезпечують автоматизований аналіз неструктурованих текстових даних резюме (CV) та описів вакансій (JD). Ці методи дозволяють здійснювати витяг сутностей (імен, посад, навичок, кваліфікацій, освітніх установ), нормалізацію й семантичне зіставлення кандидатів із вимогами вакансій на основі векторних репрезентацій. Використання NLP у рекрутингу суттєво підвищує точність оцінювання релевантності, скорочує час прийняття рішень і зменшує вплив людського чинника [10].

На рисинку 1.8 подано загальну схему NLP-пайплайну рекрутингу, що охоплює повний цикл обробки текстів. Вхідними даними є резюме та описи вакансій, отримані через ATS-системи або REST-інтерфейси. Після попередньої

обробки (токенізація, лематизація, очищення) застосовуються методи POS-tagging та NER для виявлення лінгвістичних сутностей, які потім нормалізуються до канонічних форм відповідно до таксономій (наприклад, ESCO, O*NET) [19]. На цьому етапі формуються структуровані профілі кандидатів, де навички, посади та досвід представлені уніфіковано.

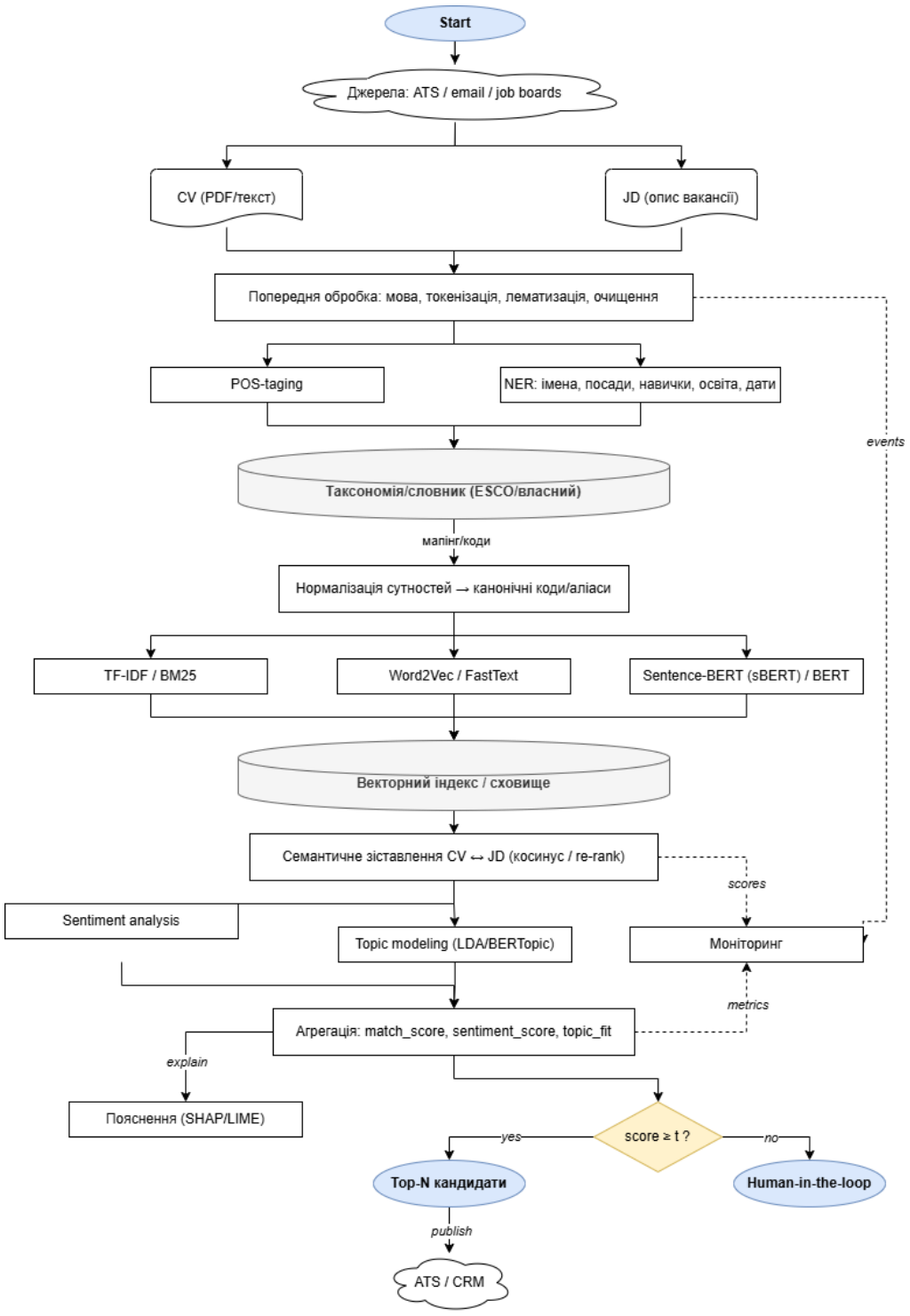


Рис. 1.8 NLP-пайплайн семантичного зіставлення резюме та вакансій у системах рекрутингу

Для моделювання семантичної близькості між кандидатами та вакансіями використовуються векторні моделі. Початкові методи TF-IDF і BM25 залишаються ефективними для статистичного аналізу термінів, однак сучасні системи поступово переходять до контекстно-залежних моделей типу Word2Vec, FastText, BERT та Sentence-BERT. Вони дозволяють обчислювати косинусну схожість між описом вакансії та резюме з урахуванням контексту значень слів і багатомовних варіантів [11]. Результати подібного зіставлення формують векторні індекси, на основі яких здійснюється re-ranking кандидатів за рівнем відповідності.

Додатковим компонентом сучасних систем є аналіз тональності (sentiment analysis), що використовується для виявлення комунікативних і поведінкових характеристик кандидатів у текстах супровідних листів, а також тематичне моделювання (topic modeling) на базі LDA або BERTopic, яке допомагає оцінити культурну відповідність претендента цінностям компанії. Отримані метрики (match_score, sentiment_score, topic_fit) агрегуються в інтегральний показник релевантності, який використовується для формування топ-N списку кандидатів.

Наявність Human-in-the-loop компонента (людського контролю) у таких системах дає змогу проводити перевірку результатів, виправлення помилкових класифікацій і адаптацію моделей на основі пояснюваних алгоритмів (SHAP, LIME). Це забезпечує поєднання автоматизованої оцінки та контрольованої корекції, що підвищує довіру до систем на ринку корпоративного рекрутингу [12].

Подальша еволюція систем автоматизованого рекрутингу пов’язана з переходом до мультимодальних і гібридних архітектур, що об’єднують методи обробки природної мови (NLP), комп’ютерного зору (CV) та машинного навчання (ML) для спільного аналізу різнорідних джерел даних - PDF-резюме, відео-інтерв’ю та соціальних профілів. Як показано на рисинку 1.9, така система поєднує компоненти збору, обробки та валідації інформації з різних модальностей у єдиному інтелектуальному контурі.

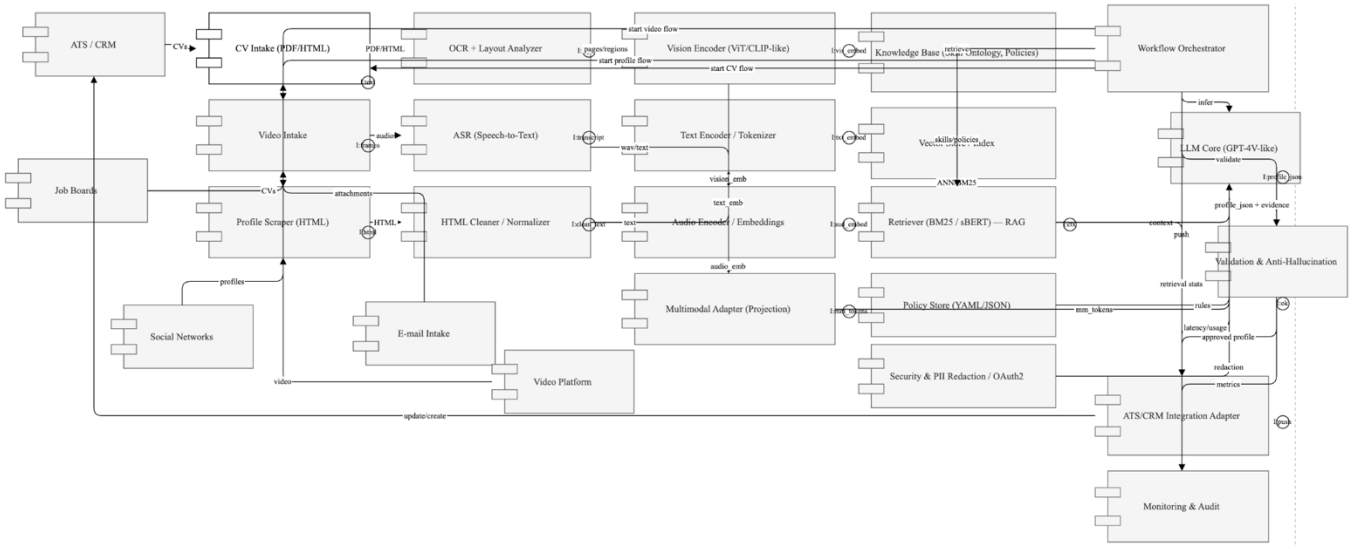


Рис. 1.9 Архітектура мультимодальної інтелектуальної системи рекрутингу з використанням GPT-4 Vision / CLIP та RAG-підходів

Вхідні потоки формуються з кількох каналів: ATS-систем, job boards, e-mail, відеоплатформ та соціальних мереж. Компонент CV Intake виконує аналіз PDF/HTML-резюме, де OCR + Layout Analyzer ідентифікує структуру документа (заголовки, секції, таблиці), а Vision Encoder на базі моделей ViT / CLIP формує візуально-текстові ембеддинги. Для відеоінтерв'ю застосовується ASR (Speech-to-Text) і подальше кодування аудіо-сигналів у текстові вектори. Соціальні профілі агрегуються через Profile Scraper і проходять очищення в HTML Normalizer.

Подальша інтеграція забезпечується Multimodal Adapter, що уніфікує векторні подання з різних джерел (текст, зображення, аудіо). Отримані репрезентації зберігаються у Vector Index (ANN / BM25) і використовуються Retriever-модулем, який реалізує механізм Retrieval-Augmented Generation (RAG). На етапі генерації профілів працює LLM Core (GPT-4V-like), який поєднує витягнуті ознаки з бази знань (Skill Ontology, Policies) для побудови структурованого опису кандидата.

Критично важливим елементом є блок Validation & Anti-Hallucination, який застосовує метрики BLEU, ROUGE та Levenshtein distance для верифікації результатів генерації, усунення "галюцинацій" великих мовних моделей та контролю достовірності згенерованих даних. Адаптивні правила політик у Policy

Store (YAML/JSON) забезпечують відповідність процесу стандартам GDPR і вимогам компанії, тоді як модуль Security & PII Redaction / OAuth2відповідає за захист персональних даних.

Інтеграційний адаптер ATS/CRM отримує лише валідаційно підтверджені профілі разом із поясненнями моделі. Таким чином формується замкнений цикл від збору мультимодальних даних до аналітичного профілю кандидата, оптимізованого для подальшого навчання моделей і підвищення якості відбору [13].

Для проведення порівняльного аналізу ефективності різних підходів до автоматизації рекрутингу застосовано систему критеріїв, що охоплює точність, швидкодію, стійкість до форматів даних та тлумачуваність результатів. У таблиці наведено узагальнену характеристику методів - від класичних ATS-систем до сучасних мультимодальних LLM-архітектур, що поєднують NLP, CV і RAG-підходи.

Як показано у таблиці 1.2, статистичні та rule-based рішення залишаються придатними для малих обсягів даних, але поступаються за точністю та гнучкістю моделям на основі трансформерів. Алгоритмічні DSS-системи забезпечують прозорість прийняття рішень, проте мають обмежену здатність до контекстного узагальнення. Глибокі нейронні мережі (BERT, Sentence-Transformers) значно підвищують якість семантичного зіставлення, а мультимодальні LLM-моделі демонструють найвищу ефективність у роботі з неструктурованими резюме, соціальними профілями та відеоінтерв’ю [15].

Таблиця 1.2

Порівняльна характеристика підходів до автоматизації рекрутингу

Метод / Система	Тип моделі	Точність (Precision / F1)	Швидкодія (обробка резюме/с)	Приклади / технології
Rule-based / експертні ATS	Статистична, експертна	0.65 / 0.58	≈ 25 – 30	Regex, TF-IDF, Lucene
DSS (Decision Support System)	Алгоритмічна (ILP, TOPSIS)	0.72 / 0.68	≈ 40 – 45	AHP, TOPSIS, скорингові моделі
OCR + ML ATS	ML (класифікаційна)	0.78 / 0.74	≈ 35 – 40	scikit-learn, pdfminer, Weka
BERT / Sentence-BERT	Контекстна NLP-модель	0.87 / 0.85	≈ 20 – 25	Hugging Face Transformers, PyTorch
LLM (GPT-4V, Claude, Gemini)	Мультимодальна LLM	0.91 / 0.89	≈ 15 – 20	GPT-4V, RAG, CLIP, LangChain

Продовження таблиці 1.2

Метод / Система	Стійкість до форматів даних	Тлумачуваність
Rule-based / експертні ATS	Низька (чутливість до структури CV)	Висока (прості правила)
DSS (Decision Support System)	Середня	Висока
OCR + ML ATS	Висока для PDF / DOCX	Середня
BERT / Sentence-BERT	Висока (багатомовна)	Середня / обмежена
LLM (GPT-4V, Claude, Gemini)	Дуже висока (текст + зображення + відео)	Обмежена, компенсується SHAP/LIME

З наведеного порівняння видно, що спостерігається стійка тенденція переходу від ручних і статистичних методів до контекстно-семантичних LLM-моделей, які забезпечують комплексне розуміння змісту резюме та вакансій. Такі системи демонструють найвищу точність за рахунок глибокого контекстного навчання та можливості обробки мультимодальних даних, проте потребують значних обчислювальних ресурсів і додаткових механізмів пояснюваності [16].

У результаті проведеного огляду сучасних підходів до автоматизації процесів рекрутингу встановлено, що еволюція технологій від традиційних статистичних і rule-based систем до глибоких контекстно-залежних моделей значно підвищила ефективність обробки резюме та вакансій. Класичні ATS і DSS-рішення забезпечують структурну впорядкованість і контрольованість рішень, проте не здатні адекватно інтерпретувати семантику текстових даних. Методи машинного навчання, зокрема на базі TF-IDF, Word2Vec та BERT, суттєво розширили можливості семантичного зіставлення та класифікації, водночас потребуючи великих обсягів якісно анотованих даних для навчання. Найбільш перспективним напрямом виявилися мультимодальні та LLM-моделі, які здатні інтегрувати інформацію з різних джерел (текст, зображення, аудіо) і створювати контекстно-повні профілі кандидатів [15, 16].

Водночас визначено низку системних проблем і обмежень, характерних для поточних досліджень і практичних реалізацій: (1) недостатня кількість репрезентативних і збалансованих датасетів, що ускладнює генералізацію моделей; (2) наявність алгоритмічної та вибіркової упередженості, яка може призводити до дискримінаційних рішень; (3) феномен “галюцинацій” у великих мовних моделях, коли генеруються неправдиві або неузгоджені факти; (4) ризики розголошення персональних даних та проблеми з дотриманням вимог GDPR у процесі обробки резюме.

З огляду на виявлені тенденції обґрунтовано необхідність подальших досліджень, спрямованих на створення надійних, прозорих і пояснюваних інтелектуальних систем рекрутингу, що базуються на великих мовних і мультимодальних моделях із вбудованими механізмами контролю галюцинацій,

адаптивного навчання та етичного аудиту. Подальший розвиток таких систем має бути зосереджений на забезпеченні балансу між високою точністю прогнозів, конфіденційністю обробки персональних даних і можливістю інтерпретації результатів для користувача, що є ключовою умовою їх впровадження в корпоративних середовищах рекрутингу нового покоління.

1.2 Аналіз існуючих рішень

Аналіз сучасних рекрутингових платформ свідчить про активну інтеграцію методів аналітики даних, алгоритмів машинного навчання та автоматизованих механізмів оцінювання компетентностей, що дозволяє зменшити витрати часу на попередній відбір та підвищити об’єктивність прийняття кадрових рішень.

Одним із найбільш поширених рішень є система LinkedIn Talent Insights, орієнтована на стратегічне HR-планування та оцінювання глобальних і локальних ринків праці. На рисунку (рис. 1.10) наведено типовий аналітичний дашборд, який демонструє географічний розподіл фахівців, популярні навички, титули, роботодавців та метрики залученості. Така структура даних дає змогу формувати прогностичні моделі щодо доступності талантів та приймати рішення щодо найму на основі фактичних ринкових показників.

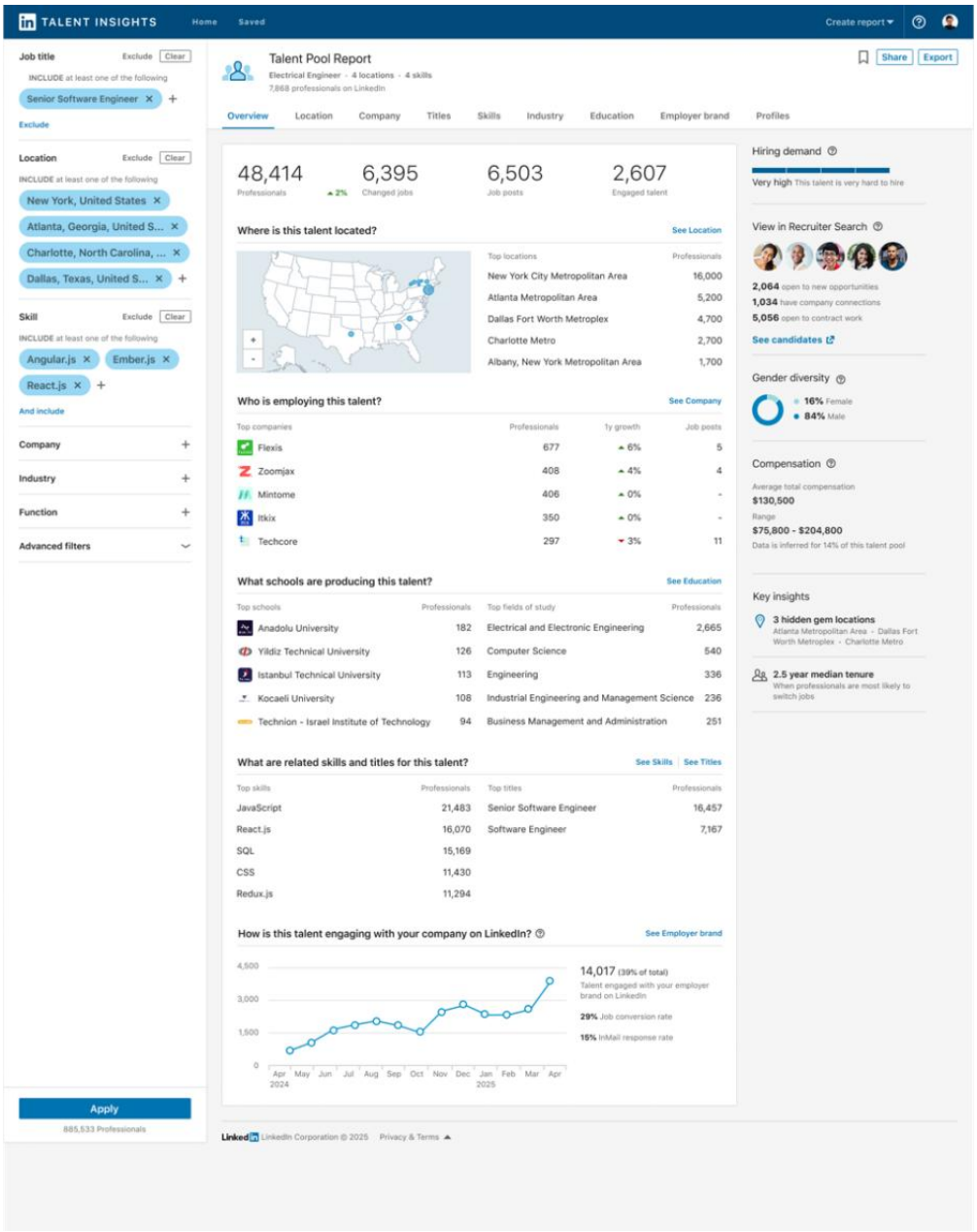


Рис. 1.10 Аналітичний дашборд системи LinkedIn Talent Insights

Ще одним високорівневим рішенням є Workday Recruiting, призначене для великих корпоративних середовищ. Платформа забезпечує автоматизацію всього рекрутингового конвеєра - від реєстрації вакансії до фінального ухвалення рішення. На рисунку показано інтерфейс керування кандидатами (рис. 1.11), у якому представлено етапи скринінгу, запланованих інтерв'ю, перевірок бекграунду та узгодження статусів. Workday інтегрується з ERP-системами підприємства, забезпечує централізоване керування потоками кандидатів та підтримує складні бізнес-процеси, однак вимагає значних ресурсів для впровадження.

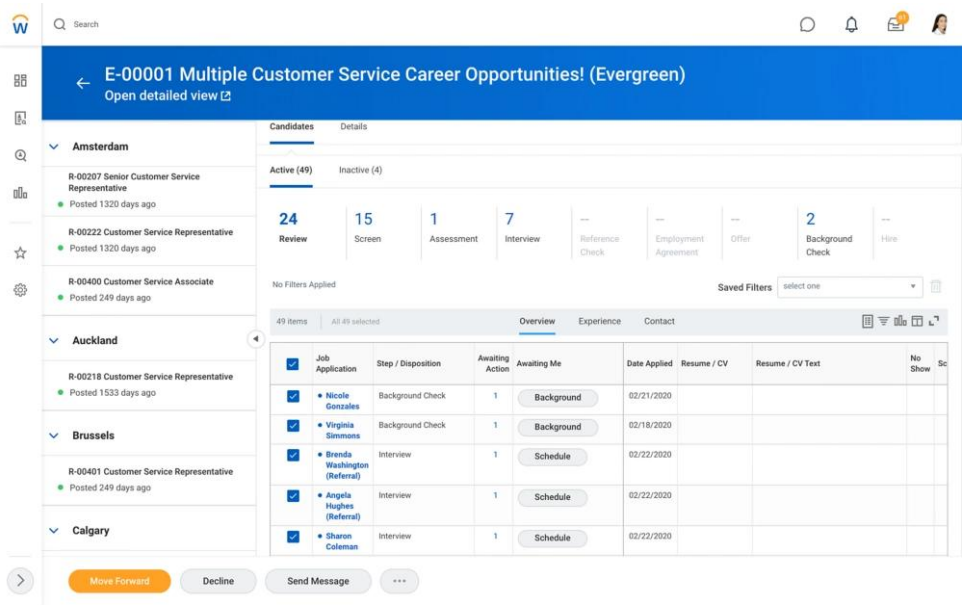


Рис. 1.11 Інтерфейс керування кандидатами у системі Workday Recruiting

Система BambooHR ATS орієнтована на малі та середні компанії та забезпечує базові функції відстеження кандидатів, управління статусами та формування рейтингових оцінок. На рисунку (рис. 1.12) продемонстровано інтерфейс з оглядом кандидатів, їх актуальних станів, рейтингу та історії комунікації. Хоча функціональні можливості цієї платформи обмежені порівняно з корпоративними системами, її перевагами є простота, мінімальні вимоги до інфраструктури та швидкість інтеграції.

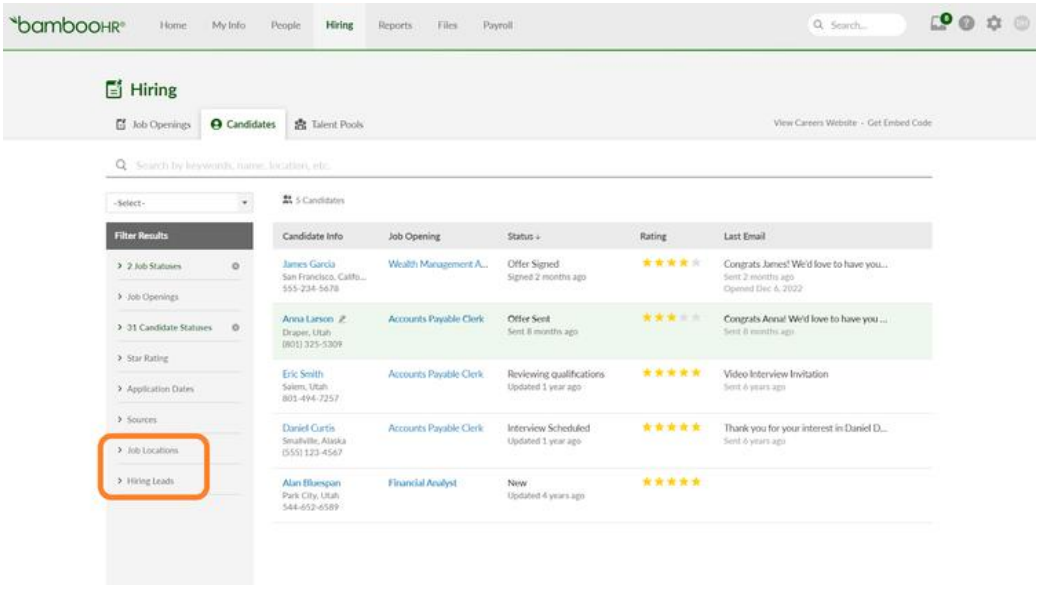


Рис. 1.12 Інтерфейс відстеження кандидатів та їх статусів у BambooHR ATS

Більш інтелектуалізованим рішенням є Manatal, яке впроваджує алгоритми машинного навчання для автоматичного ранжування кандидатів, аналізу навичок і побудови рекомендаційних оцінок. На рисунку (рис. 1.13) наведено модуль автоматизованого оцінювання компетентностей, де система аналізує технічні й переносні навички, досвід та показники відповідності вакансії, формуючи інтегральний match-score. Такий підхід забезпечує високу швидкість первинного скринінгу та підвищує об’єктивність відбору.

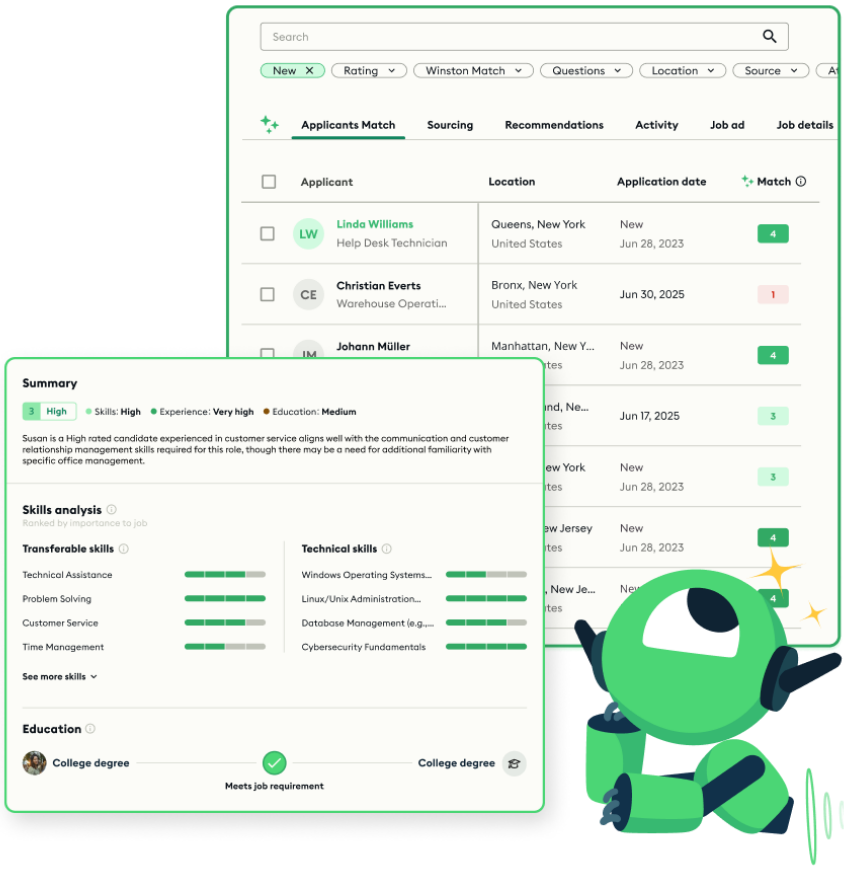


Рис. 1.13 Модуль оцінювання компетентностей та релевантності у системі Manatal

Окреме місце серед корпоративних рішень займає Oracle Taleo Recruiting Cloud, що використовується глобальними компаніями з високими вимогами до масштабованості, аналітики та інтегрованості з HR-екосистемою. На рисунку подано аналітичну панель Taleo (рис. 1.14), яка містить агреговані показники щодо статусів реквізицій, графічні KPI, кількість відкритих вакансій та аналітику щодо

етапів рекрутингу. Система забезпечує централізоване оброблення великих обсягів кандидатських даних, проте її впровадження є складним та ресурсомістким.

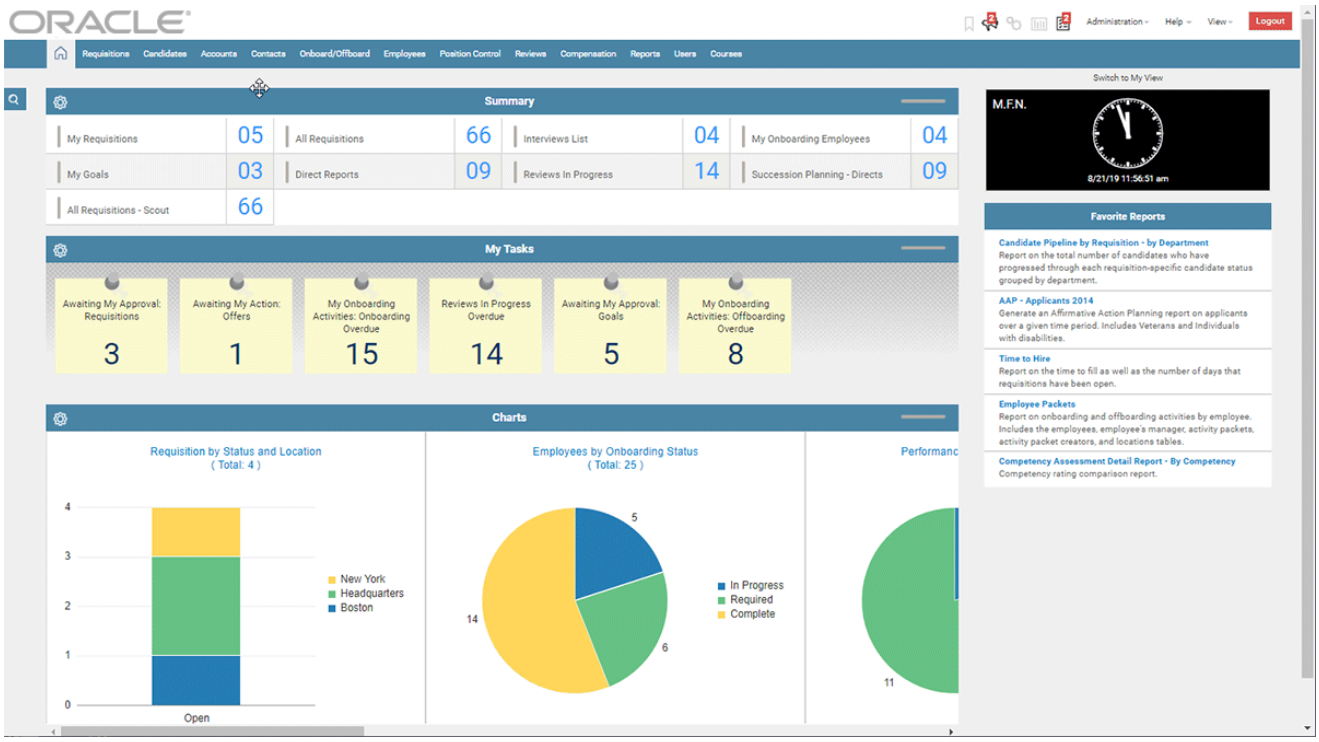


Рис. 1.14 Аналітична панель рекрутингових процесів у Oracle Taleo Recruiting

Для узагальнення функціональних можливостей наведених платформ сформовано порівняльну таблицю, у якій враховано ключові характеристики: наявність алгоритмів машинного навчання, підтримку NLP-аналізу резюме, інтеграцію з ATS, рівень автоматизації рекрутингових етапів, глибину аналітики та можливість адаптивного ранжування кандидатів. У таблицю також включено нашу інтелектуальну систему оптимізації рекрутингу, у якій передбачено розширену підтримку ML-модулів, семантичний аналіз резюме, механізм формування релевантності, адаптивний профіль кандидата та модуль прогнозування відповідності вакансіям (таблиця 1.3). Порівняльний аналіз підтверджує, що незважаючи на функціональність існуючих систем, жодне із рішень не поєднує комплексну NLP-обробку, повністю адаптивне ML-ранжування та механізми пояснюваності результатів, що обґрунтовує необхідність розроблення проєктованої системи.

Таблиця 1.3

Порівняльний аналіз функціональних можливостей рекрутингових платформ

Критерій	LinkedIn Talent Insights	Workday Recruiting	BambooHR ATS
Тип системи	Аналітична HR-платформа	Корпоративна HR-система	ATS-система
Автоматичний аналіз кандидатів	+	+	±
Машинне навчання	±	±	–
NLP-аналіз резюме	–	–	–
Ранжування кандидатів	±	+	+
Аналітичні звіти	+	+	±
Простота використання	Середня	Низька	Висока
Масштабованість	Висока	Висока	Середня
Гнучкість налаштування	Середня	Висока	Середня

Продовження таблиці 1.3

Критерій	Manatal	Oracle Taleo Recruiting	TalentRadar Recruit AI
Тип системи	Інтелектуальна рекрутинг-система	Інтелектуальна система	Інтелектуальна система
Автоматичний аналіз кандидатів	+	+	+
Машинне навчання	+	+	+
NLP-аналіз резюме	+	+	+

Продовження таблиці 1.3

Порівняльний аналіз функціональних можливостей рекрутингових платформ

Критерій	Manatal	Oracle Taleo Recruiting	TalentRadar Recruit AI
Ранжування кандидатів	+	+	+
Аналітичні звіти	+	+	+
Простота використання	Середня	Середня	Висока
Масштабованість	Середня	Середня	Висока
Гнучкість налаштування	Висока	Висока	Висока

1.3 Аналіз вимог системи рекрутингу

Розроблення експертної системи формального визначення вимог, що регламентують оптимізації процесу рекрутингу потребує функціональну поведінку системи, її продуктивність, надійність, захищеність та здатність до інтеграції з зовнішніми HR-сервісами. Вимоги формуються на основі аналізу предметної області, характеристик існуючих рішень, бізнес-процесів рекрутингу та специфіки опрацювання текстових і мультимодальних даних кандидатів.

Функціональні вимоги експертної системи оптимізації рекрутингу:

1. Система повинна приймати резюме у форматах PDF, DOCX, HTML і виконувати їх автоматизоване парсування;

2. Модуль повинен виконувати видобування ключових компетенцій, вимог, досвіду та умов праці з описів вакансій;

3. Система повинна здійснювати токенізацію, лематизацію, NER, нормалізацію навичок та POS-аналіз;

4. На основі витягнутих сутностей система формує уніфікований структурований профіль;

5. Модель машинного навчання повинна обчислювати match-score між резюме та вакансією;

6. Система повинна будувати топ-N список кандидатів за інтегральним індексом відповідності;

7. Система повинна пропонувати альтернативні вакансії кандидатам або альтернативних кандидатів рекрутеру;

8. Платформа повинна візуалізувати статистику найму, ефективність моделей та метрики відбору;

9. Усі операції користувача повинні логуватися з мітками часу для подальшого аудиту;

10. Система повинна підтримувати REST API для зв’язку з ATS/CRM-платформами.

Нефункціональні вимоги відображають характеристики якості експертної системи та її здатність працювати стабільно в умовах змінних обсягів даних і високої інтенсивності запитів.

Нефункціональні вимоги експертної системи оптимізації рекрутингу:

1. Час обробки одного резюме не повинен перевищувати 1 секунду при навантаженні до 50 одночасних користувачів.

2. Мінімальний коефіцієнт F1 для класифікації релевантності – 0.85.

3. Система повинна підтримувати горизонтальне масштабування сервісів ML/NLP.

4. Гарантована доступність сервісу – не менше 99.5%.

5. Повна підтримка PDF, DOCX, RTF, HTML + автономний fallback OCR.

6. Інтерфейс повинен забезпечувати час реакції менше 200 мс.

7. Підтримка української та англійської мов обробки текстів.

8. Система повинна вести структурований журнал усіх операцій ML-модулів.

9. Платформа має працювати у Docker-середовищі та підтримувати Kubernetes.

10. Моделі повинні проходити періодичне перенавчання не рідше одного разу на місяць.

Окремо виділено вимоги до захищеності системи, оскільки експертна система працює з персональними даними кандидатів, інформацією про досвід, освіту, сертифікації та інші чутливі відомості. Захист інформації має ґрунтуватися на сучасних криптографічних протоколах і відповідати нормативним вимогам GDPR та політикам конфіденційності HR-процесів.

Вимоги до безпеки експертної системи оптимізації рекрутингу:

1. Доступ до системи здійснюється через OAuth2 / JWT-токени.
2. Рольова модель доступу (Admin, Recruiter, Analyst, Viewer).
3. Дані повинні зберігатися у зашифрованому вигляді (AES-256) та передаватися через TLS 1.3.
4. Відповідність вимогам GDPR, автоматичне маскування PII-даних.
5. Виявлення аномальної активності (IDS) у запитах користувачів.
6. Усі доступи до персональних даних фіксуються у захищеному журналі безпеки.
7. Щоденне резервне копіювання БД зі зберіганням історії не менше 30 днів.
8. Перевірка хеш-контрольних сум документів резюме.
9. Перевірка справжності файлів резюме (MIME-аналіз, сигнатури).
10. Обмеження для запобігання prompt injection, data poisoning та model inversion атак.

Важливим елементом є реалізація журнала безпеки, який забезпечує фіксацію доступу до персональних даних, формування аудиторських слідів та контроль відповідності операцій політикам доступу.

Проведений аналіз встановлює, що формалізація вимог є ключовою основою побудови експертної системи, оскільки дозволяє узгодити технічні обмеження, бізнес-цілі, вимоги до моделей машинного навчання та нормативні критерії опрацювання персональних даних. Сформульовані вимоги забезпечують цілісний фундамент для подальшого проєктування архітектури, моделювання інформаційних потоків і реалізації функціональних модулів системи, гарантуючи її точність, надійність і відповідність практичним потребам рекрутингових процесів.

1.4 Постановка завдання

Постановка завдання визначає цілісну структуру функціонування експертної системи оптимізації рекрутингу та формалізує взаємозв’язки між вхідними, проміжними та вихідними даними. Основою побудови системи є моделі обробки текстової та мультимодальної інформації, включно з даними резюме, описів вакансій, історичних записів про прийняття рішень та метаданими рекрутингових процесів.

Вхідні дані надходять до системи із зовнішніх джерел — ATS-платформ, електронної пошти, форм завантаження та сторонніх HR-сервісів — і проходять етапи попередньої обробки, нормалізації та семантичного аналізу. Структуровані вхідні дані використовуються для формування профілю кандидата, зіставлення його компетентностей із вимогами вакансії, прогнозування відповідності та ранжування альтернатив.

У таблиці наведено узагальнену характеристику вхідних та вихідних даних, що визначають логічну модель роботи системи (таблиця 1.4 та 1.5). Вхідні дані охоплюють резюме в різних форматах, описи вакансій, метрики продуктивності та історичні рішення рекрутерів.

Вихідними даними є структуровані профілі кандидатів, значення релевантності, інтегральні показники відповідності, ранжовані списки кандидатів та звітні аналітичні метрики. Отримані результати забезпечують можливість автоматизованого оцінювання, підтримки прийняття рішень і формування рекомендацій для оптимізації рекрутингового процесу.

Таблиця 1.4

Вхідні дані експертної системи оптимізації рекрутингу

№	Тип даних	Характеристика
1	Резюме (PDF, DOCX, HTML)	Первинні текстові документи, що містять інформацію про досвід, освіту, навички, сертифікації.
2	Описи вакансій	Текстові профілі посад із вимогами, компетенціями та умовами праці.
3	Метадані найму	Статуси, результати інтерв'ю, попередні рішення рекрутерів.
4	Соціальні профілі та портфоліо	Додаткові джерела перевірки навичок і досвіду кандидата.
5	Історичні дані моделей	Попередні значення match-score, ранжування, оцінки точності моделей.

Таблиця 1.5

Вихідні дані експертної системи оптимізації рекрутингу

№	Тип даних	Характеристика
1	Структурований профіль кандидата	Нормалізований набір сутностей (посади, навички, досвід, освіта).
2	Релевантність кандидата	Числове значення match-score, сформоване моделлю ML/NLP.
3	Ранжований список кандидатів	Відсортований перелік кандидатів за інтегральною оцінкою відповідності.
4	Аналітичні метрики відбору	Статистика, графіки, агреговані показники роботи моделей.
5	Рекомендації	Альтернативні вакансії для кандидата або альтернативні кандидати для вакансії.

Поставлене завдання полягає у створенні експертної системи, здатної забезпечити повний цикл обробки даних - від аналізу резюме та вакансій до формування обґрунтованих оцінок відповідності кандидатів. Система повинна реалізовувати модулі NLP-аналізу, моделі машинного навчання для класифікації та ранжування, механізми формування інтегрального профілю та засоби генерації аналітичних висновків.

Крім того, важливим завданням є забезпечення точності моделей, стійкості до неструктурованих даних, інтерпретованості результатів і відповідності вимогам безпеки щодо опрацювання персональних даних. Сукупність цих характеристик визначає практичну цінність та ефективність експертної системи, забезпечуючи її застосовність у реальних рекрутингових процесах.

1.5 Висновки до першого розділу

У першому розділі було здійснено системний аналіз сучасного стану, методів та технологій автоматизації рекрутингових процесів, що дало змогу сформувати комплексне бачення еволюції галузі – від rule-based систем і статистичних методів до глибоких контекстно-семантичних моделей та мультимодальних архітектур з використанням NLP, CV та ML-алгоритмів.

Проведений огляд підтвердив, що класичні ATS та DSS-рішення забезпечують базову структурованість процесу найму, проте є обмеженими у здатності обробляти неструктуровані текстові дані та враховувати контекст взаємозв’язків між навичками й вимогами вакансій.

Методи машинного навчання та глибокого контекстного моделювання значно підвищують точність оцінювання релевантності, проте потребують високих обчислювальних ресурсів, якісних анотованих датасетів і наявності механізмів пояснюваності результатів.

Виконаний аналіз існуючих ПЗ-рішень (LinkedIn Talent Insights, Workday Recruiting, BambooHR ATS, Manatal, Oracle Taleo) засвідчив, що жодна з платформ не забезпечує комплексної інтеграції адаптивного ML-ранжування, глибокої NLP-

обробки та аналітичних моделей із урахуванням пояснюваності та мультимодальної взаємодії даних.

На основі цього було сформовано вимоги до системи, що включають функціональні модулі обробки резюме та вакансій, семантичного зіставлення профілів, побудови індексів релевантності, ранжування кандидатів та генерації аналітичних висновків, а також нефункціональні критерії продуктивності, масштабованості, точності моделей і відповідності нормам безпеки. Формалізація вхідних та вихідних даних у межах постановки завдання дозволила визначити логічну структуру системи та взаємозв’язки між її компонентами.

Отримані результати створюють теоретичну та методичну основу для розроблення архітектури експертної системи оптимізації рекрутингу, що забезпечуватиме високий рівень автоматизації, точності та надійності в реальних умовах функціонування.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальна структура проєктованої системи

Інтелектуальна система оптимізації процесу рекрутингу проєктується як веборієнтоване програмне забезпечення, призначене для автоматизації основних етапів підбору персоналу: подання заявки кандидатом, завантаження резюме, створення вакансії, аналізу відповідності кандидата вимогам вакансії, ранжування претендентів, формування рекомендацій і підтримки кадрового рішення.

Основна відмінність системи полягає у використанні NLP- та ML-модулів, які забезпечують автоматизоване виділення навичок, досвіду, освіти, ключових слів, семантичних ознак і формування показника відповідності кандидата вакансії.

Система орієнтована на декілька категорій користувачів. Кандидат взаємодіє із системою через вебінтерфейс, подає заявку, завантажує резюме та отримує повідомлення про статус розгляду. Рекрутер створює вакансії, запускає аналіз кандидатів, переглядає рейтинг, пояснення ML-рекомендацій і приймає попереднє рішення. Менеджер з найму оцінює відібраних кандидатів, погоджує рішення або повертає заявку на доопрацювання. Адміністратор керує користувачами, ролями, налаштуваннями системи, аудитом і контролем роботи програмних модулів.

Архітектура системи побудована за модульним принципом. До складу системи входять вебінтерфейс, модуль автентифікації, модуль керування вакансіями, модуль керування кандидатами, модуль обробки заявок, NLP/ML-модуль аналізу, модуль звітності, модуль сповіщень, база даних та модуль інтеграції з ATS/CRM.

Такий поділ дозволяє ізолювати функціональні частини системи, спростити супровід програмного забезпечення та забезпечити можливість подальшого розширення.

У загальному вигляді інформаційний процес системи починається із надходження даних про кандидата та вакансію. Після цього виконується перевірка резюме, парсинг тексту, NLP-обробка, формування структурованого профілю кандидата, обчислення Match Score, ранжування та відображення результатів користувачу. На основі сформованої оцінки рекрутер або менеджер з найму ухвалює рішення щодо подальшого етапу взаємодії з кандидатом.

Діаграма прецедентів системи наведена на рисинку 2.1.

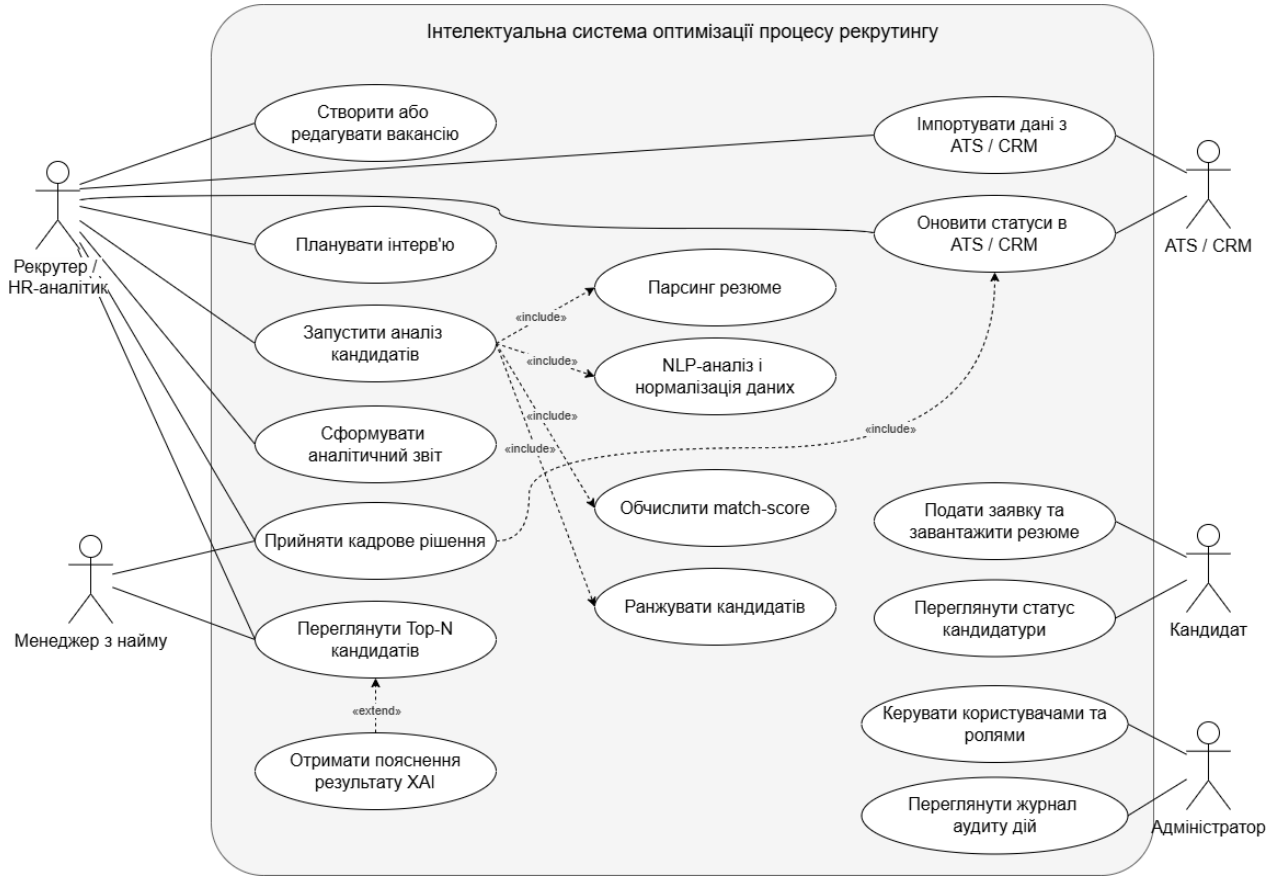


Рис. 2.1 Діаграма прецедентів інтелектуальної системи оптимізації процесу рекрутингу

Діаграма прецедентів відображає основні ролі користувачів і функціональні можливості системи. До базових прецедентів належать подання заявки, перегляд статусу кандидатури, створення або редагування вакансії, запуск аналізу кандидатів, парсинг резюме, NLP-аналіз, обчислення match-score, ранжування кандидатів, отримання пояснення результату XAI, прийняття кадрового рішення,

формування аналітичного звіту, імпорт і оновлення даних в ATS/CRM. Зв’язки include показують, що оцінювання кандидата неможливе без попереднього парсингу резюме, NLP-обробки та обчислення показника відповідності. Зв’язки extend відображають додаткові сценарії, наприклад отримання пояснення результату після перегляду Top-N кандидатів.

2.2 Проєктування поведінки системи

Для опису часової логіки роботи програмного забезпечення побудовано діаграму послідовності, наведену на рисинку 2.2.

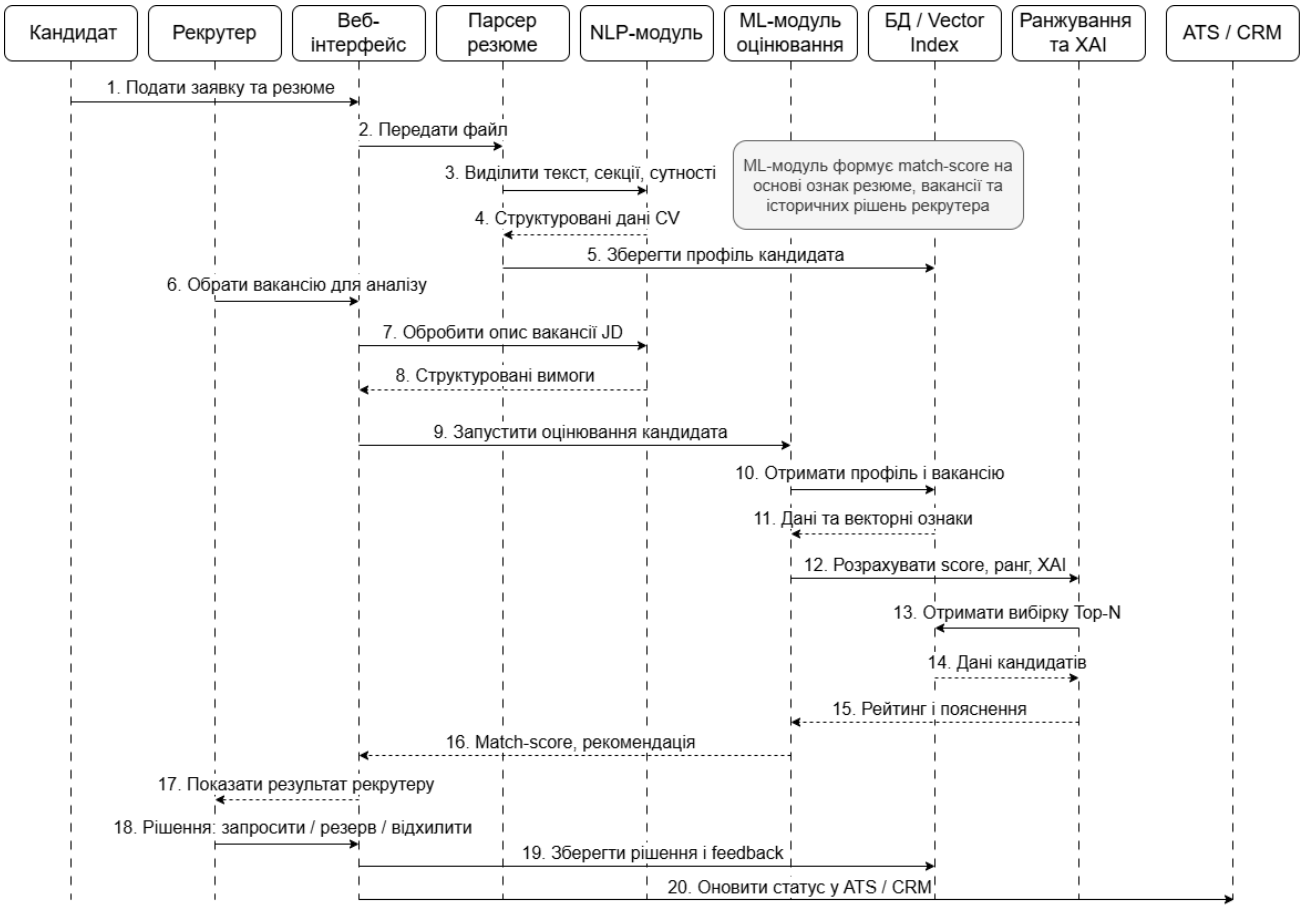


Рис. 2.2 Діаграма послідовності процесу автоматизованого оцінювання кандидата

Діаграма послідовності показує взаємодію між кандидатом, рекрутером, вебінтерфейсом, парсером резюме, NLP-модулем, ML-модулем оцінювання, базою даних, модулем ранжування, XAI-модулем та ATS/CRM.

Кандидат подає заявку та завантажує резюме через вебінтерфейс. Далі файл передається до парсера резюме, який виділяє текст, секції документа та основні сутності. NLP-модуль виконує нормалізацію даних, виділення навичок і формування структурованого профілю кандидата.

Після вибору вакансії рекрутером система обробляє опис вакансії, виділяє вимоги, компетентності та ключові критерії. ML-модуль отримує профіль кандидата і вимоги вакансії, формує векторні ознаки, обчислює match-score та передає результат до модуля ранжування.

Модуль ранжування порівнює кандидата з іншими претендентами, формує позицію у рейтингу та повертає Top-N список. XAI-модуль формує пояснення, яке містить ключові фактори оцінювання: збіг навичок, відповідність досвіду, освіти, зарплатних очікувань та локації.

Результатом роботи сценарію є відображення рекрутеру рейтингу кандидата, пояснення рекомендації та можливості прийняти рішення: запросити кандидата, залишити у резерві або відхилити. Після цього система зберігає рішення у базі даних і за потреби оновлює статус у зовнішній ATS/CRM.

Логіку бізнес-процесу системи деталізовано за допомогою діаграми активності, наведеної на рисинку 2.3.

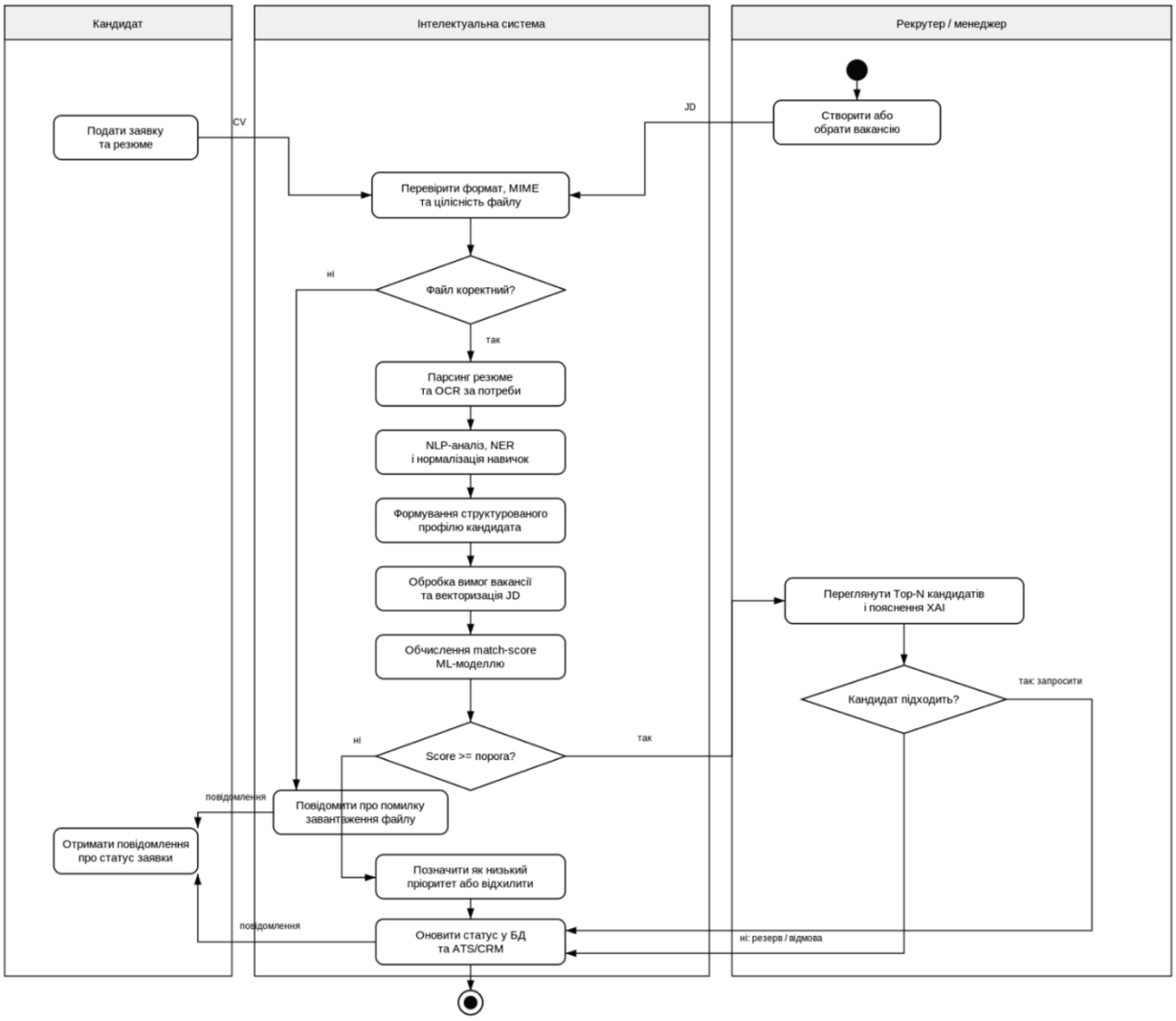


Рис. 2.3 Діаграма активності процесу автоматизованого підбору кандидата

Діаграма активності відображає послідовність дій від створення вакансії та подання резюме до формування оцінки й ухвалення кадрового рішення. На початковому етапі система перевіряє формат і цілісність файлу резюме. Якщо файл некоректний, користувач отримує повідомлення про помилку. Якщо файл відповідає вимогам, виконується парсинг, NLP-аналіз, формування профілю кандидата, обробка вимог вакансії, обчислення match-score та генерація пояснення результату.

Після оцінювання система визначає, чи перевищує показник відповідності встановлений поріг. Якщо значення нижче порогового, кандидат отримує низький пріоритет або відхиляється. Якщо кандидат відповідає вимогам, він додається до Top-N списку, а рекрутер переглядає рейтинг і пояснення. Завершальним етапом є

оновлення статусу заявки, надсилання повідомлення кандидату та збереження зворотного зв’язку для подальшого вдосконалення моделі.

2.3 Діаграма класів системи

Для опису статичної структури програмного забезпечення побудовано UML-діаграму класів, наведену на рисинку 2.4.

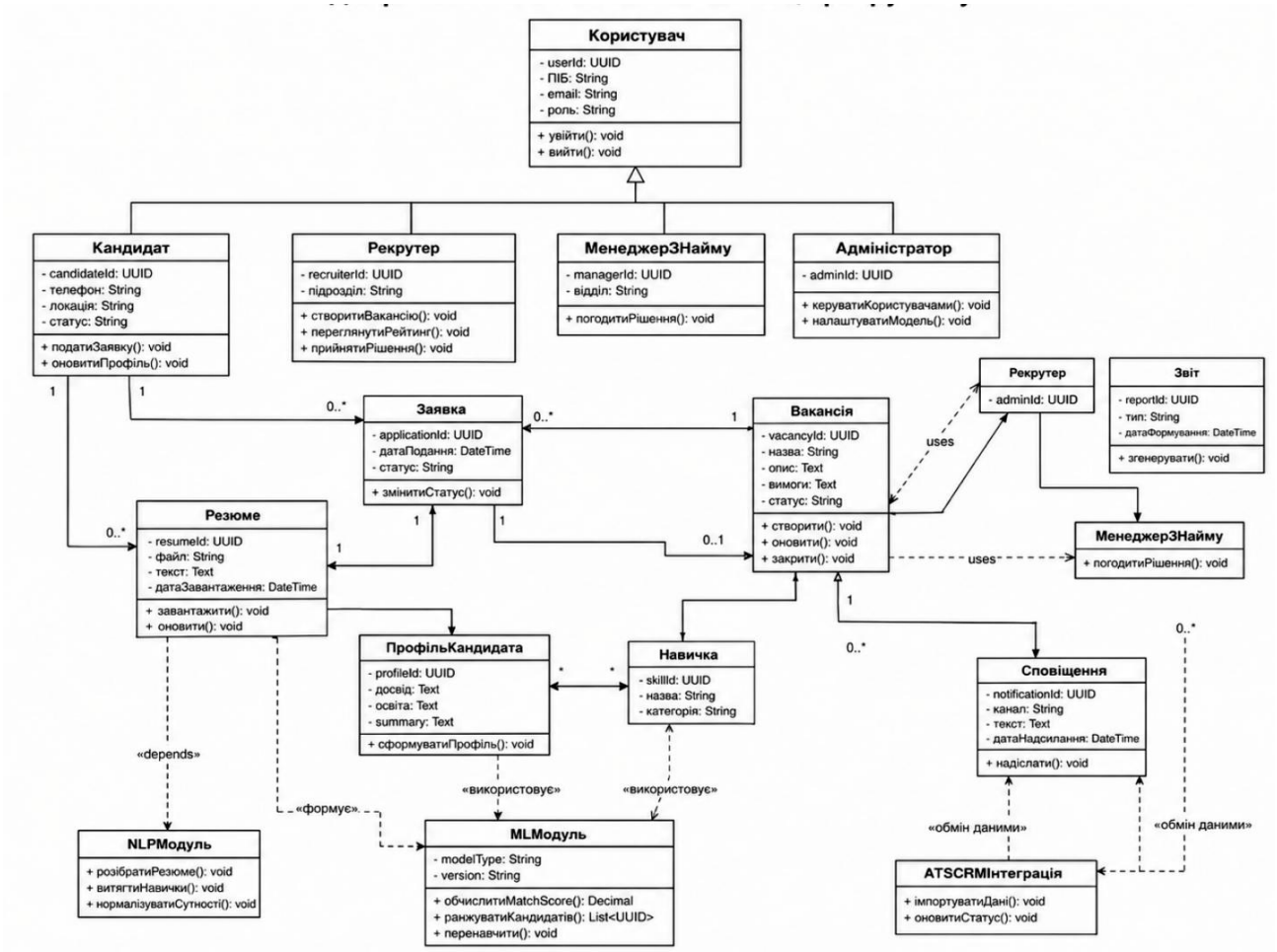


Рис. 2.4 Діаграма класів системи оптимізації рекрутингу

Основним базовим класом системи є «Користувач», який містить загальні атрибути userId, ПІБ, email і роль. Від нього успадковуються класи «Кандидат», «Рекрутер», «МенеджерЗНайму» та «Адміністратор». Така структура забезпечує

єдину логіку автентифікації та рольового доступу, але дозволяє розділити функціональність залежно від типу користувача.

Клас «Кандидат» пов’язаний з класами «Резюме» та «Заявка». Один кандидат може мати декілька резюме та декілька заявок на різні вакансії. Клас «Резюме» зберігає інформацію про файл, текст документа та дату завантаження. На основі резюме формується об’єкт «ПрофільКандидата», який містить досвід, освіту, узагальнений опис і набір навичок. Клас «Навичка» використовується для уніфікованого представлення технічних, професійних і загальних компетентностей кандидата.

Клас «Вакансія» містить назву, опис, вимоги та статус вакансії. Він пов’язаний із заявками кандидатів і використовується ML-модулем для обчислення відповідності. Клас «Оцінювання» зберігає результати аналізу: matchScore, sentimentScore, topicFit і rankPosition. Саме цей клас відображає результат інтелектуального аналізу кандидата та використовується під час ранжування і прийняття кадрового рішення.

Допоміжні класи «NLPМодуль» і «MLМодуль» реалізують спеціалізовану логіку аналізу. NLP-модуль відповідає за парсинг резюме, виділення навичок, нормалізацію сутностей і підготовку текстових ознак. ML-модуль виконує обчислення Match Score, ранжування кандидатів і перенавчання моделі на основі накопичених даних. Класи «Звіт», «Сповіщення» та «ATSCRMІнтеграція» забезпечують аналітику, комунікацію з кандидатами та синхронізацію із зовнішніми HR-системами.

2.4 Діаграми кооперації

Для деталізації взаємодії об’єктів у ключових сценаріях системи побудовано три діаграми кооперації. Перша діаграма описує автоматизоване оцінювання кандидата та наведена на рисинку 2.5.

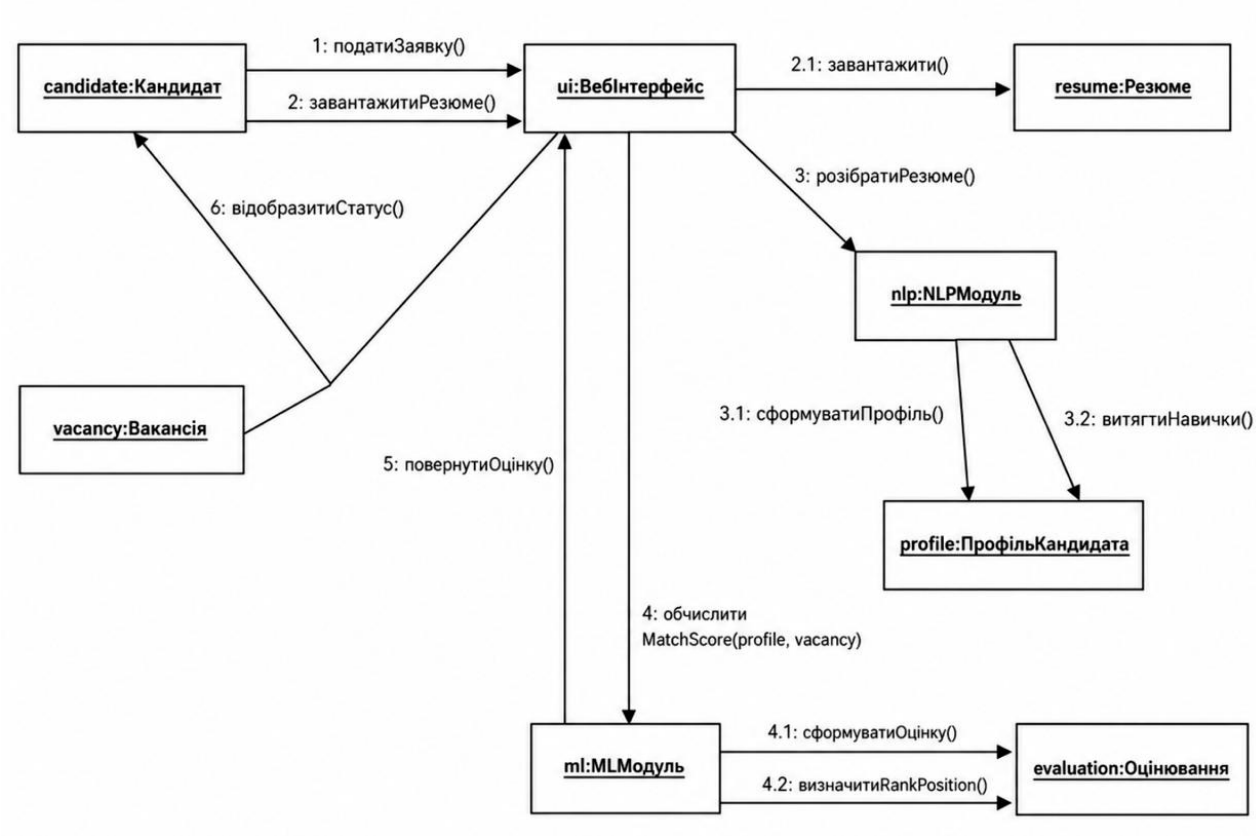


Рис. 2.5 Діаграма кооперації автоматизованого оцінювання кандидата

У межах цього сценарію кандидат подає заявку та завантажує резюме через вебінтерфейс. Інтерфейс передає файл об’єкту «Резюме», після чого NLP-модуль виконує розбір документа, формує профіль кандидата та виділяє навички. Далі ML-модуль отримує профіль кандидата і дані вакансії, обчислює Match Score, формує оцінку та визначає позицію кандидата у рейтингу. Після завершення обробки результат повертається до вебінтерфейсу, а кандидат отримує оновлений статус заявки.

Друга діаграма кооперації описує процес ранжування кандидатів за обраною вакансією і наведена на рисинку 2.6.

У цьому сценарії рекрутер обирає вакансію, після чого вебінтерфейс отримує її дані та імпортує кандидатів із внутрішньої бази або ATS/CRM. ML-модуль формує оцінки для кожного кандидата, сортує їх за Match Score і повертає Top-N список. Рекрутер переглядає рейтинг і за потреби формує звіт. Така кооперація забезпечує автоматизовану підтримку етапу попереднього скринінгу кандидатів.

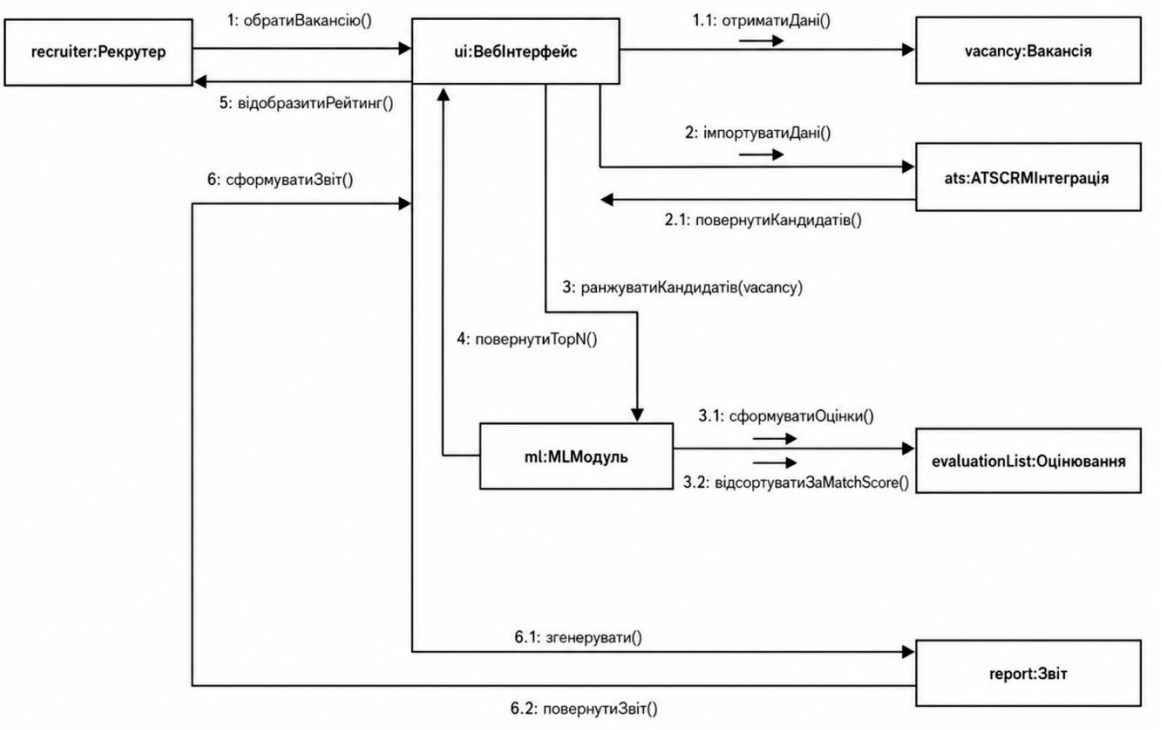


Рис. 2.6 Діаграма кооперації ранжування кандидатів за вакансією

Третя діаграма кооперації відображає процес прийняття кадрового рішення, наведений на рисинку 2.7.

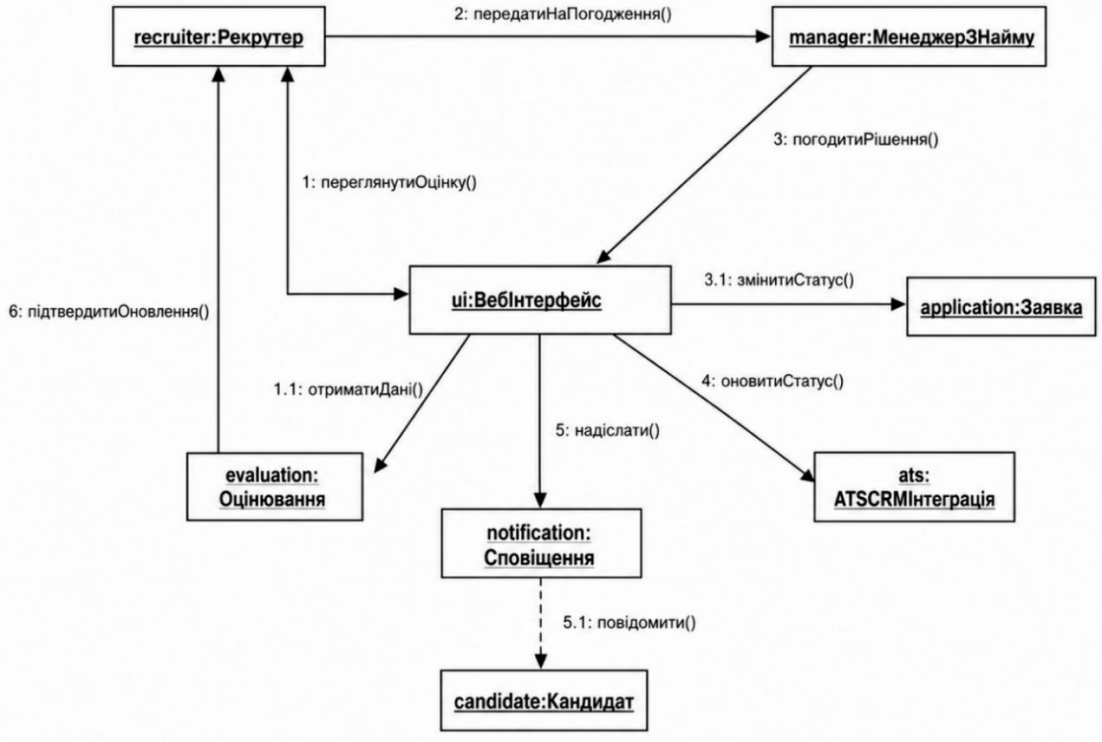


Рис. 2.7 Діаграма кооперації прийняття кадрового рішення

Рекрутер переглядає оцінку кандидата, отримує пояснення результату та передає кандидата на погодження менеджеру з найму. Менеджер аналізує дані та погоджує або відхиляє рішення. Після цього вебінтерфейс змінює статус заявки, передає оновлення до ATS/CRM, формує сповіщення та надсилає повідомлення кандидату. Такий підхід забезпечує контрольованість процесу найму та фіксацію всіх кадрових рішень у системі.

2.5 Компонентна структура системи

Компонентна діаграма системи наведена на рисинку 2.8.

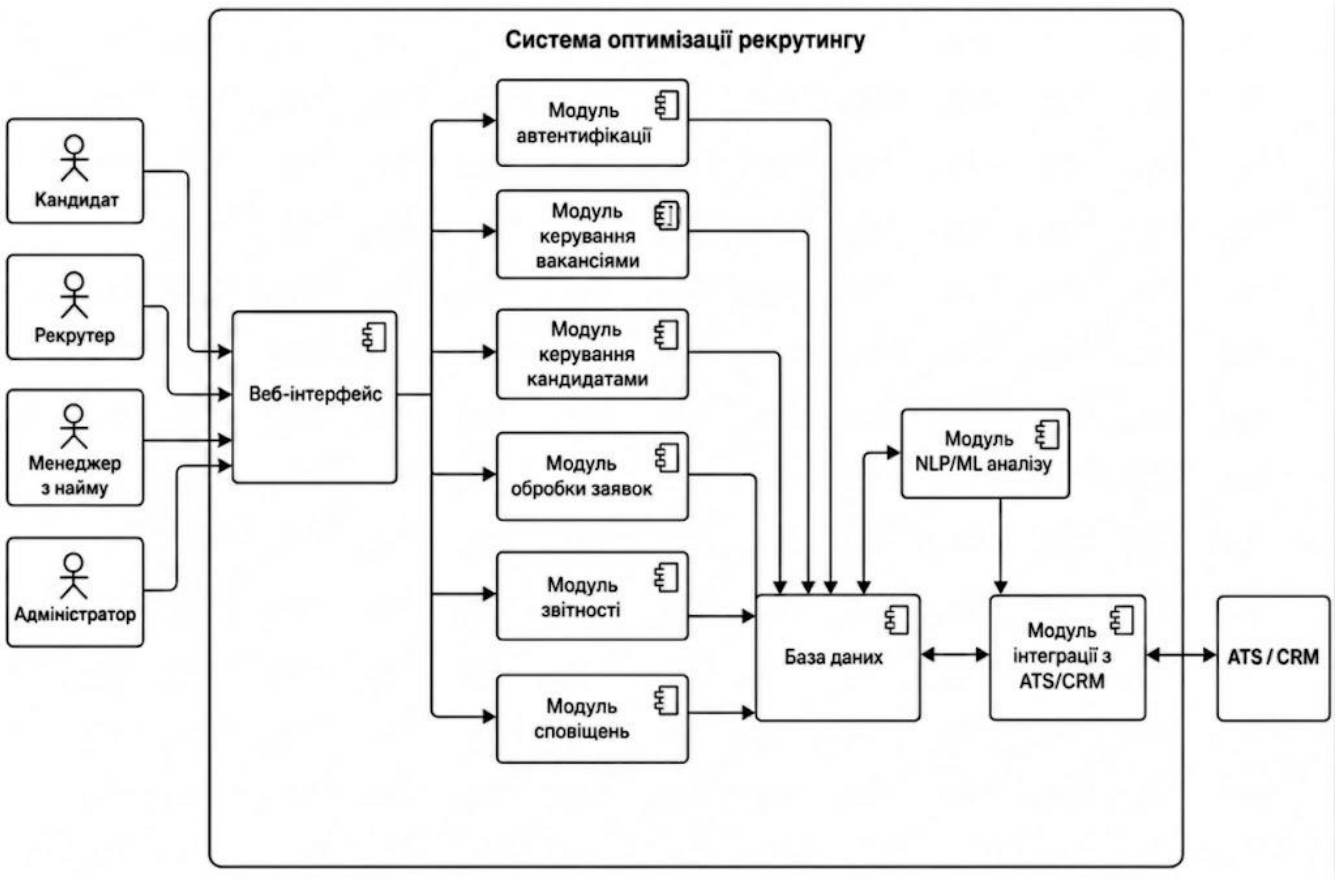


Рис. 2.8 Діаграма компонентів системи оптимізації рекрутингу

Компонентна структура системи складається з клієнтського вебінтерфейсу, набору прикладних модулів, бази даних, модуля NLP/ML-аналізу та інтеграційного

модуля. Кандидат, рекрутер, менеджер з найму й адміністратор працюють із системою через вебінтерфейс. Вебінтерфейс передає запити до модулів автентифікації, керування вакансіями, керування кандидатами, обробки заявок, звітності та сповіщень.

Модуль автентифікації перевіряє користувача та визначає його роль. Модуль керування вакансіями забезпечує створення, редагування та закриття вакансій. Модуль керування кандидатами відповідає за зберігання профілів, резюме, контактних даних і параметрів кандидата. Модуль обробки заявок зв’язує кандидата з вакансією, контролює етап розгляду та передає дані для інтелектуального аналізу.

Модуль NLP/ML-аналізу є центральним інтелектуальним компонентом системи. Він виконує виділення текстових ознак, нормалізацію навичок, оцінювання відповідності резюме вимогам вакансії, ранжування кандидатів і формування пояснення рекомендації. База даних зберігає користувачів, вакансії, кандидатів, заявки, результати оцінювання, журнал дій і службові налаштування. Модуль інтеграції з ATS/CRM забезпечує синхронізацію заявок і статусів із зовнішніми HR-системами.

2.6 Пакетна структура системи

Пакетна структура системи побудована за багаторівневим принципом і включає пакети presentation, application, domain, infrastructure та integration. Пакет presentation містить інтерфейсні частини для кандидата, рекрутера та адміністратора. Пакет application реалізує прикладну логіку: автентифікацію, роботу з вакансіями, обробку заявок, ранжування та формування звітів. Пакет domain містить основні предметні сутності: candidate, vacancy, application та evaluation.

Пакет infrastructure відповідає за технічну підтримку системи: роботу з базою даних, репозиторіями та NLP/ML-двигуном. Пакет integration містить адаптери для взаємодії із зовнішніми системами: ATS, CRM і сервісами сповіщень. Така

структура дозволяє відокремити предметну область від технічної реалізації та спростити подальший супровід системи.

Діаграма пакетів наведена на рисинку 2.9.

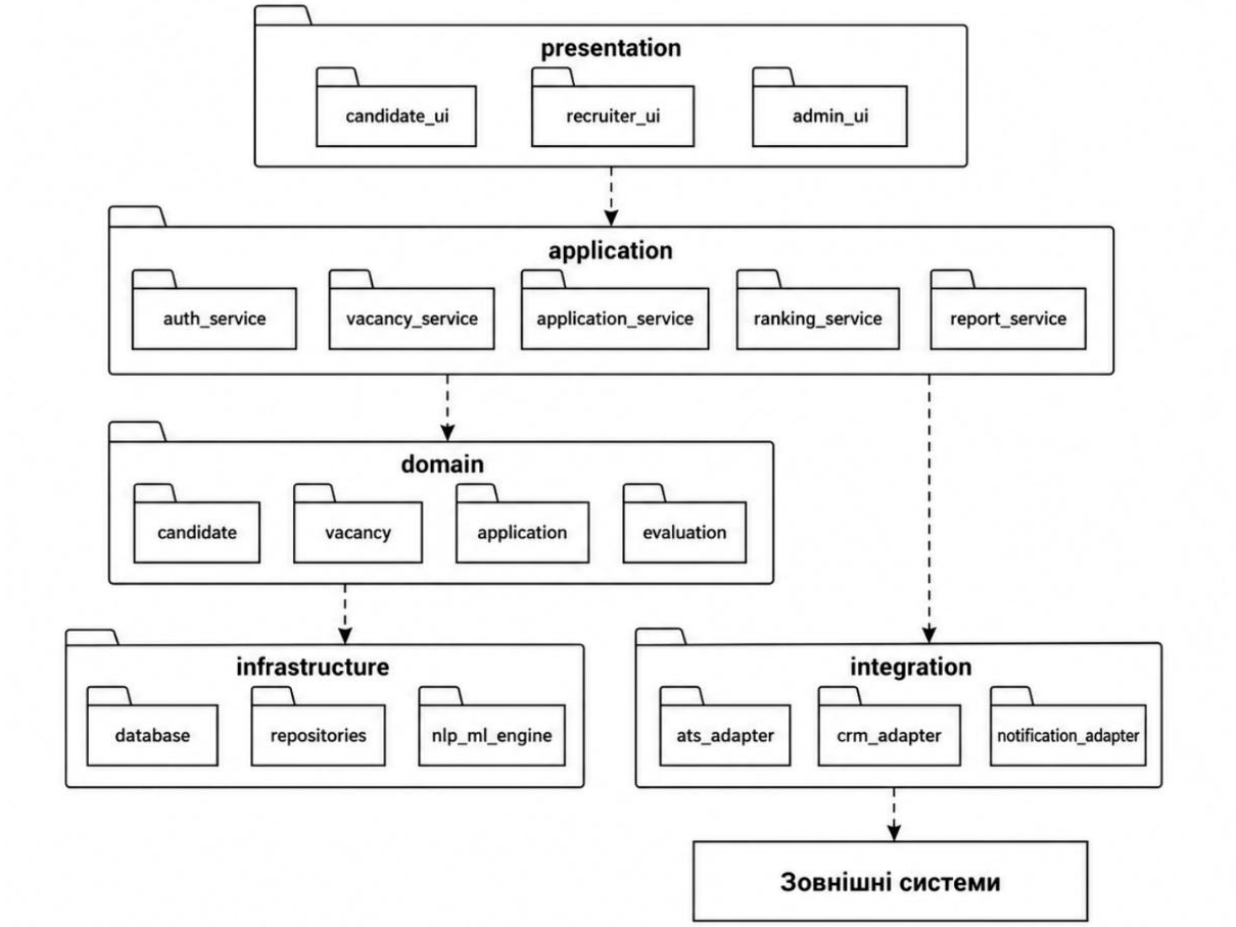


Рис. 2.9 Діаграма пакетів системи оптимізації рекрутингу

2.7 Проєктування алгоритмів роботи системи

Для опису логіки роботи програмних модулів побудовано блок-схеми алгоритмів. Перший алгоритм описує автоматизоване оцінювання кандидата і наведений на рисинку 2.10.

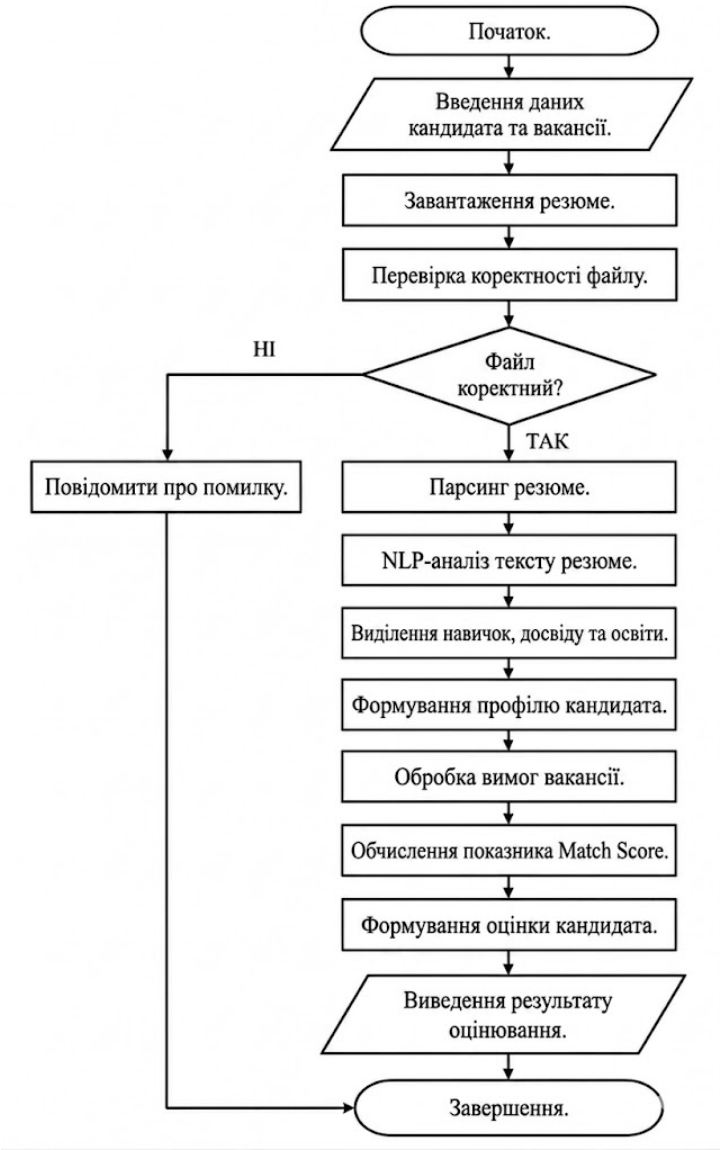


Рис. 2.10 Блок-схема алгоритму автоматизованого оцінювання кандидата

Алгоритм починається з введення даних кандидата та вакансії. Після завантаження резюме система перевіряє коректність файлу. Якщо файл не відповідає вимогам, користувач отримує повідомлення про помилку. Якщо файл коректний, виконується парсинг резюме, NLP-аналіз тексту, виділення навичок, досвіду та освіти. Після цього формується профіль кандидата, обробляються вимоги вакансії та обчислюється Match Score. Завершальним етапом є формування оцінки та виведення результату.

Другий алгоритм описує ранжування кандидатів за вакансією і наведений на рисинку 2.11.

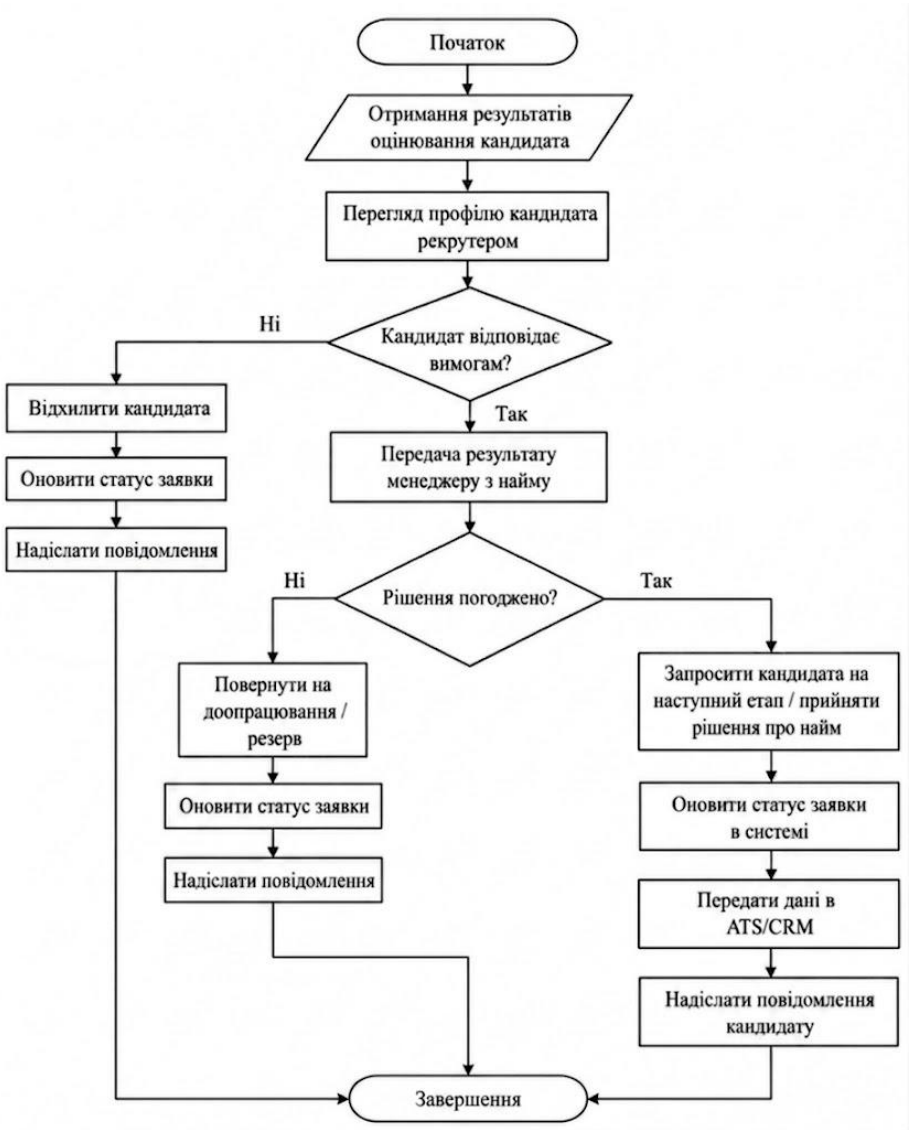


Рис. 2.11 Блок-схема алгоритму ранжування кандидатів за вакансією

Після вибору вакансії система завантажує список кандидатів. Якщо список відсутній, процес завершується без ранжування. Якщо кандидати наявні, система ініціалізує список оцінок і послідовно обробляє кожного кандидата. Для кожного профілю обчислюється Match Score, після чого оцінка додається до загального списку. Коли всі кандидати оброблені, система сортує їх за показником відповідності, формує Top-N список і виводить рейтинг.

Третій алгоритм описує прийняття кадрового рішення і наведений на рисинку 2.12.

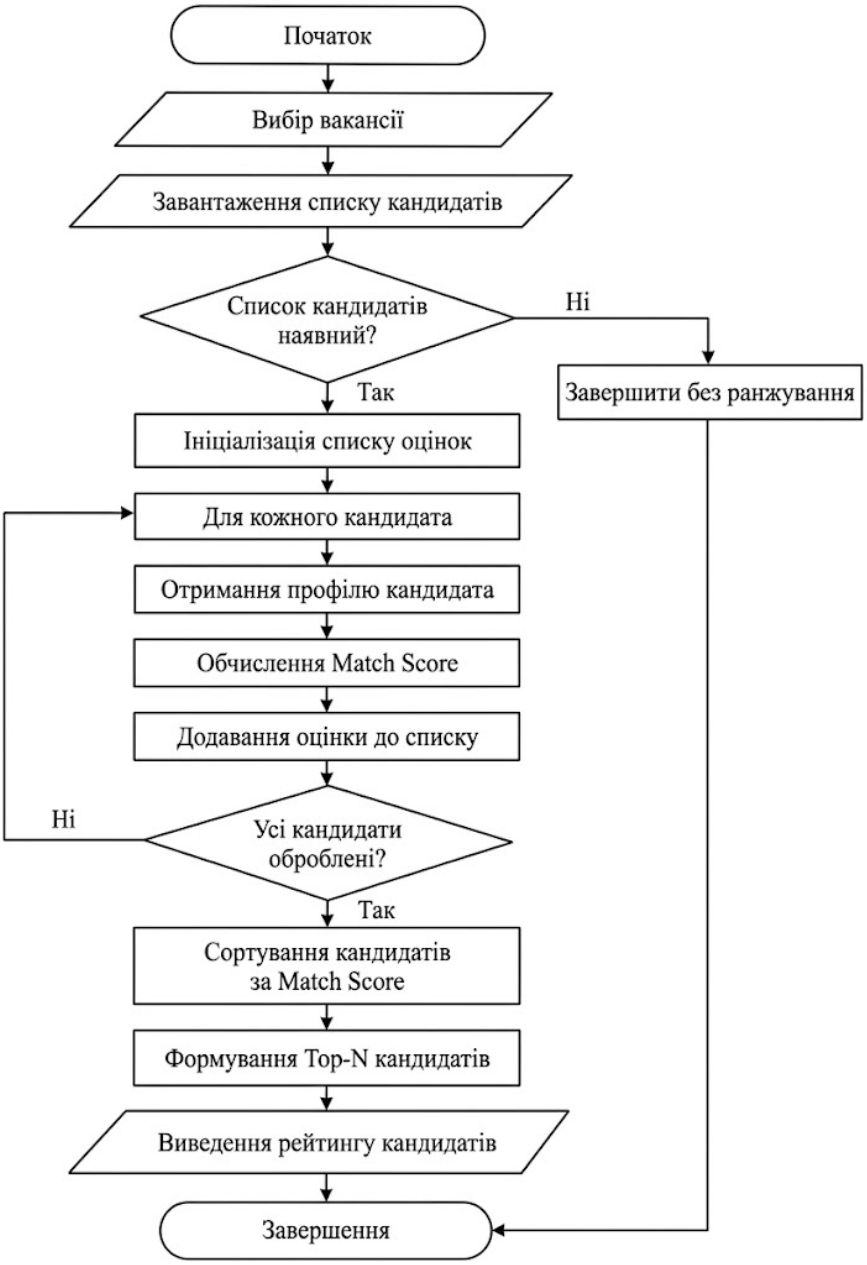


Рис. 2.12 Блок-схема алгоритму прийняття кадрового рішення

На першому етапі рекрутер отримує результати оцінювання кандидата та переглядає його профіль. Якщо кандидат не відповідає вимогам, заявка відхиляється, статус оновлюється, а кандидат отримує повідомлення. Якщо кандидат відповідає вимогам, результат передається менеджеру з найму. У разі погодження кандидат запрошується на наступний етап або приймається рішення про найм. Якщо рішення не погоджено, заявка повертається на доопрацювання або переводиться в резерв. Усі зміни фіксуються в базі даних і передаються до ATS/CRM.

2.8 Діаграма розгортання системи

Діаграма розгортання системи наведена на рисинку 2.13.

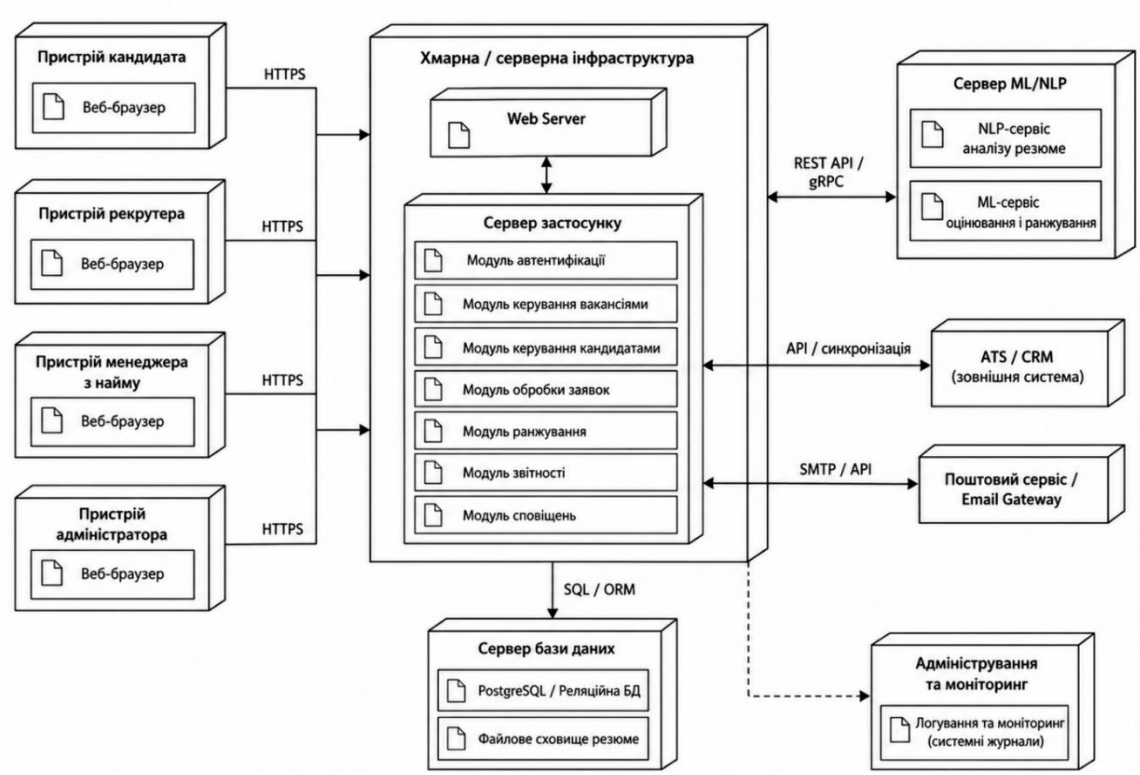


Рис. 2.13 Діаграма розгортання системи оптимізації рекрутингу

Діаграма розгортання відображає фізичну організацію програмного забезпечення. Користувачі взаємодіють із системою через веббраузер на пристроях кандидата, рекрутера, менеджера з найму та адміністратора. Передавання даних між клієнтськими пристроями і серверною інфраструктурою здійснюється через захищений протокол HTTPS.

У центральній частині розміщено хмарну або серверну інфраструктуру, яка містить Web Server і сервер застосунку. Web Server відповідає за приймання HTTP/HTTPS-запитів, віддавання статичних ресурсів і перенаправлення запитів до прикладного рівня. Сервер застосунку виконує основну бізнес-логіку системи:

автентифікацію, керування вакансіями, кандидатами, заявками, ранжуванням, звітністю і сповіщеннями.

Окремим вузлом виділено сервер ML/NLP, на якому розгортаються сервіси аналізу резюме, оцінювання та ранжування кандидатів. Обмін між сервером застосунку і ML/NLP-сервером здійснюється через REST API або gRPC. Такий поділ дозволяє масштабувати інтелектуальні модулі незалежно від основної прикладної частини.

Сервер бази даних містить реляційну базу PostgreSQL і файлове сховище резюме [20]. Доступ до даних здійснюється через SQL/ORM-рівень. Зовнішні сервіси ATS/CRM підключаються через API-синхронізацію, а поштовий сервіс або Email Gateway використовується для надсилання повідомлень кандидатам і користувачам системи [18]. Додатковий вузол адміністрування та моніторингу забезпечує журналювання подій, контроль працездатності системи та аналіз системних повідомлень.

2.9 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного та програмного забезпечення інтелектуальної системи оптимізації процесу рекрутингу. Побудовано UML-діаграми прецедентів, послідовності, активності, класів, кооперації, компонентів, пакетів і розгортання. Визначено основні ролі користувачів, функціональні сценарії, структуру класів, модульну архітектуру та фізичне розміщення системи. Окрему увагу приділено NLP/ML-модулю, який забезпечує аналіз резюме, формування профілю кандидата, обчислення Match Score і ранжування претендентів. Розроблені моделі є основою для реалізації програмного забезпечення, побудови бази даних, створення інтерфейсу користувача та проведення тестування.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів

Реалізація інтелектуальної системи оптимізації процесу рекрутингу виконана у вигляді веборієнтованого програмного забезпечення з клієнтським інтерфейсом, прикладною логікою, базою даних і модулем ML-аналізу. Такий підхід забезпечує доступність системи з різних пристроїв, централізоване зберігання інформації, зручне адміністрування та можливість масштабування окремих модулів.

Для клієнтської частини використано вебінтерфейс, який забезпечує взаємодію користувачів із системою через браузер. Інтерфейс реалізує сторінки входу, аналітичного огляду, бази кандидатів, ML-ранжування, інтерв'ю, звітів та адміністрування. Візуальне оформлення виконане у темному стилі з контрастними акцентами, що покращує сприйняття даних на аналітичних сторінках.

Прикладна логіка системи включає модулі автентифікації, керування кандидатами, вакансіями, заявками, ранжуванням, сповіщеннями та звітністю. Для зберігання даних використовується реляційна база даних. Вона забезпечує структуроване збереження користувачів, кандидатів, вакансій, заявок, результатів оцінювання, інтерв'ю, звітів і журналу дій. Окремий ML-модуль відповідає за обчислення Match Score, формування рейтингу кандидатів та пояснення результату.

Загальний склад програмних модулів наведено в таблиці 3.1.

Таблиця 3.1

Основні модулі програмного забезпечення

Модуль	Призначення
Модуль автентифікації	Вхід користувачів, перевірка ролей, контроль доступу
Модуль кандидатів	Додавання, пошук, перегляд і редагування профілів кандидатів
Модуль вакансій	Створення вакансій, зберігання вимог, контроль статусу
Модуль заявок	Прив’язка кандидата до вакансії, етапи розгляду, статуси
NLP/ML-модуль	Аналіз резюме, обчислення Match Score, ранжування
Модуль інтерв’ю	Планування співбесід і збереження результатів
Модуль звітності	Формування аналітичних звітів та експорт результатів

Наведені модулі утворюють єдину функціональну структуру, у якій кожна частина виконує окрему задачу, але взаємодіє з іншими через базу даних і прикладний рівень.

3.2 Реалізація інтерфейсу користувача

Інтерфейс системи побудовано за принципом розділення функцій відповідно до ролі користувача. Після входу до системи користувач отримує доступ лише до тих модулів, які відповідають його правам. Адміністратор має повний доступ до аналітики, кандидатів, вакансій, ML-ранжування, інтерв’ю, звітів та адміністрування. Рекрутер працює з кандидатами, вакансіями, рейтингами та інтерв’ю. Менеджер з найму переглядає рекомендації, пояснення та приймає кадрові рішення.

Початковим елементом роботи з програмою є форма входу та швидкої реєстрації, наведена на рисинку 3.1.

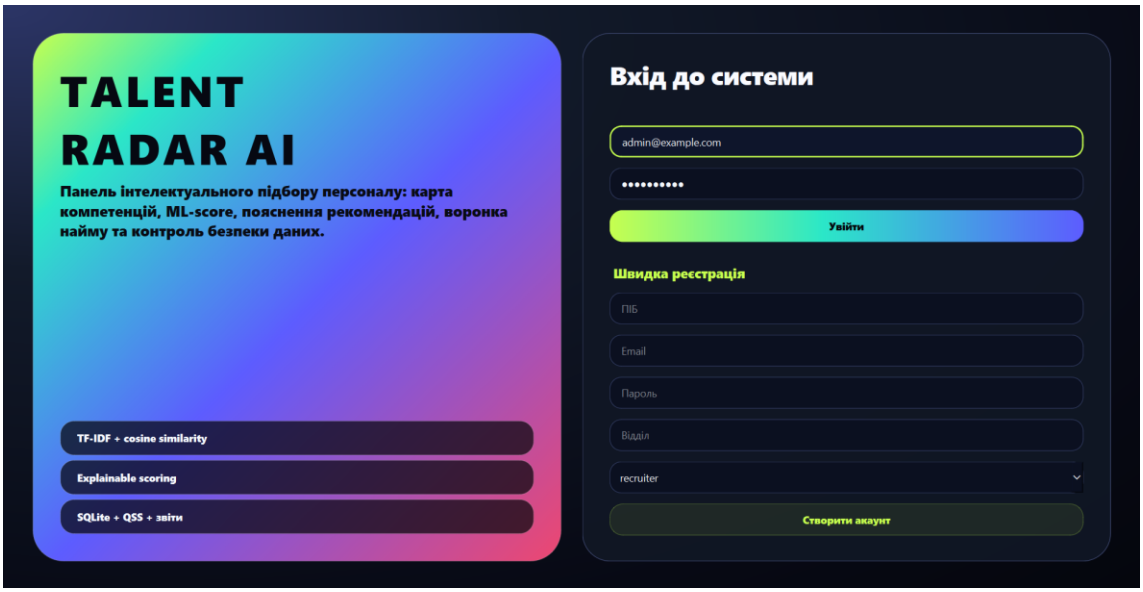


Рис. 3.1 Вікно входу та швидкої реєстрації користувача

Форма входу містить поля для введення email і пароля. Також передбачено блок швидкої реєстрації, у якому користувач може вказати ПІБ, email, пароль, відділ і роль. Це дозволяє створювати облікові записи рекрутерів, аналітиків або адміністраторів без окремого зовнішнього модуля. Після успішного входу система визначає роль користувача та відкриває відповідну навігаційну панель.

Аналітичний огляд системи наведено на рисинку 3.2.

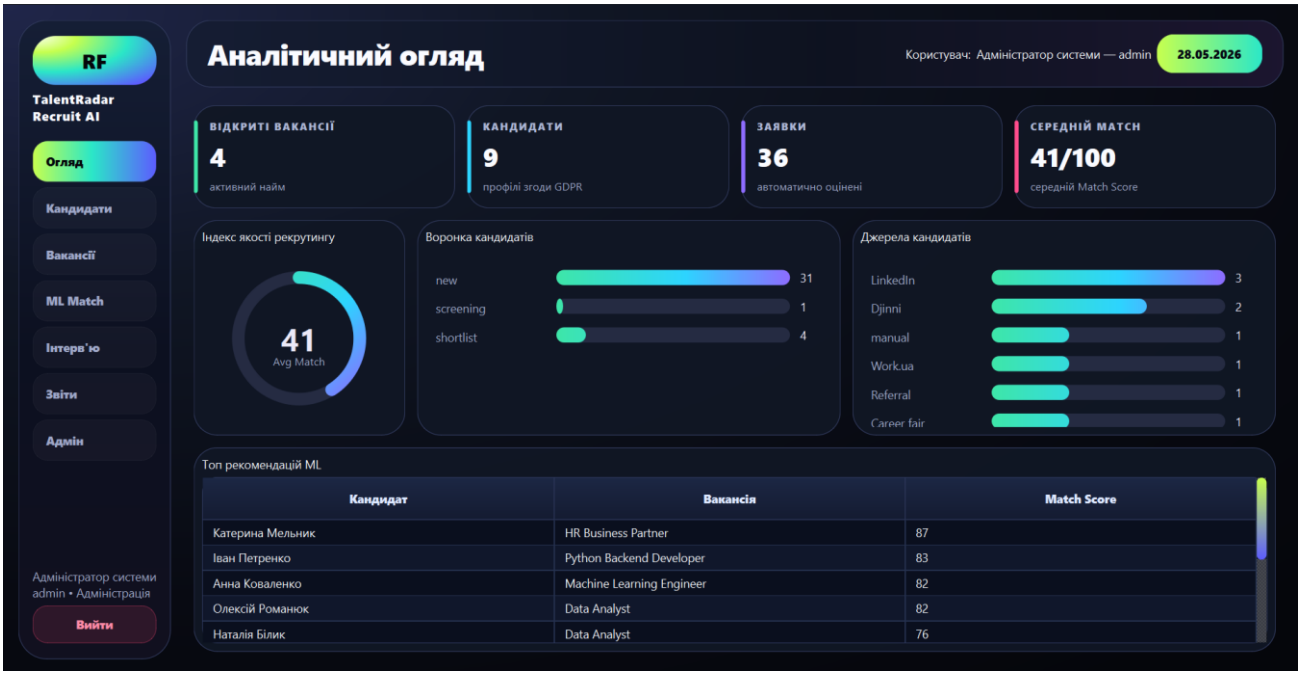


Рис. 3.2 Аналітичний огляд системи оптимізації рекрутингу

На сторінці аналітичного огляду відображаються ключові показники роботи системи: кількість відкритих вакансій, кількість кандидатів, кількість заявок і середній Match Score. Додатково подано індекс якості рекрутингу, воронку кандидатів, джерела кандидатів і таблицю топ-рекомендацій ML-модуля. Таке подання дозволяє швидко оцінити загальний стан процесу підбору персоналу та виявити напрями, які потребують уваги.

Сторінка бази кандидатів наведена на рисинку 3.3.

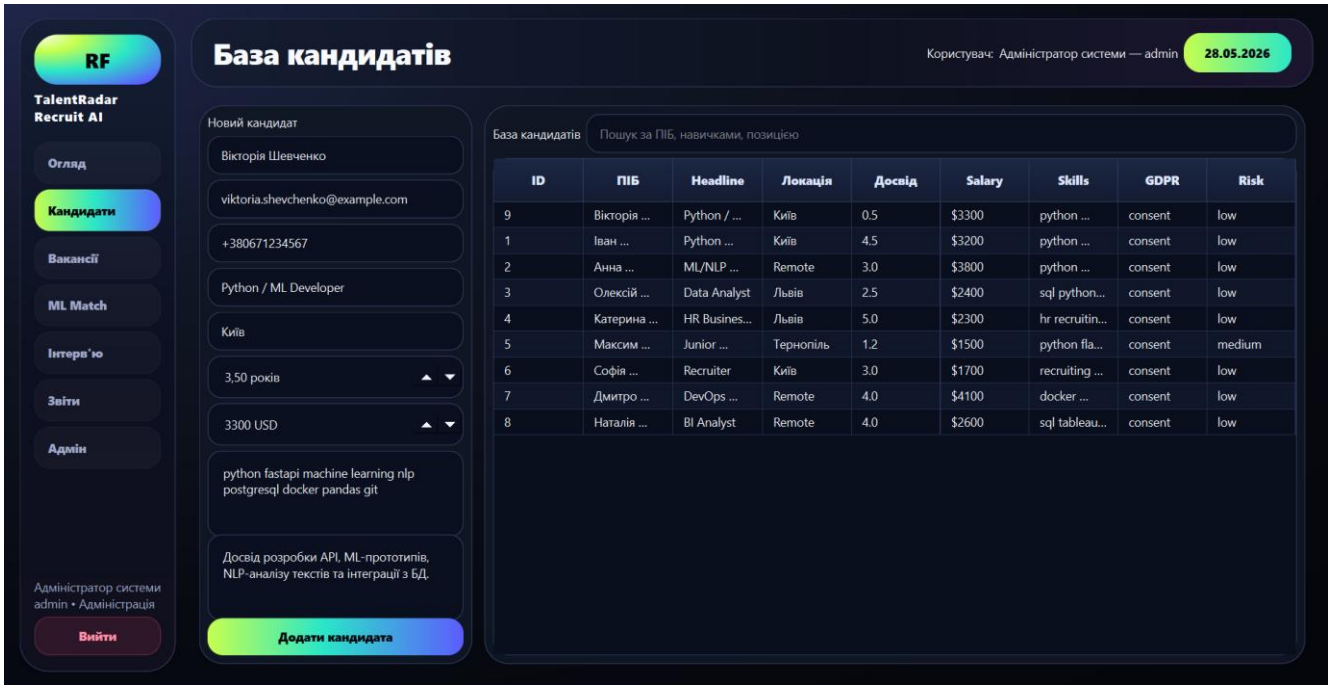


Рис. 3.3 Сторінка бази кандидатів

На цій сторінці реалізовано додавання нового кандидата, збереження контактних даних, професійного заголовка, локації, досвіду, зарплатних очікувань, навичок та опису досвіду. Праворуч від форми розміщена таблиця кандидатів із можливістю пошуку за ПІБ, навичками або позицією. У таблиці відображаються ідентифікатор кандидата, ПІБ, headline, локація, досвід, salary, skills, GDPR-статус і рівень ризику. Це забезпечує централізоване ведення кандидатської бази.

Сторінка ML-ранжування кандидатів наведена на рисинку 3.4.

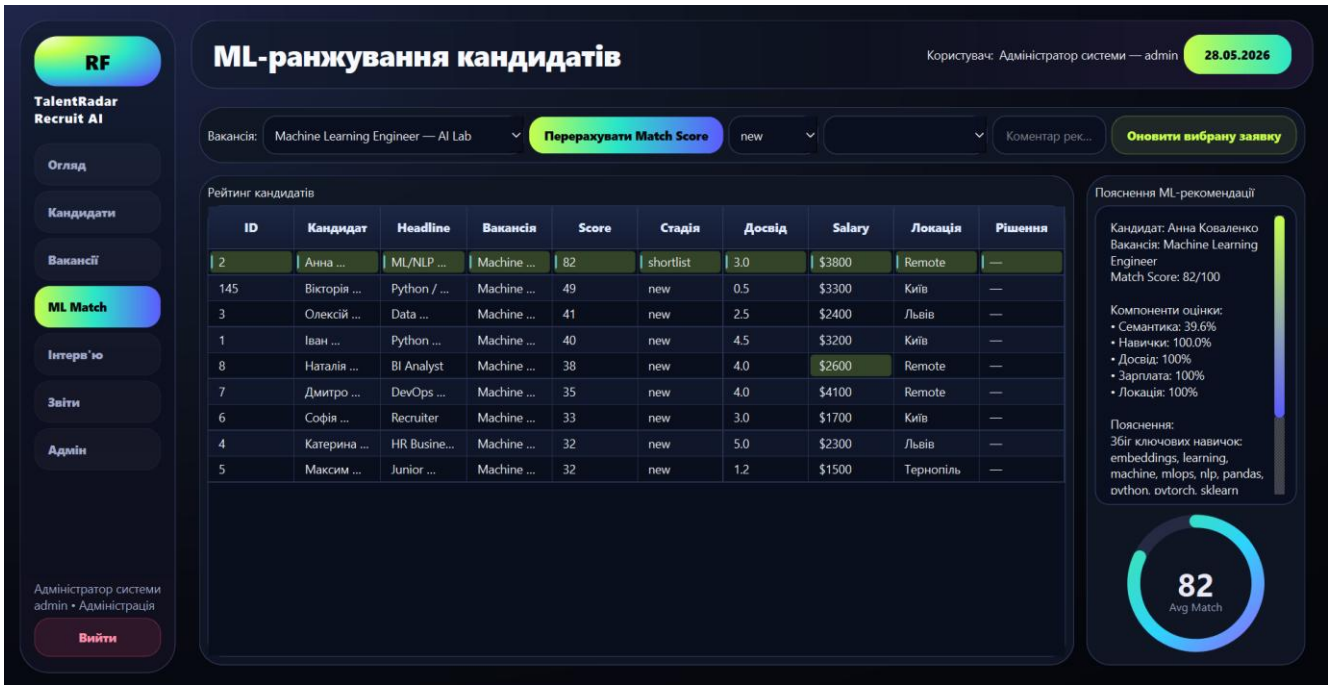


Рис. 3.4 Сторінка ML-ранжування кандидатів

У модулі ML-ранжування користувач обирає вакансію та запускає перерахунок Match Score. Система аналізує профілі кандидатів, порівнює їх із вимогами вакансії, формує рейтинг і підсвічує найкращого кандидата.

У правій частині сторінки відображається пояснення ML-рекомендації. Пояснення містить компоненти оцінки: семантичну відповідність, збіг навичок, досвід, зарплатні очікування та локацію. Це підвищує прозорість роботи моделі та дозволяє рекрутеру зрозуміти, чому кандидат отримав певну оцінку.

Сторінка планування інтерв’ю наведена на рисинку 3.5.

Модуль інтерв’ю дозволяє обрати заявку, вказати дату і час співбесіди, інтерв’юера, формат проведення та запланувати інтерв’ю. Нижче розміщено календар інтерв’ю, у якому наведено список запланованих співбесід із зазначенням кандидата, вакансії, дати, інтерв’юера, формату, статусу та feedback. Такий модуль забезпечує організацію наступних етапів відбору після автоматизованого ранжування.

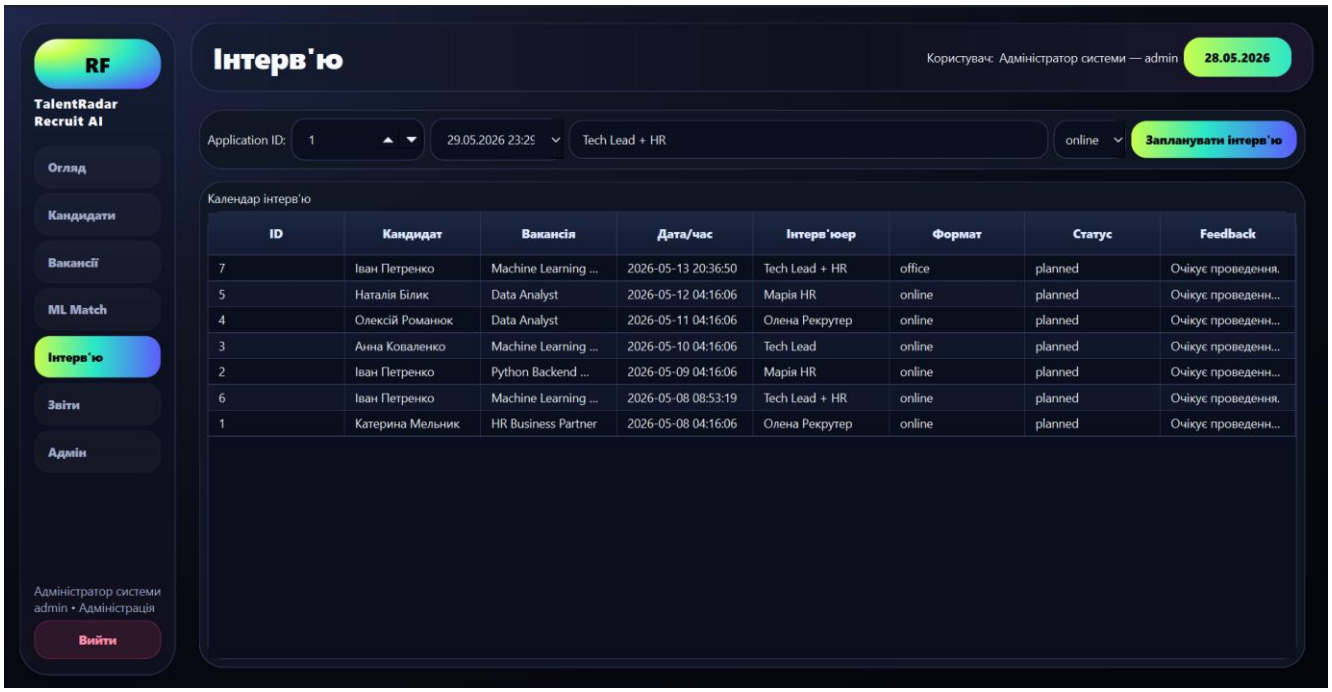


Рис. 3.5 Сторінка планування інтерв’ю

Сторінка звітів та експорту наведена на рисинку 3.6.

Рис. 3.6 Сторінка звітів та експорту результатів

На сторінці звітів реалізовано експорт даних у CSV та HTML, оновлення звіту, перегляд динаміки Match Score по топ-заявках і зведений рейтинг кандидатів. Графік дозволяє оцінити розподіл оцінок серед найкращих заявок, а таблиця рейтингу містить дані про кандидата, вакансію, стадію, score, локацію та прийняте рішення. Цей модуль використовується для аналітики та підготовки звітних матеріалів для рекрутера або керівника.

Сторінка адміністрування наведена на рисинку 3.7.

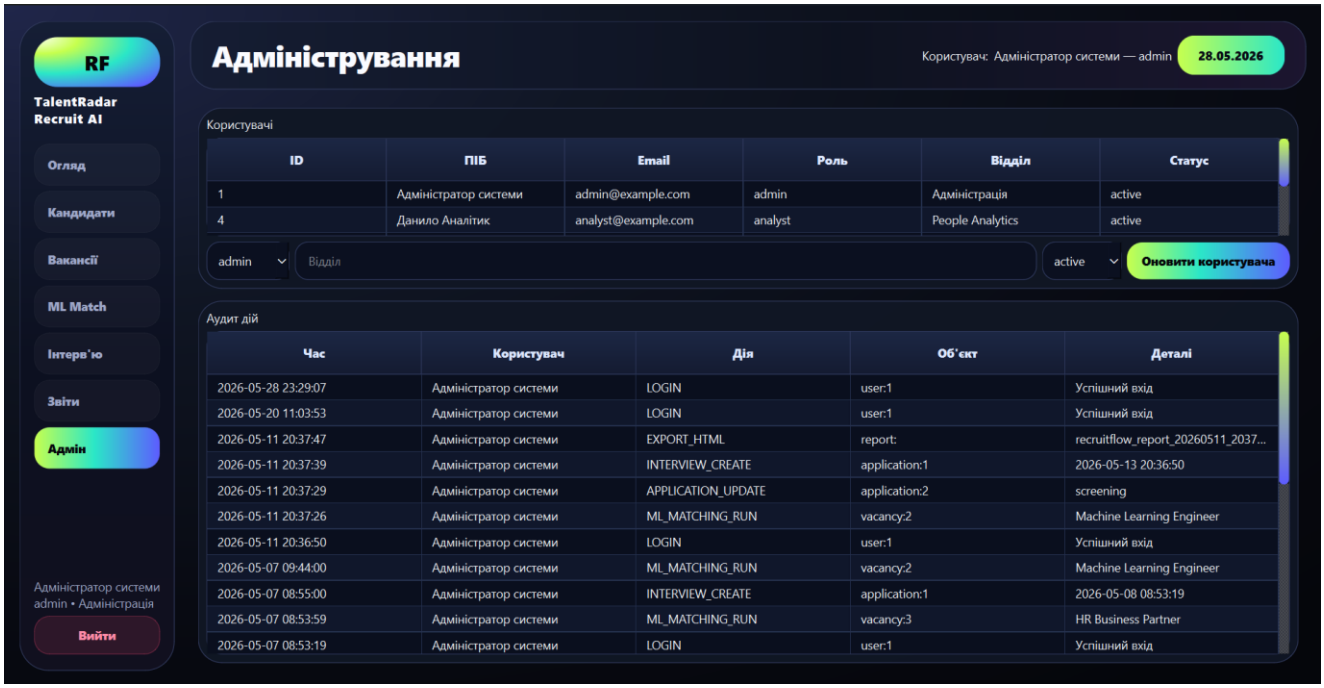


Рис. 3.7 Сторінка адміністрування користувачів і аудиту дій

Адміністративний модуль забезпечує керування користувачами, ролями, відділами та статусами облікових записів. Окремо реалізовано журнал аудиту дій, у якому зберігається час операції, користувач, дія, об’єкт і деталі. Такий підхід дає змогу контролювати вхід у систему, запуск ML-ранжування, оновлення даних і створення демонстраційних наборів.

3.3 Реалізація основних програмних функцій

Реалізована система підтримує повний цикл роботи з даними кандидата. Спочатку користувач створює або імпортує профіль кандидата. Після цього дані зберігаються в базі та можуть бути використані для порівняння з вакансіями. Кожен кандидат має набір характеристик: ім’я, email, контактний телефон, професійний заголовок, локацію, досвід, зарплатні очікування, навички, опис досвіду, GDPR-статус і рівень ризику.

Функція керування вакансіями дозволяє створювати опис посади, визначати вимоги до кандидата, потрібні навички та інші критерії. Ці дані використовуються ML-модулем під час обчислення Match Score. Оцінювання кандидата виконується за сукупністю ознак: збіг ключових навичок, семантична близькість профілю до опису вакансії, відповідність досвіду, зарплатних очікувань і локації.

Узагальнений алгоритм формування Match Score можна подати формулою (3.1):

$$\begin{aligned}\text{MatchScore} = {} & w_1 \cdot S_{\text{skills}} + w_2 \cdot S_{\text{semantic}} + w_3 \cdot \\ & S_{\text{experience}} + w_4 \cdot S_{\text{salary}} + w_5 \cdot S_{\text{location}},\end{aligned} \tag{3.1}$$

де S_{skills} – показник збігу навичок;

S_{semantic} – семантична близькість резюме та вакансії;

$S_{\text{experience}}$ – відповідність досвіду;

S_{salary} – відповідність зарплатних очікувань;

S_{location} – відповідність локації;

w_1, w_2, w_3, w_4, w_5 – вагові коефіцієнти.

У програмній реалізації результат нормалізується до шкали від 0 до 100 балів. Після цього система сортує кандидатів за спаданням оцінки та формує рейтинг. Найкращі кандидати потрапляють до Top-N списку, який використовується рекрутером під час прийняття рішення.

Для збереження результатів у системі використовується база даних, у якій виділено основні інформаційні сутності, наведені в таблиці 3.2.

Таблиця 3.2

Основні інформаційні сутності системи

Сутність	Призначення
User	Зберігання облікових записів і ролей користувачів
Candidate	Дані кандидатів, контактна інформація, професійний профіль
Vacancy	Опис вакансій, вимоги, статуси
Application	Заявки кандидатів на вакансії та етапи розгляду
MatchResult	Результати ML-оцінювання та ранжування
Interview	Заплановані інтерв'ю, формат, інтерв'юер, feedback
Report	Зведені аналітичні результати та експорт
AuditLog	Журнал дій користувачів і системних подій

Наведена структура забезпечує узгоджене зберігання даних і дозволяє відстежувати повний шлях кандидата: від створення профілю до фінального кадрового рішення.

3.4 Розгортання системи та склад інсталяційного пакета

Розгортання системи виконується за клієнт-серверною архітектурою. Користувачі працюють із програмним забезпеченням через веббраузер, а всі основні обчислення виконуються на сервері застосунку. Інтелектуальні операції NLP/ML-аналізу можуть бути винесені в окремий сервіс, що дозволяє незалежно масштабувати модуль оцінювання кандидатів.

Фізична структура розгортання відповідає діаграмі, наведеній на рисинку 2.13. Клієнтські пристрої підключаються до серверної інфраструктури через HTTPS. На сервері застосунку розміщуються модулі автентифікації, керування вакансіями, кандидатами, заявками, ранжуванням, звітністю і сповіщеннями. Сервер ML/NLP забезпечує аналіз резюме та розрахунок рейтингу кандидатів. Сервер бази даних зберігає реляційні дані й файли резюме. Для зовнішньої взаємодії використовуються ATS/CRM та поштовий сервіс.

Склад інсталяційного пакета наведено в таблиці 3.3.

Таблиця 3.3

Склад інсталяційного пакета системи

Елемент пакета	Призначення
app/	Основний код серверної частини
frontend/	Файли вебінтерфейсу
ml/	Модуль аналізу, ранжування та пояснення результатів
database/	Схема бази даних і початкові дані
reports/	Шаблони звітів та експортовані файли
logs/	Системні журнали й аудит дій
config/	Налаштування підключення до БД, ATS/CRM і пошти
README.md	Інструкція із запуску та використання

Після встановлення залежностей і налаштування конфігураційних параметрів система запускається локально або на сервері. Для тестового використання передбачено демонстраційні облікові записи адміністратора та рекрутера. Це дає змогу швидко перевірити роботу модулів без додаткового створення користувачів і початкових даних.

3.5 Висновки до третього розділу

У третьому розділі описано реалізацію програмного забезпечення інтелектуальної системи оптимізації рекрутингу. Розглянуто вибір технологічних засобів, структуру програмних модулів, реалізацію інтерфейсу користувача, роботу з кандидатами, ML-ранжування, планування інтерв’ю, звіти, експорт і адміністрування. Показано, що система забезпечує повний цикл обробки кандидатських даних: від створення профілю до формування рейтингу та прийняття кадрового рішення. Реалізований інтерфейс є функціональним, структурованим і орієнтованим на практичну роботу рекрутера, менеджера з найму та адміністратора.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів

Тестування інтелектуальної системи оптимізації процесу рекрутингу спрямоване на перевірку коректності роботи основних функціональних модулів, стабільності обробки кандидатських даних, правильності формування рейтингу та зручності користувацького інтерфейсу. Особлива увага приділяється модулям, які впливають на результат кадрового рішення: модулю кандидатів, модулю вакансій, модулю заявок, ML-ранжуванню, поясненню рекомендацій, звітності та адмініструванню.

План тестування охоплює функціональні, інтеграційні, інтерфейсні, продуктивні та безпекові перевірки. Функціональні тести визначають, чи правильно система виконує основні сценарії. Інтеграційні тести перевіряють взаємодію між модулями. Інтерфейсні тести оцінюють зручність роботи користувача. Продуктивні тести визначають швидкість обробки даних. Безпекові тести перевіряють контроль доступу та аудит дій.

План тестування наведено в таблиці 4.1.

Таблиця 4.1

План тестування програмних модулів

Модуль	Вид тестування	Очікуваний результат
Автентифікація	Функціональне, безпекове	Вхід лише для зареєстрованих користувачів, коректне визначення ролі
База кандидатів	Функціональне	Додавання, перегляд і пошук кандидатів працюють коректно
Вакансії	Функціональне	Вакансії створюються, редагуються та використовуються в аналізі

Продовження таблиці 4.1

План тестування програмних модулів

Модуль	Вид тестування	Очікуваний результат
Заявки	Інтеграційне	Кандидат коректно зв’язується з вакансією
ML-ранжування	Функціональне, продуктивне	Match Score розраховується, кандидати сортуються за рейтингом
XAI-пояснення	Функціональне	Відображаються фактори, які вплинули на рекомендацію
Інтерв’ю	Функціональне	Інтерв’ю створюється і відображається в календарі
Звіти	Функціональне	Формується таблиця рейтингу та графік динаміки

Для кожного модуля визначено вхідні дані, очікуваний результат і критерій успішності. Тест вважається пройденим, якщо фактичний результат відповідає очікуваному, а система не формує помилок виконання.

4.2 Тестування інтерфейсних модулів системи

Тестування інтерфейсу виконувалося за основними сценаріями використання системи. Першим перевірено сценарій входу та реєстрації користувача. У формі входу перевірялися коректність введення email і пароля, відображення активного поля, робота кнопки входу та створення нового облікового запису. Результат тестування підтвердив, що форма входу коректно приймає дані та забезпечує перехід до основного інтерфейсу відповідно до ролі користувача.

На сторінці аналітичного огляду перевірено відображення кількісних показників, діаграми якості рекрутингу, воронки кандидатів, джерел кандидатів і таблиці топ-рекомендацій. Дані відображаються узгоджено: кількість кандидатів, заявок і середній Match Score відповідають поточному набору демонстраційних даних. Це підтверджує коректність роботи модуля агрегованої аналітики.

Під час тестування сторінки бази кандидатів перевірено додавання нового кандидата, заповнення полів, пошук і відображення записів у таблиці. Система коректно приймає ПІБ, email, телефон, headline, локацію, досвід, salary, skills і опис професійного досвіду. Після додавання кандидата дані потрапляють до таблиці та можуть бути використані під час ML-ранжування.

У модулі ML-ранжування перевірено вибір вакансії, запуск перерахунку Match Score, сортування кандидатів і відображення пояснення рекомендації. Система правильно визначає кандидата з найвищим показником відповідності та показує складові оцінки. Наприклад, для кандидата з високим збігом навичок і відповідною семантичною близькістю система формує вищу позицію в рейтингу. Наявність пояснення ML-рекомендації дозволяє перевірити не лише числовий результат, а й логіку його формування.

На сторінці інтерв’ю протестовано створення запису про співбесіду. Перевірялися вибір application ID, дати, інтерв’юера, формату проведення та статусу. Після додавання запис відображається у календарі інтерв’ю. Це підтверджує працездатність модуля планування наступних етапів відбору.

На сторінці звітів перевірено формування зведеного рейтингу, побудову графіка динаміки Match Score і роботу кнопок експорту CSV та HTML. Система виводить рейтинг кандидатів у табличному вигляді, що дозволяє використовувати результати для подальшого аналізу або включення у звітні матеріали.

Адміністративний модуль протестовано шляхом перегляду користувачів, оновлення ролі, статусу та аналізу журналу аудиту. У журналі фіксуються події входу, запуску ML-ранжування та створення демонстраційних даних. Це підтверджує наявність базового контролю дій користувачів.

Узагальнені результати тестування інтерфейсу наведено в таблиці 4.2.

Таблиця 4.2

Результати тестування інтерфейсних модулів

Сторінка	Перевірені дії	Результат
Вхід до системи	Введення email, пароля, швидка реєстрація	Працює коректно
Аналітичний огляд	Відображення KPI, графіків, таблиці рекомендацій	Працює коректно
База кандидатів	Додавання кандидата, пошук, перегляд таблиці	Працює коректно
ML-ранжування	Перерахунок Match Score, пояснення, рейтинг	Працює коректно
Інтерв’ю	Планування співбесіди, перегляд календаря	Працює коректно
Звіти та експорт	Формування графіка, таблиці, експорт	Працює коректно
Адміністрування	Перегляд користувачів і журналу аудиту	Працює коректно

Проведене тестування показало, що інтерфейсні модулі працюють узгоджено та забезпечують виконання основних сценаріїв користувача.

4.3 Оцінювання результатів тестування та ефективності системи

Оцінювання ефективності системи виконано за критеріями функціональної повноти, коректності обробки даних, зручності інтерфейсу, прозорості рекомендацій і готовності до практичного використання. Функціональна повнота визначається наявністю основних модулів: кандидатів, вакансій, заявок, ML-ранжування, інтерв’ю, звітності й адміністрування. Коректність обробки даних підтверджена тестами додавання кандидатів, формування рейтингу та збереження результатів.

Одним із важливих показників ефективності є автоматизація первинного скринінгу. У традиційному підході рекрутер вручну переглядає резюме, порівнює навички, досвід і вимоги вакансії. У розробленій системі ці дії частково автоматизовані: система формує Match Score, сортує кандидатів і показує пояснення рекомендації. Це зменшує обсяг ручної роботи та дозволяє рекрутеру зосередитися на якісній перевірці найрелевантніших кандидатів.

Порівняльна характеристика ручного та автоматизованого підходу наведена в таблиці 4.3.

Таблиця 4.3

Порівняння ручного та автоматизованого підбору кандидатів

Критерій	Ручний підхід	Розроблена система
Аналіз резюме	Виконується рекрутером вручну	Частково автоматизований
Порівняння з вакансією	Суб’єктивне	На основі Match Score
Ранжування кандидатів	Формується вручну	Формується автоматично
Пояснення рішення	Залежить від рекрутера	Відображається XAI-пояснення
Звітність	Потребує ручного оформлення	Формується в системі
Аудит дій	Обмежений	Фіксується у журналі

Результати тестування підтверджують, що система забезпечує зручну централізацію рекрутингових даних, підтримує автоматизоване ранжування кандидатів і формує аналітичні матеріали. Найбільш важливим результатом є наявність пояснюваності ML-рекомендації, оскільки рекрутер бачить не лише підсумковий бал, а й фактори, які вплинули на оцінку.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано тестування інтелектуальної системи оптимізації процесу рекрутингу. Перевірено роботу модулів автентифікації, кандидатів, вакансій, заявок, ML-ранжування, інтерв'ю, звітності та адміністрування. Результати тестування показали, що система коректно виконує основні функціональні сценарії, забезпечує збереження даних, формує рейтинг кандидатів і відображає пояснення ML-рекомендацій.

Інтерфейсні тести підтвердили зручність роботи зі сторінками входу, аналічного огляду, бази кандидатів, ML-ранжування, інтерв'ю, звітів та адміністрування. Система дозволяє централізовано керувати кандидатами, вакансіями, заявками і результатами оцінювання. Завдяки використанню Match Score та пояснюваного аналізу підвищується прозорість кадрових рішень і зменшується обсяг ручної роботи рекрутера.

Отже, розроблене програмне забезпечення відповідає поставленим вимогам і може використовуватися як функціональний прототип інтелектуальної системи підтримки рекрутингу.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено інтелектуальну систему оптимізації процесу рекрутингу з використанням машинного навчання. Розроблена система призначена для автоматизації обробки кандидатських даних, аналізу резюме, порівняння профілю кандидата з вимогами вакансії, формування рейтингу претендентів, пояснення результатів оцінювання та підтримки прийняття кадрових рішень.

У першому розділі було досліджено предметну область рекрутингу та визначено основні проблеми традиційного підходу до підбору персоналу. Встановлено, що ручний аналіз резюме потребує значних часових витрат, залежить від суб'єктивної оцінки рекрутера та не забезпечує достатньої масштабованості під час роботи з великою кількістю кандидатів. Проведено аналіз сучасних підходів до автоматизації рекрутингу, зокрема rule-based систем, статистичних методів, систем підтримки прийняття рішень, NLP-підходів, моделей машинного навчання та гібридних інтелектуальних рішень. Також проаналізовано існуючі програмні продукти, серед яких LinkedIn Talent Insights, Workday Recruiting, BambooHR ATS, Manatal та Oracle Taleo Recruiting Cloud. На основі аналізу сформовано висновок, що доцільним є створення системи, яка поєднує автоматизоване ранжування кандидатів, NLP-аналіз резюме, пояснення результатів і аналітичну підтримку рекрутера.

У роботі було сформульовано функціональні, нефункціональні, технічні та безпекові вимоги до системи. Визначено, що програмне забезпечення має забезпечувати завантаження та обробку резюме, створення вакансій, формування структурованого профілю кандидата, обчислення показника відповідності Match Score, ранжування кандидатів, формування аналітичних звітів, ведення журналу дій та інтеграцію із зовнішніми HR-системами. Окрему увагу приділено вимогам захисту персональних даних, рольовому доступу, аудиту дій користувачів і прозорості роботи інтелектуальних модулів.

У другому розділі виконано проєктування інформаційного та програмного забезпечення системи. Побудовано UML-діаграми прецедентів, послідовності, активності, класів, кооперації, компонентів, пакетів і розгортання. Діаграма прецедентів дозволила визначити основні ролі користувачів і функції системи. Діаграма послідовності описала порядок взаємодії між кандидатом, рекрутером, вебінтерфейсом, NLP-модулем, ML-модулем, базою даних і зовнішньою ATS/CRM. Діаграма активності відобразила повний процес обробки кандидатської заявки від завантаження резюме до прийняття кадрового рішення.

Під час проєктування статичної структури системи було визначено основні класи: користувач, кандидат, рекрутер, менеджер з найму, адміністратор, резюме, вакансія, заявка, профіль кандидата, навичка, оцінювання, звіт, сповіщення, NLP-модуль, ML-модуль та ATS/CRM-інтеграція. Побудовані діаграми кооперації деталізували ключові сценарії роботи: автоматизоване оцінювання кандидата, ранжування кандидатів за вакансією та прийняття кадрового рішення. Компонентна і пакетна діаграми дали змогу визначити модульну структуру системи, а діаграма розгортання показала фізичну організацію програмного забезпечення в серверному середовищі.

У третьому розділі було реалізовано програмне забезпечення інтелектуальної системи оптимізації рекрутингу. Розроблена система містить модулі входу та реєстрації, аналітичного огляду, бази кандидатів, ML-ранжування, інтерв’ю, звітів та адміністрування. Інтерфейс системи побудовано у вигляді веборієнтованого застосунку з темним візуальним оформленням, навігаційною панеллю та окремими сторінками для основних функціональних сценаріїв.

Реалізований модуль кандидатів забезпечує створення та перегляд профілів кандидатів, збереження контактних даних, професійного заголовка, локації, досвіду, зарплатних очікувань, навичок і короткого опису досвіду. Модуль ML-ранжування дозволяє обрати вакансію, перерахувати Match Score, сформувати рейтинг кандидатів і переглянути пояснення рекомендації. Пояснення містить складові оцінювання, зокрема семантичну відповідність, збіг навичок, досвід, зарплатні очікування та локацію. Це підвищує прозорість роботи системи та

дозволяє рекрутеру не лише бачити числовий бал, а й розуміти причини сформованої рекомендації.

У системі також реалізовано модуль планування інтерв'ю, який дає змогу створювати записи про співбесіди, зазначати дату, формат, інтерв'юера та статус. Модуль звітів забезпечує перегляд динаміки Match Score, формування зведеного рейтингу кандидатів і експорт результатів. Адміністративний модуль дозволяє переглядати користувачів, змінювати ролі та статуси, а також контролювати журнал аудиту дій. Наявність аудиту є важливою для контролю використання системи та фіксації ключових операцій.

У четвертому розділі проведено тестування розробленої системи. Було перевірено роботу модулів автентифікації, бази кандидатів, ML-ранжування, інтерв'ю, звітів, експорту та адміністрування. Тестування підтвердило, що система коректно виконує основні функціональні сценарії: вхід користувача, додавання кандидата, перегляд аналітики, запуск ранжування, формування пояснення ML-рекомендації, планування інтерв'ю, створення звітів і ведення журналу дій.

Результати тестування показали, що розроблена система забезпечує централізоване зберігання рекрутингових даних, прискорює первинний аналіз кандидатів, автоматизує ранжування претендентів і надає рекрутеру інструменти для прийняття обґрунтованих кадрових рішень. У порівнянні з ручним підходом система дозволяє зменшити трудомісткість первинного скринінгу, підвищити структурованість роботи з кандидатами та забезпечити прозоріше оцінювання за рахунок використання Match Score і пояснюваних рекомендацій.

Практичне значення роботи полягає в тому, що розроблене програмне забезпечення може використовуватися як функціональний прототип інтелектуальної системи підтримки рекрутингу. Воно може бути застосоване для автоматизації процесу попереднього відбору кандидатів, ведення бази кандидатів, аналізу відповідності вакансіям, формування рейтингів, планування інтерв'ю та підготовки аналітичних звітів. Система також може бути розширена шляхом підключення реальних ATS/CRM-сервісів, удосконалення моделей машинного

навчання, додавання багатомовної обробки резюме та впровадження складніших механізмів пояснення рішень.

Отже, мету кваліфікаційної роботи досягнуто. У результаті виконання роботи було проаналізовано предметну область, сформовано вимоги, спроєктовано архітектуру, розроблено програмне забезпечення, реалізовано основні функціональні модулі та проведено тестування системи. Розроблена інтелектуальна система оптимізації процесу рекрутингу з використанням машинного навчання відповідає поставленим вимогам і забезпечує автоматизовану підтримку рекрутера під час аналізу, оцінювання та відбору кандидатів.

Робота пройшла апробацію. За результатами дослідження опубліковано такі тези доповідей:

1. Дудник М. Є. Шахматов І. О. Інтелектуальна мультимодальна система аналізу кандидатів для вакансій на основі LLM-технологій та векторних індексів. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.279-283.

2. Дудник М. Є. Довженко Т.П. Забезпечення безпеки даних в інтелектуальній системі оптимізації процесу рекрутингу. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.278-280.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jantan H., Hamdan A. R., Othman Z. Intelligent decision support system for the recruitment and selection process using machine learning [Електронний ресурс] // International Journal of Computer Science and Network Security. – 2010. – Т. 10, № 3. – С. 28–34. – Режим доступу: https://www.ijcsns.org (дата звернення: 01.10.2025).

2. Saaty T. L. Decision making with the analytic hierarchy process [Електронний ресурс] // International Journal of Services Sciences. – 2008. – Т. 1, № 1. – С. 83–98. – Режим доступу: https://doi.org/10.1504/IJSSCI.2008.017590(дата звернення: 01.10.2025).

3. Reimers N., Gurevych I. Sentence-BERT: Sentence embeddings using Siamese BERT-Networks [Електронний ресурс] // arXiv preprint arXiv:1908.10084. – 2019. – Режим доступу: https://arxiv.org/abs/1908.10084 (дата звернення: 05.10.2025).

4. Robertson S. E., Zaragoza H. The Probabilistic Relevance Framework: BM25 and beyond [Електронний ресурс] // Foundations and Trends in Information Retrieval. – 2009. – Т. 3, № 4. – С. 333–389. – Режим доступу: https://doi.org/10.1561/1500000019 (дата звернення: 5.10.2025).

5. Al-Qallaf A., et al. Machine Learning Models for Resume Classification and Ranking: A Comparative Study. – IEEE Access, 2022. – Vol. 10. – P. 126381–126394. – Режим доступу: https://doi.org/10.1109/ACCESS.2022.3221043(дата звернення: 7.10.2025).

6. Intelligent Resume Screening System Using NLP and Machine Learning. – International Journal of Innovative Science and Research Technology (IJISRT), 2025. – Режим доступу: https://eprint.ijisrt.org/ (дата звернення: 8.10.2025).

7. Zimmermann A., et al. Recommending Candidates in E-Recruitment: An Information Retrieval Approach. – arXiv preprint, 2016. – Режим доступу: https://arxiv.org/abs/1606.08031 (дата звернення: 9.10.2025).

8. Kumar V., Singh R., Gupta D. Automated Resume Classification and Similarity Detection Using NLP and Machine Learning Techniques. – IJISRT, 2020. – Режим доступу: https://ijisrt.com (дата звернення: 9-.10.2025).

9. Receiver Operating Characteristic (ROC) Curve Examples. – Scikit-learn Official Documentation, 2024. – Режим доступу: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (дата звернення: 11.10.2025).

10. Wang S., et al. An Intelligent Resume Screening System Based on NLP and Deep Learning. – Expert Systems with Applications. – 2024. – Vol. 237. – P. 121715. – Режим доступу: https://doi.org/10.1016/j.eswa.2024.121715 (дата звернення: 12.10.2025).

11. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. – arXiv preprint, 2019. – Режим доступу: https://arxiv.org/abs/1810.04805 (дата звернення: 15.10.2025).

12. Lundberg S., Lee S.-I. A Unified Approach to Interpreting Model Predictions (SHAP). – Advances in Neural Information Processing Systems (NIPS), 2017. – Режим доступу: https://arxiv.org/abs/1705.07874 (дата звернення: 15.10.2025).

13. Li Y., Huang J., et al. Multimodal Large Language Models for Recruitment Data Understanding. – IEEE Access,2024. – Vol. 12. – P. 98856–98872. – Режим доступу: https://doi.org/10.1109/ACCESS.2024.9885671 (дата звернення: 16.10.2025).

14. Kumar A., Zhou T. Reducing Hallucinations in Large Language Models via Retrieval and Validation Pipelines. – arXiv preprint, 2025. – Режим доступу: https://arxiv.org/abs/2503.01892 (дата звернення: 17.10.2025).

15. Chen L., Zhao H. *Comparative Study of ATS, ML and LLM-based Recruitment Systems. – Information Systems Frontiers,* 2024. – Vol. 26, No. 3. – P. 455–472. – Режим доступу: https://doi.org/10.1007/s10796-024-10452-z (дата звернення: 23.10.2025).

16. Zhang Q., et al. *Towards Explainable Recruitment AI: Evaluation of NLP and LLM Models in Hiring Pipelines. – arXiv preprint*, 2025. – Режим доступу: https://arxiv.org/abs/2502.04122 (дата звернення: 30.10.2025).

17. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2021. Vol. 12. P. 2825–2830. URL: https://scikit-learn.org

18. FastAPI Official Documentation. Advanced Features and Resource Management. 2025. URL: https://fastapi.tiangolo.com (date of access: 15.02.2026).

19. Bird S., Klein E., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. 2nd ed. Sebastopol: O'Reilly Media, 2024. 412 p.

20. PostgreSQL Global Development Group. PostgreSQL 16.0 Documentation: Relational Database Systems and SQL Standards. Structured Query Publishing, 2023. 1250 p.

21. Mitrofanova A., Siniak V. Application of Tokenization and Cosine Similarity in Modern HR Semantic Matching Systems. International Journal of Computer Science and Information Technologies. 2025. Vol. 16, no. 2. P. 89–97. URL: https://doi.org/10.20998/ijcsit.2025.02.11

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система оптимізації процесу рекрутингу з використанням методів машинного навчання

Виконала студентка 4 курсу
групи ПД-42
Дудник Марія Євгенівна
Керівник роботи
канд.техн.наук, Довженко Тимур Павлович

Київ – 2026

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращення ефективності підбору персоналу шляхом впровадження інтелектуальної системи оптимізації процесу рекрутингу.

Об’єкт дослідження: процес рекрутингу як цілісна інформаційно-керуюча система.

Предмет дослідження: методи та алгоритми машинного навчання, моделі оброблення резюме й вакансій, а також програмні засоби оптимізації рекрутингового процесу на основі аналітики даних.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної області, дослідити процеси підбору кандидатів та виявити ключові проблеми автоматизації обробки резюме.
2. Провести порівняльний аналіз існуючих програмних рішень на ринку, що дозволить визначити їхні недоліки та обґрунтувати доцільність створення нової системи.
3. Визначити та сформулювати функціональні та нефункціональні вимоги, що дозволить встановити технічні характеристики та критерії успішності готового продукту.
4. Провести аналіз та обґрунтувати вибір технологічного стеку та моделей машинного навчання, що забезпечить високу ефективність та швидкість обробки даних.
5. Спроєктувати та реалізувати архітектуру застосунку, структуру бази даних, інтелектуальні модулі оцінювання кандидатів та інтерфейс користувача з розмежуванням прав доступу.
6. Провести тестування розробленого програмного забезпечення для перевірки коректності роботи інтерфейсних модулів, точності алгоритмів.

3

АНАЛІЗ АНАЛОГІВ

Критерій	LinkedIn Talent Insights	Workday Recruiting	BambooHR ATS	Manatal	Oracle Taleo Recruiting	TalentRadar Recruit AI
Тип системи	Аналітична HR-платформа	Корпоративна HR-система	ATS-система	Інтелектуальна рекрутинг-система	Інтелектуальна система	Інтелектуальна система
Автоматичний аналіз кандидатів	+	+	±[1]	+	+	+
Машинне навчання	±[2]	±[3]	–	+	+	+
NLP-аналіз резюме	–	–	–	+	+	+
Ранжування кандидатів	±[4]	+	+	+	+	+
Аналітичні звіти	+	+	±[5]	+	+	+
Простота використання	Середня	Низька	Висока	Середня	Середня	Висока
Масштабованість	Висока	Висока	Середня	Середня	Середня	Висока
Гнучкість налаштування	Середня	Висока	Середня	Висока	Висока	Висока

±[1]: базова (картки та статуси); система не вміє самостійно читати й аналізувати текст у файлах резюме.
±[2]: Алгоритми підбирають кандидатів лише всередині своєї соцмережі, але не вміють працювати із завантаженою базою.
±[3]: Штучний інтелект націлений на загальну статистику компанії, а не на глибокий аналіз навичок у резюме.
±[4]: Сортує лише відкриті профілі користувачів мережі; автоматичний аналіз звичайних PDF- або Word-файлів не підтримується.
±[5]: Показує лише просту статистику; автоматичного розрахунку рейтингу (Match Score) немає.

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

1. Система повинна приймати резюме у форматах PDF, DOCX, HTML і виконувати їх автоматизоване парсування.
2. Модуль повинен виконувати видобування ключових компетенцій, вимог, досвіду та умов праці з описів вакансій.
3. Система повинна здійснювати токенізацію, лематизацію, NER (розпізнавання іменованих сутностей), нормалізацію навичок та POS-аналіз.
4. На основі витягнутих сутностей система формує уніфікований структурований профіль.
5. Модель машинного навчання повинна обчислювати *match-score* між резюме та вакансією.
6. Система повинна будувати топ-N список кандидатів за інтегральним індексом відповідності.
7. Система повинна пропонувати альтернативні вакансії кандидатам або альтернативних кандидатів рекрутеру.
8. Платформа повинна візуалізувати статистику найму, ефективність моделей та метрики відбору.
9. Усі операції користувача повинні логуватися з мітками часу для подальшого аудиту.
10. Система повинна підтримувати REST API для зв'язку з ATS/CRM-платформами.

Нефункціональні вимоги до системи

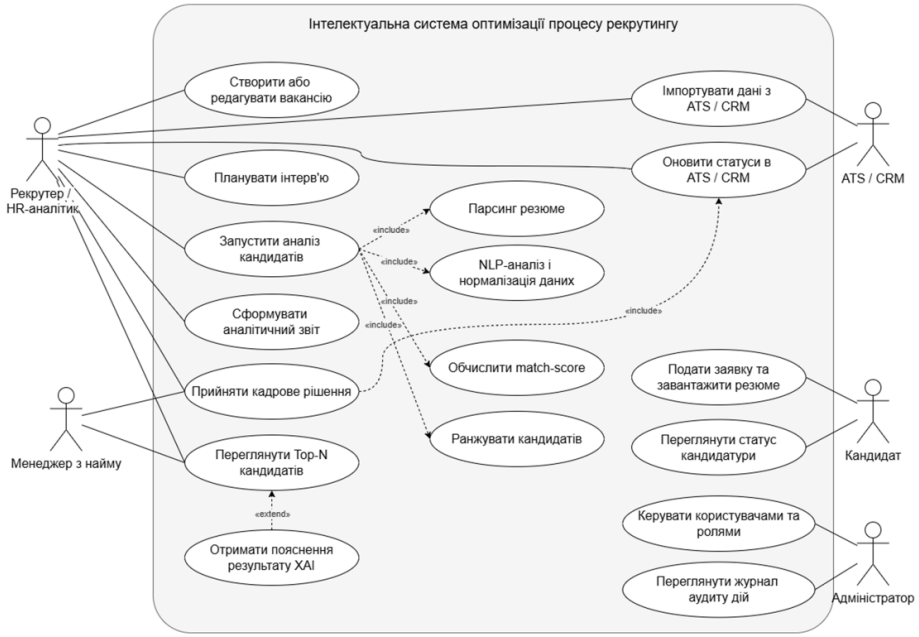
1. Час обробки одного резюме не повинен перевищувати 1 секунду при навантаженні до 50 одночасних користувачів.
2. Мінімальний коефіцієнт F1 для класифікації релевантності – 0.85.
3. Система повинна підтримувати горизонтальне масштабування сервісів ML/NLP.
4. Гарантована доступність сервісу – не менше 99.5%.
5. Повна підтримка PDF, DOCX, RTF, HTML +автономний fallback OCR (оптичне розпізнавання тексту).
6. Інтерфейс повинен забезпечувати час реакції менше 200 мс.
7. Підтримка української та англійської мов обробки текстів.
8. Система повинна вести структурований журнал усіх операцій ML-модулів.
9. Платформа має працювати у Docker-середовищі та підтримувати Kubernetes.
10. Моделі повинні проходити періодичне перенавчання не рідше одного разу на місяць.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

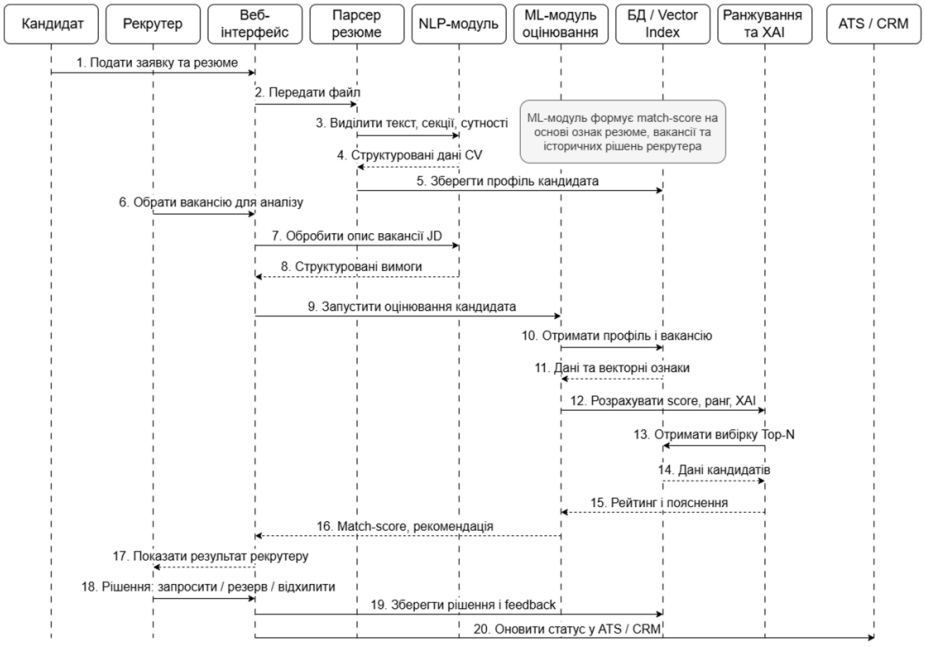
6

ДІАГРАМА ПРЕЦЕДЕНТІВ



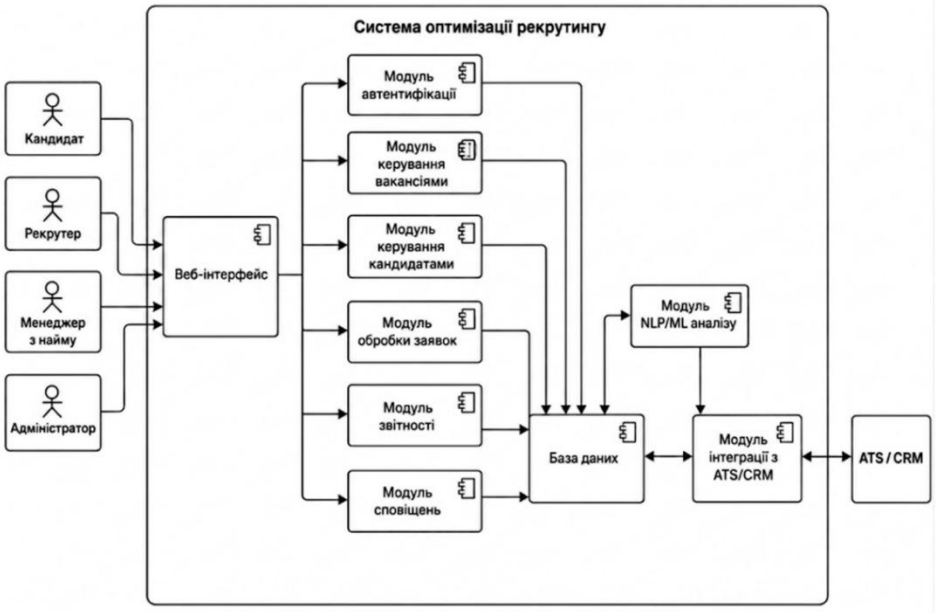
7

ДІАГРАМА ПОСЛІДОВНОСТІ



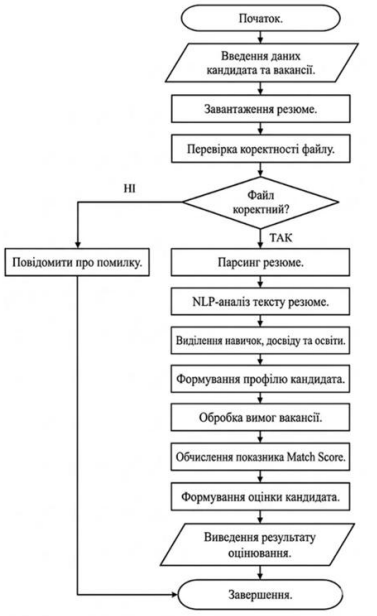
8

ДІАГРАМА КОМПОНЕНТІВ

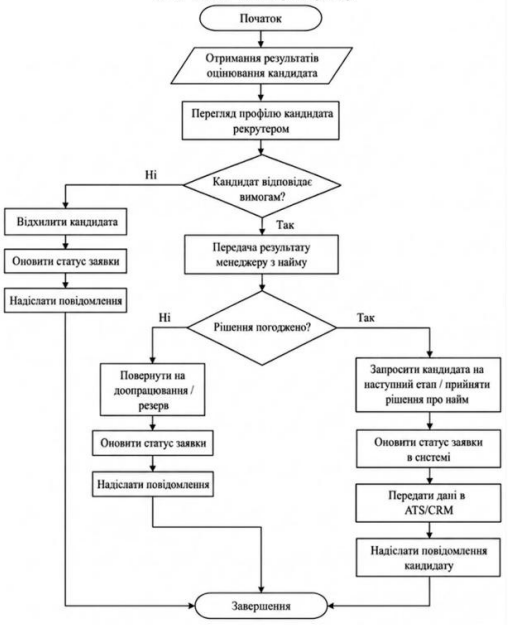


9

БЛОК-СХЕМА АЛГОРИТМУ АВТОМАТИЗОВАНОГО ОЦІНЮВАННЯ КАНДИДАТА

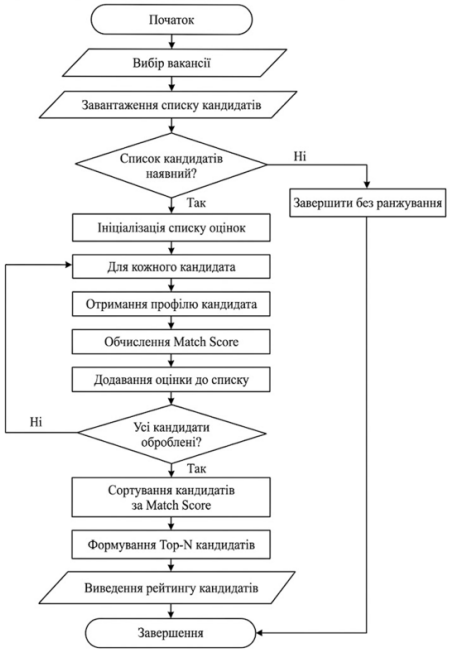


БЛОК-СХЕМА АЛГОРИТМУ РАНЖУВАННЯ КАНДИДАТІВ ЗА ВАКАНСІЄЮ

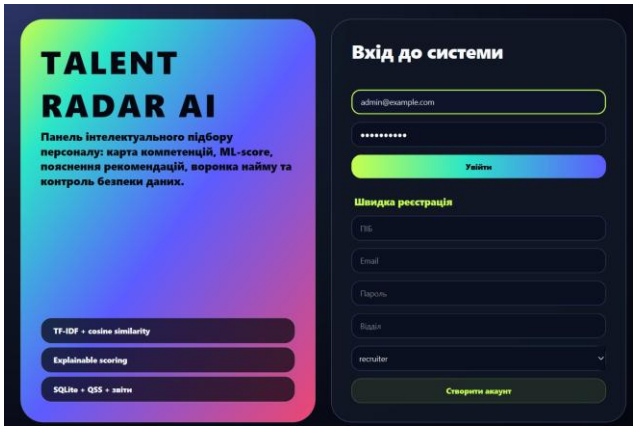


10

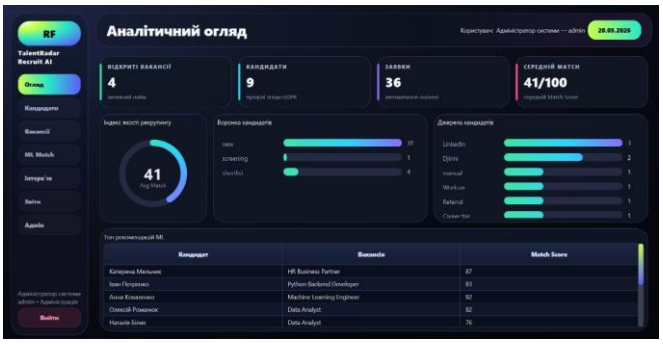
БЛОК-СХЕМА АЛГОРИТМУ ПРИЙНЯТТЯ КАДРОВОГО РІШЕННЯ



ЕКРАННІ ФОРМИ

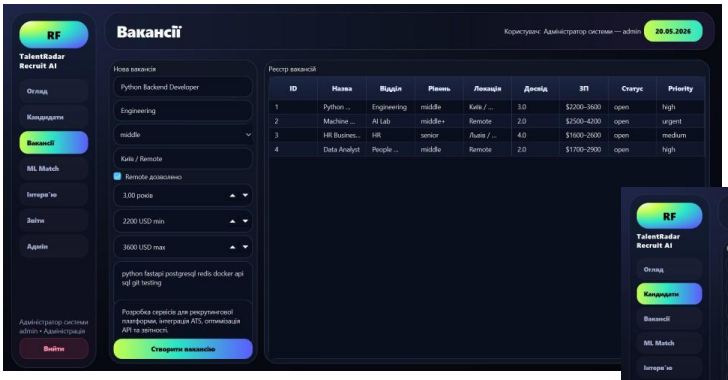


Вікно входу та швидкої реєстрації користувача

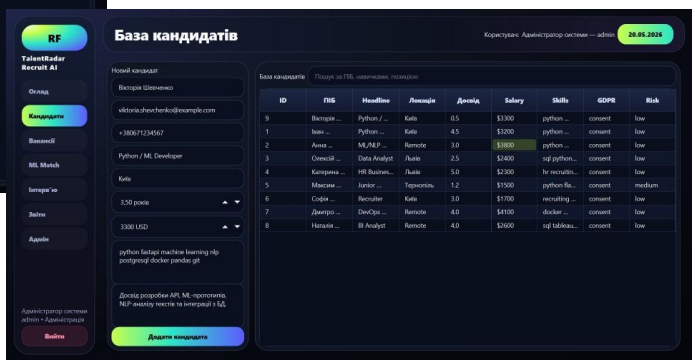


Аналітичний огляд системи оптимізації рекрутингу

ЕКРАННІ ФОРМИ

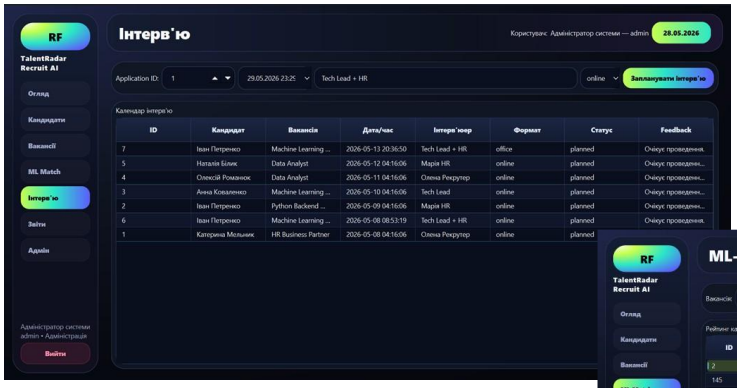


Сторінка бази вакансій

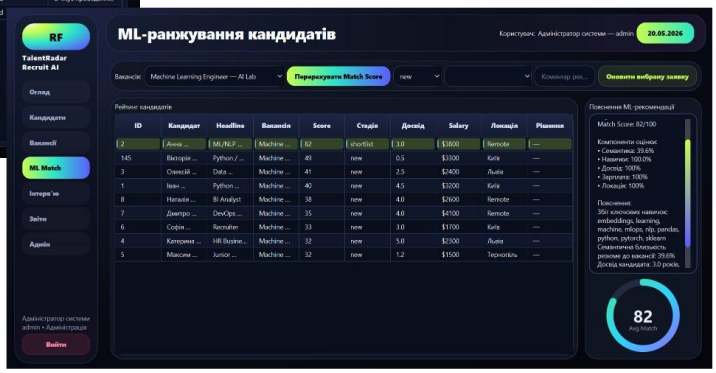


Сторінка бази кандидатів

ЕКРАННІ ФОРМИ

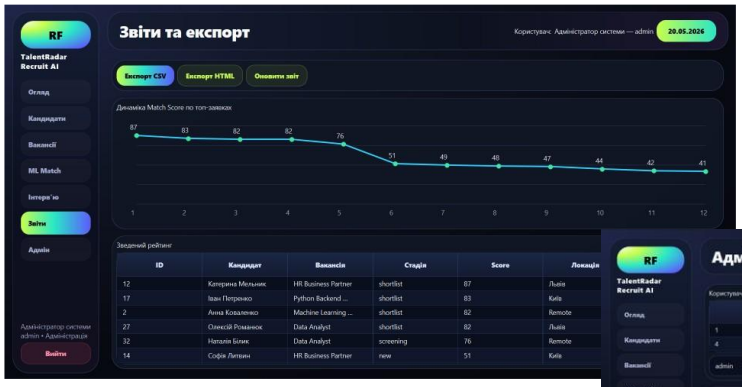


Сторінка планування інтерв'ю

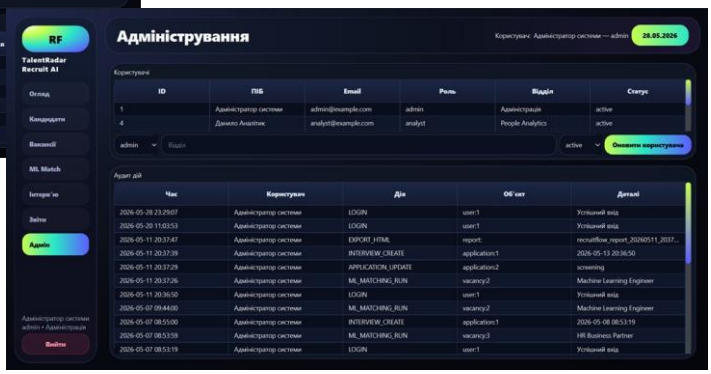


Сторінка ML-ранжування кандидатів

ЕКРАННІ ФОРМИ



Сторінка звітів та експорту результатів



Сторінка адміністрування користувачів і аудиту дій 14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Дудник М. Є. Шахматов І. О. Інтелектуальна мультимодальна система аналізу кандидатів для вакансій на основі LLM-технологій та векторних індексів. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.279-283.
2. Дудник М. Є. Довженко Т.П. Забезпечення безпеки даних в інтелектуальній системі оптимізації процесу рекрутингу. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2026. С.278-280.

16

ВИСНОВКИ

1. Проаналізовано предметну область рекрутингу та процеси підбору персоналу, на основі чого виявлено проблему високої трудомісткості ручного скринінгу та обґрунтовано доцільність впровадження інтелектуальної автоматизації.
2. Проведено порівняльний аналіз існуючих програмних рішень на ринку, визначено їхні ключові недоліки та обмеження, що дозволило обґрунтувати необхідність створення нової системи.
3. Визначено та сформульовано функціональні та нефункціональні вимоги до системи.
4. Обрано стек вебтехнологій, а саме мова програмування Python, фреймворки FastAPI та React, системи керування базами даних PostgreSQL та інструментів Docker і обґрунтовано комплекс моделей машинного навчання і NLP-інструментів для ефективного аналізу й векторизації неструктурованих текстів.
5. Спроєктовано та реалізовано архітектуру застосунку, структуру бази даних та інтелектуальні модулі розрахунку інтегрального показника відповідності із функцією формування пояснення результатів та рольовим розмежуванням прав доступу.
6. Проведено комплексне тестування розробленого програмного забезпечення, у ході якого успішно пройдено перевірку працездатності інтерфейсних модулів, підтверджено стабільність інтеграційних сценаріїв та зафіксовано високу точність роботи алгоритмів оцінювання кандидатів.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

Фрагмент програмного коду створення структури бази даних

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    role TEXT NOT NULL CHECK (role IN ('admin',
'recruiter', 'manager', 'analyst')),
    department TEXT,
    status TEXT NOT NULL DEFAULT 'active',
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP
);

CREATE TABLE candidates (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    phone TEXT,
    headline TEXT,
    location TEXT,
    experience_years REAL DEFAULT 0,
    salary_expectation REAL,
    skills TEXT,
    summary TEXT,
    gdpr_status TEXT DEFAULT 'consent',
    risk_level TEXT DEFAULT 'low',
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP
);

CREATE TABLE vacancies (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    department TEXT,
    description TEXT,
    requirements TEXT,
    required_skills TEXT,
    experience_min REAL DEFAULT 0,
    salary_min REAL,
    salary_max REAL,
    location TEXT,
    status TEXT NOT NULL DEFAULT 'open',
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP
);

CREATE TABLE applications (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    candidate_id INTEGER NOT NULL,
    vacancy_id INTEGER NOT NULL,
    stage TEXT NOT NULL DEFAULT 'new',
    status TEXT NOT NULL DEFAULT 'active',
    decision TEXT,
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT
CURRENT_TIMESTAMP,
    FOREIGN KEY (candidate_id) REFERENCES
candidates(id),
    FOREIGN KEY (vacancy_id) REFERENCES
vacancies(id)
);

CREATE TABLE match_results (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    application_id INTEGER NOT NULL,
    match_score INTEGER NOT NULL,
    semantic_score REAL DEFAULT 0,
    skill_score REAL DEFAULT 0,
    experience_score REAL DEFAULT 0,
    salary_score REAL DEFAULT 0,
    location_score REAL DEFAULT 0,
    explanation TEXT,
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP,
    FOREIGN KEY (application_id) REFERENCES
applications(id)
);

CREATE TABLE interviews (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    application_id INTEGER NOT NULL,
    interview_datetime DATETIME NOT NULL,
    interviewer TEXT NOT NULL,
    format TEXT DEFAULT 'online',
    status TEXT DEFAULT 'planned',
    feedback TEXT,
    FOREIGN KEY (application_id) REFERENCES
applications(id)
);

CREATE TABLE audit_log (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    action TEXT NOT NULL,
    object_type TEXT,
    object_id INTEGER,
    details TEXT,
    created_at DATETIME DEFAULT
CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
INSERT INTO users (full_name, email, password_hash,
role, department, status)
VALUES
```

```
('Адміністратор системи', 'admin@example.com',
'hashed_admin_password', 'admin', 'Адміністрація',
'active'),
('Марія HR', 'hr@example.com', 'hashed_hr_password',
'recruiter', 'HR', 'active'),
('Данило Аналітик', 'analyst@example.com',
'hashed_analyst_password', 'analyst', 'People Analytics',
'active');

INSERT INTO vacancies (title, department, description,
requirements, required_skills, experience_min,
salary_min, salary_max, location)
VALUES
('Python Backend Developer', 'IT', 'Розроблення
серверної частини вебсервісів', 'Python, FastAPI,
PostgreSQL, Docker', 'python fastapi postgresql docker
api', 2, 2000, 4000, 'Київ'),
('Machine Learning Engineer', 'AI', 'Розроблення
моделей машинного навчання для HR-аналітики',
'Python, ML, NLP, PyTorch, pandas', 'python machine
learning nlp pytorch pandas', 2, 2500, 4500, 'Remote'),
('Data Analyst', 'Analytics', 'Аналіз даних рекрутингу
та побудова звітів', 'SQL, Python, Power BI,
статистика', 'sql python analytics power bi statistics', 1,
1500, 3000, 'Львів');
```

Фрагмент програмного коду ML-модуля розрахунку Match Score

```
import re
from typing import Dict, List


def normalize_text(text: str) -> str:
    """
    Нормалізація текстових даних резюме або вакансії.
    """
    text = text.lower()
    text = re.sub(r"[^a-za-яіїєґ0-9\s+#.]", " ", text)
    text = re.sub(r"\s+", " ", text)
    return text.strip()


def split_skills(skills: str) -> List[str]:
    """
    Перетворення рядка навичок у нормалізований
список.
    """
    if not skills:
        return []

    items = re.split(r"[,;/\n]", skills)
    return [normalize_text(item) for item in items if
item.strip()]


def calculate_skill_score(candidate_skills: str,
vacancy_skills: str) -> float:
    """
    Розрахунок відсотка збігу навичок кандидата з
вимогами вакансії.
    """
    candidate_set = set(split_skills(candidate_skills))
    vacancy_set = set(split_skills(vacancy_skills))

    if not vacancy_set:
        return 0.0

    matched = candidate_set.intersection(vacancy_set)
    return len(matched) / len(vacancy_set)


def calculate_experience_score(candidate_years: float,
required_years: float) -> float:
    """
    Оцінювання відповідності досвіду кандидата
мінімальним вимогам вакансії.
    """
    if required_years <= 0:
        return 1.0

    score = candidate_years / required_years
    return min(score, 1.0)


def calculate_salary_score(expected_salary: float,
salary_min: float, salary_max: float) -> float:
    """
    Оцінювання відповідності зарплатних очікувань
кандидата бюджету вакансії.
    """
    if expected_salary is None or salary_min is None or
salary_max is None:
        return 0.5

    if salary_min <= expected_salary <= salary_max:
        return 1.0

    if expected_salary > salary_max:
        difference = expected_salary - salary_max
        return max(0.0, 1.0 - difference / salary_max)

    if expected_salary < salary_min:
        return 0.8

    return 0.0


def calculate_location_score(candidate_location: str,
vacancy_location: str) -> float:
    """
    Оцінювання відповідності локації кандидата
формату роботи.
    """
    candidate_location =
normalize_text(candidate_location or "")
    vacancy_location = normalize_text(vacancy_location
or "")

    if not vacancy_location:
        return 0.5

    if vacancy_location == "remote":
        return 1.0
```

```
    if candidate_location == vacancy_location:
        return 1.0

    return 0.4


def calculate_semantic_score(candidate_summary: str,
vacancy_description: str) -> float:
    """
    Спрощена оцінка семантичної близькості тексту
резюме та опису вакансії.
    У реальній системі цей блок може бути замінений
моделлю Sentence-BERT або іншим NLP-підходом.
    """
    candidate_tokens =
set(normalize_text(candidate_summary).split())
    vacancy_tokens =
set(normalize_text(vacancy_description).split())

    if not vacancy_tokens:
        return 0.0

    intersection =
candidate_tokens.intersection(vacancy_tokens)
    union = candidate_tokens.union(vacancy_tokens)

    if not union:
        return 0.0

    return len(intersection) / len(union)


def calculate_match_score(candidate: Dict, vacancy:
Dict) -> Dict:
    """
    Формування інтегрального показника Match Score.
    """
    skill_score = calculate_skill_score(
        candidate.get("skills", ""),
        vacancy.get("required_skills", "")
    )

    semantic_score = calculate_semantic_score(
        candidate.get("summary", ""),
        vacancy.get("description", "")
    )

    experience_score = calculate_experience_score(
        float(candidate.get("experience_years", 0)),
        float(vacancy.get("experience_min", 0))
    )

    salary_score = calculate_salary_score(
        candidate.get("salary_expectation"),
        vacancy.get("salary_min"),
        vacancy.get("salary_max")
    )

    location_score = calculate_location_score(
        candidate.get("location", ""),
        vacancy.get("location", "")
    )

    weights = {
        "skills": 0.35,
        "semantic": 0.30,
        "experience": 0.15,
        "salary": 0.10,
        "location": 0.10
    }

    total_score = (
        skill_score * weights["skills"] +
        semantic_score * weights["semantic"] +
        experience_score * weights["experience"] +
        salary_score * weights["salary"] +
        location_score * weights["location"]
    )

    match_score = round(total_score * 100)

    explanation = (
        f"Збіг навичок: {skill_score * 100:.1f}%; "
        f"семантична близькість: {semantic_score *
100:.1f}%; "
        f"відповідність досвіду: {experience_score *
100:.1f}%; "
        f"зарплатні очікування: {salary_score *
100:.1f}%; "
        f"локація: {location_score * 100:.1f}%."
    )

    return {
        "match_score": match_score,
        "skill_score": round(skill_score, 3),
        "semantic_score": round(semantic_score, 3),
        "experience_score": round(experience_score, 3),
        "salary_score": round(salary_score, 3),
        "location_score": round(location_score, 3),
        "explanation": explanation
    }


def rank_candidates(candidates: List[Dict], vacancy:
Dict) -> List[Dict]:
    """
    Ранжування кандидатів за вакансією.
    """
    results = []

    for candidate in candidates:
        score_data = calculate_match_score(candidate,
vacancy)
        results.append({
            "candidate_id": candidate["id"],
            "candidate_name": candidate["full_name"],
            "vacancy_id": vacancy["id"],
            "vacancy_title": vacancy["title"],
            "match_score": score_data["match_score"],
            "explanation": score_data["explanation"]
        })
```

```
    results.sort(key=lambda item: item["match_score"],
reverse=True)
    return results
```

Фрагмент програмного коду серверної частини системи

```
from flask import Flask, request, jsonify, session

from werkzeug.security import generate_password_hash,
check_password_hash
import sqlite3
from datetime import datetime

app = Flask(__name__)
app.secret_key = "recruitment_system_secret_key"

DATABASE = "recruitment.db"


def get_connection():
    connection = sqlite3.connect(DATABASE)
    connection.row_factory = sqlite3.Row
    return connection


def write_audit(user_id: int, action: str, object_type: str =
None, object_id: int = None, details: str = None):
    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        INSERT INTO audit_log (user_id, action,
object_type, object_id, details, created_at)
        VALUES (?, ?, ?, ?, ?, ?)
        """,
        (user_id, action, object_type, object_id, details,
datetime.now())
    )

    connection.commit()
    connection.close()


@app.route("/api/register", methods=["POST"])
def register():
    data = request.json

    full_name = data.get("full_name")
    email = data.get("email")
    password = data.get("password")
    role = data.get("role", "recruiter")
    department = data.get("department")

    if not full_name or not email or not password:
        return jsonify({"error": "Не всі обов'язкові поля
заповнені"}), 400

    password_hash = generate_password_hash(password)

    connection = get_connection()
    cursor = connection.cursor()

    try:
        cursor.execute(
            """
            INSERT INTO users (full_name, email,
password_hash, role, department, status)
            VALUES (?, ?, ?, ?, ?, 'active')
            """,
            (full_name, email, password_hash, role,
department)
        )
        connection.commit()
    except sqlite3.IntegrityError:
        return jsonify({"error": "Користувач з таким email
вже існує"}), 409
    finally:
        connection.close()

    return jsonify({"message": "Обліковий запис
створено"}), 201


@app.route("/api/login", methods=["POST"])
def login():
    data = request.json

    email = data.get("email")
    password = data.get("password")

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute("SELECT * FROM users WHERE
email = ? AND status = 'active'", (email,))
    user = cursor.fetchone()
    connection.close()

    if not user or not
check_password_hash(user["password_hash"],
password):
        return jsonify({"error": "Невірний email або
пароль"}), 401

    session["user_id"] = user["id"]
    session["role"] = user["role"]

    write_audit(user["id"], "LOGIN", "user", user["id"],
"Успішний вхід")

    return jsonify({
        "message": "Вхід виконано успішно",
        "user": {
            "id": user["id"],
            "full_name": user["full_name"],
            "role": user["role"],
            "department": user["department"]
        }
    })
```

```
@app.route("/api/candidates", methods=["POST"])
def create_candidate():
    data = request.json

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        INSERT INTO candidates (
            full_name, email, phone, headline, location,
            experience_years, salary_expectation, skills,
summary,
            gdpr_status, risk_level
        )
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """,
        (
            data.get("full_name"),
            data.get("email"),
            data.get("phone"),
            data.get("headline"),
            data.get("location"),
            data.get("experience_years", 0),
            data.get("salary_expectation"),
            data.get("skills"),
            data.get("summary"),
            data.get("gdpr_status", "consent"),
            data.get("risk_level", "low")
        )
    )

    candidate_id = cursor.lastrowid
    connection.commit()
    connection.close()

    write_audit(
        session.get("user_id"),
        "CANDIDATE_CREATED",
        "candidate",
        candidate_id,
        "Створено нового кандидата"
    )

    return jsonify({"message": "Кандидата додано",
"candidate_id": candidate_id}), 201


@app.route("/api/candidates", methods=["GET"])
def get_candidates():
    search = request.args.get("search", "")

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        SELECT *
        FROM candidates
        WHERE full_name LIKE ?
           OR skills LIKE ?
           OR headline LIKE ?
        ORDER BY id DESC
        """,
        (f"%{search}%", f"%{search}%", f"%{search}%")
    )

    candidates = [dict(row) for row in cursor.fetchall()]
    connection.close()

    return jsonify(candidates)


@app.route("/api/vacancies", methods=["POST"])
def create_vacancy():
    data = request.json

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        INSERT INTO vacancies (
            title, department, description, requirements,
            required_skills, experience_min, salary_min,
salary_max,
            location, status
        )
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """,
        (
            data.get("title"),
            data.get("department"),
            data.get("description"),
            data.get("requirements"),
            data.get("required_skills"),
            data.get("experience_min", 0),
            data.get("salary_min"),
            data.get("salary_max"),
            data.get("location"),
            data.get("status", "open")
        )
    )

    vacancy_id = cursor.lastrowid
    connection.commit()
    connection.close()

    write_audit(
        session.get("user_id"),
        "VACANCY_CREATED",
        "vacancy",
        vacancy_id,
        "Створено вакансію"
    )

    return jsonify({"message": "Вакансію створено",
"vacancy_id": vacancy_id}), 201


@app.route("/api/applications/<int:application_id>/decis
ion", methods=["POST"])
def update_application_decision(application_id):
```

```
    data = request.json

    stage = data.get("stage")
    decision = data.get("decision")

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        UPDATE applications
        SET stage = ?, decision = ?, updated_at = ?
        WHERE id = ?
        """,
        (stage, decision, datetime.now(), application_id)
    )

    connection.commit()
    connection.close()

    write_audit(
        session.get("user_id"),
        "APPLICATION_DECISION_UPDATED",
        "application",
        application_id,
        f"Оновлено рішення: {decision}"
    )

    return jsonify({"message": "Статус заявки
оновлено"})


@app.route("/api/interviews", methods=["POST"])
def create_interview():
    data = request.json

    connection = get_connection()
    cursor = connection.cursor()

    cursor.execute(
        """
        INSERT INTO interviews (
            application_id, interview_datetime, interviewer,
            format, status, feedback
        )
        VALUES (?, ?, ?, ?, ?, ?)
        """,
        (
            data.get("application_id"),
            data.get("interview_datetime"),
            data.get("interviewer"),
            data.get("format", "online"),
            data.get("status", "planned"),
            data.get("feedback")
        )
    )

    interview_id = cursor.lastrowid
    connection.commit()
    connection.close()

    write_audit(
        session.get("user_id"),
        "INTERVIEW_CREATED",
        "interview",
        interview_id,
        "Заплановано інтерв'ю"
    )

    return jsonify({"message": "Інтерв'ю заплановано",
"interview_id": interview_id}), 201


if __name__ == "__main__":
    app.run(debug=True)
```