МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

СТАРИХ Олег Васильович

Веб-базована система рекомендацій робочих вакансій / Web-Based Job Recommendation System

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
О. В. Старих

_____

Науковий керівник:
к.т.н., доцент, П. Є. Биковий

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
СТАРИХ Олег Васильович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-базована система рекомендацій робочих вакансій / Web-Based Job Recommendation System
керівник роботи Биковий Павло Євгенович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 грудня 2023 р. № 753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- провести аналіз предметної області;
- провести огляд та аналіз існуючих аналогів;
- визначити основні функціональні вимоги;
- розробити основні сценарії роботи системи;
- створити зрозумілий та унікальний дизайн інтерфейсу;
- розробити функціональну веб-базовану рекомендаційну систему.

5. Перелік графічного матеріалу в роботі:
- діаграма дерева функцій;
- DFD модель;
- STD діаграма;
- алгоритм функціонального модуля генерації рекомендацій;
- реляційна модель системи;
- структурна схема програмної системи;
- UML-діаграма класів;
- UML-діаграма станів.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту.	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ О.В. Старих
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ П.Є. Биковий
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-базована система рекомендацій робочих вакансій» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 74 сторінки і містить 42 ілюстрації, 1 таблицю, 2 додатки та 28 використаних джерел.

Метою даної кваліфікаційної роботи є розробка веб-базованої системи для пошуку роботи з використанням можливостей штучного інтелекту (ШІ). Використовуючи сучасні методи машинного навчання та обробки природної мови, було створено систему, яка допомагає людям, що шукають роботу ефективніше знаходити вакансії, що найбільше відповідають їхнім навичкам і досвіду.

Для розробки системи використовувалися такі технології: HTML5, CSS3, JavaScript, ReactJS для фронтенду; Python, Flask для бекенду; MongoDB як база даних; а також бібліотеки Scikit-learn і Pymongo для реалізації алгоритмів машинного навчання та взаємодії з базою даних.

Було розроблено алгоритмічне забезпечення, яке включає методи векторизації тексту та обчислення косинусної схожості, що дозволяє надавати персоналізовані рекомендації користувачам. Інтерфейс користувача забезпечує зручну взаємодію з системою, дозволяючи швидко знаходити та фільтрувати вакансії. Крім цього проведено функціональне та адаптивне тестування для забезпечення надійності та зручності використання системи.

Ключові слова: ВЕБ-БАЗОВАНА СИСТЕМА, ШТУЧНИЙ ІНТЕЛЕКТ, ПОШУК РОБОТИ, МАШИННЕ НАВЧАННЯ, РЕКОМЕНДАЦІЙНА СИСТЕМА.

ABSTRACT

Qualification work on the topic "Web-based system for recommending job vacancies" for the degree of bachelor in specialty 122 "Computer Science" of the educational program "Computer Science" is written in 74 pages and contains 42 illustrations, 1 table, 2 appendices and 28 references.

The purpose of this qualification work is to develop a web-based job search system using the capabilities of artificial intelligence (AI). Using modern methods of machine learning and natural language processing, a system was created to help job seekers more efficiently find jobs that best match their skills and experience.

The following technologies were used to develop the system: HTML5, CSS3, JavaScript, ReactJS for the frontend; Python, Flask for the backend; MongoDB as a database; and Scikit-learn and Pymongo libraries for implementing machine learning algorithms and interacting with the database.

We developed algorithmic software that includes methods for text vectorization and cosine similarity calculation, which allows us to provide personalized recommendations to users. The user interface provides convenient interaction with the system, allowing you to quickly find and filter jobs. In addition, functional and adaptive testing was conducted to ensure the reliability and usability of the system.

Keywords: WEB-BASED SYSTEM, ARTIFICIAL INTELLIGENCE, JOB SEARCH, MACHINE LEARNING, RECOMMENDATION SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База Даних

ШІ – Штучний Інтелект

API – Application Programming Interface

CSS – Cascading Style Sheets

DFD – Data-Flow Diagram

ERD – Entity-Relationship Diagram

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

NLP – Natural Language Processing

TF-IDF – TF – Term Frequency

IDF – Inverse Document Frequency

STD – State-Transition Diagrams

UML – Unified Modeling Language

ВСТУП

З розвитком комп'ютерних технологій та Інтернету пошук роботи стає все більш динамічним і складним процесом. Сучасний ринок праці пропонує величезну кількість вакансій, що, з одного боку, розширює можливості для шукачів, а з іншого, створює певний інформаційний хаос. Потенційні кандидати часто стикаються з проблемою надмірної кількості пропозицій, що ускладнює процес вибору та прийняття рішень.

Актуальність теми створення веб-базованої системи рекомендацій робочих вакансій полягає в необхідності удосконалення процесів пошуку роботи, які стали значно складнішими через зростаючу кількість вакансій. У цих умовах інтеграція сучасних технологій, таких як штучний інтелект, у веб-додатки дозволяє підвищити ефективність та знизити час на пошук роботи, роблячи цей процес більш точним та зручним для користувачів.

Мета роботи полягає у розробці веб-базованої системи для пошуку роботи, доповненого алгоритмами ШІ. Використовуючи алгоритми ШІ, ця веб-система має на меті спростити процес пошуку роботи. Завдяки використанню алгоритмів підбору, персоналізованих рекомендацій та розширених функцій пошуку, система спрямована на підвищення ефективності та результативності процесу пошуку роботи. Для досягнення цієї мети необхідно виконати наступні завдання: провести аналіз предметної області; провести огляд та аналіз існуючих аналогів; визначити основні функціональні вимоги; розробити основні сценарії роботи системи; створити зрозумілий та унікальний дизайн інтерфейсу; розробити функціональну веб-базовану рекомендаційну систему.

Об'єктом дослідження є веб-базована система рекомендацій робочих вакансій. Предметом дослідження є методи і алгоритми машинного навчання, що застосовуються для аналізу даних про користувачів і вакансії з метою підвищення точності та релевантності рекомендацій в веб-базованій системі.

Практичне значення даної роботи полягає в розробці веб-базованої системи, яка полегшує процес пошуку роботи для користувачів. Система дозволяє швидко

та ефективно знаходити вакансії, що найбільше відповідають навичкам і досвіду користувачів, що зменшує час і зусилля, необхідні для пошуку роботи.

Робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, що містять 28 найменувань та двох додатків. Зміст роботи висвітлено на 46 сторінках основного тексту і містить 42 рисунки. В кожному з розділів описано наступну інформацію: у першому розділі було проведено детальний аналіз предметної області та огляд існуючих аналогів систем рекомендацій робочих вакансій; другий розділ роботи присвячений розробці алгоритмічної та інформаційної підтримки системи; у третьому розділі роботи розглянуто програмно-технологічне забезпечення системи, включаючи реалізацію програмного забезпечення та інтерфейс користувача.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження представлено на міжнародній науково-практичній конференції м. Ізмаїл, 18 травня 2024 р. "Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика" в секції "Математичні методи, моделі та інформаційні технології в економіці", де була представлена доповідь на тему "Веб-базована система рекомендацій робочих вакансій" і опубліковані тези по даній доповіді.

1. АНАЛІЗ ВЕБ-БАЗОВАНИХ СИСТЕМ РЕКОМЕНДАЦІЙ РОБОЧИХ ВАКАНСІЙ

1.1 Опис предметної області

Дана веб-базована система рекомендацій була створена для допомоги користувачам ефективно знаходити роботу, яка найкраще відповідає їхнім навичкам, досвіду та інтересам. Це не просто інструмент для пошуку роботи, а складна система, яка використовує алгоритми та аналітику для надання персоналізованих рекомендацій [8].

Специфіка тематики передбачає врахування факторів динамічності та мінливості, оскільки вакансії зазвичай з'являються на короткий термін та відносно швидко змінюють свій статус. Також необхідно враховувати не тільки мехпнізм представлення вакансій, але й аналіз взаємодії з користувачами. Тому ефективна розробка такого роду системи, вимагає глибокого розуміння цих двох ключових аспектів [3].

Транзакційний компонент системи відповідає за збір, зберігання та обробку даних. У випадку нашої системи це включає в себе збір та агрегування інформації про вакансії з різних джерел, включаючи популярні сайти з пошуку роботи, сайти компаній та інші платформи, також моніторинг та збір даних про взаємодію користувачів з системою.

Аналітичний компонент системи зосереджений на обробці та аналізі зібраних даних. В контексті даної системи він представляє детальний аналіз даних у різних демографічних, географічних та професійних сегментах.

Додатково до основних компонентів системи, в ході дослідження була створена діаграма дерева функцій (Рисунок 1.1), яка показує основні функції системи і є важливим інструментом для розуміння процесу функціонування системи. Дана діаграма включає кілька основних блоків, кожен з яких виконує свої завдання.

Блок "Агрегація вакансій" включає процеси збору даних та їх форматування. Збирання даних передбачає отримання інформації про вакансії з різних джерел,

використовуючи технології веб-скрапінгу. Після цього зібрані дані проходять процес форматування, що приводить їх до єдиного стандарту, зручного для подальшої обробки та аналізу. Це включає нормалізацію даних, видалення дублікатів та перевірку якості даних.

Блок "Взаємодія з користувачем" відповідає за введення ключових слів. Користувач взаємодіє з системою, вводячи ключові слова, які описують бажану вакансію або навички, а також вказує бажану локацію. Ця інформація є основою для формування пошукового запиту, який буде використовуватись для порівняння з описами вакансій.

Система рекомендацій здійснює підбір вакансій за ключовими словами та за фільтрами. Після введення ключових слів користувачем система аналізує зібрані дані, використовуючи алгоритми машинного навчання для обробки тексту, визначення ключових характеристик вакансій та надання персоналізованих рекомендацій.

Користувацький інтерфейс забезпечує інтуїтивно зрозумілий доступ до функцій системи. Він дозволяє користувачам легко знаходити та фільтрувати вакансії, переглядати рекомендації та взаємодіяти з системою. Інтерфейс включає блок рекомендованих вакансій, де користувач може переглянути список вакансій, що найкраще відповідають його запиту, та сторінку вакансії, де відображається детальна інформація про обрану вакансію.

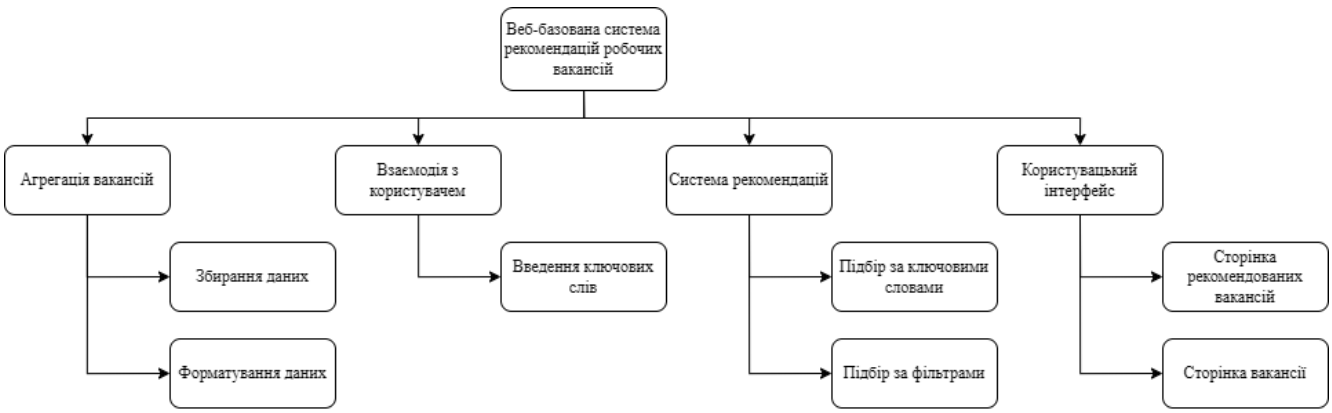

Рисунок 1.1 – Діаграма дерева функцій

1.2 Огляд і аналіз існуючих аналогів

З кожним роком все більшої популярності набувають різноманітні інтернет-ресурси з пошуку роботи, які допомагають мільйонам людей знайти роботу, а тисячам компаній – знайти та підібрати працівників на відкриті вакансії. Більшість з них схожі один на одного і виконують однакову функцію. Однак є й такі сайти, які хоч і розміщують оголошення про вакансії, але виконують різні функції. Тому було проаналізовано сайти-аналоги та зроблено порівняльний огляд деяких найпопулярніших систем пошуку роботи.

Нижче наведено сайт Robota.ua, який є найбільшим в Україні сервісом з пошуку роботи, що пропонує великий вибір можливостей для тих, хто шукає роботу, та роботодавців. На рисунку 1.2 показано головну сторінку сайту Robota.ua. Цей сайт пропонує широкий перелік вакансій у різних сферах, у тому числі IT, банківській справі, військовій справі, сфері послуг та продажу. Пошук вакансій здійснюється за назвою та розташуванням (Рисунок 1.3), рівнем заробітної плати та іншими критеріями. Крім того, Robota.ua надає можливість робити пошук за назвою компанії (Рисунок 1.4), за популярними містами та професіями (Рисунок 1.5), і крім цього створити профіль користувача, який дозволяє зберігати своє резюме, та відстежувати свої відгуки на різні вакансії [17].

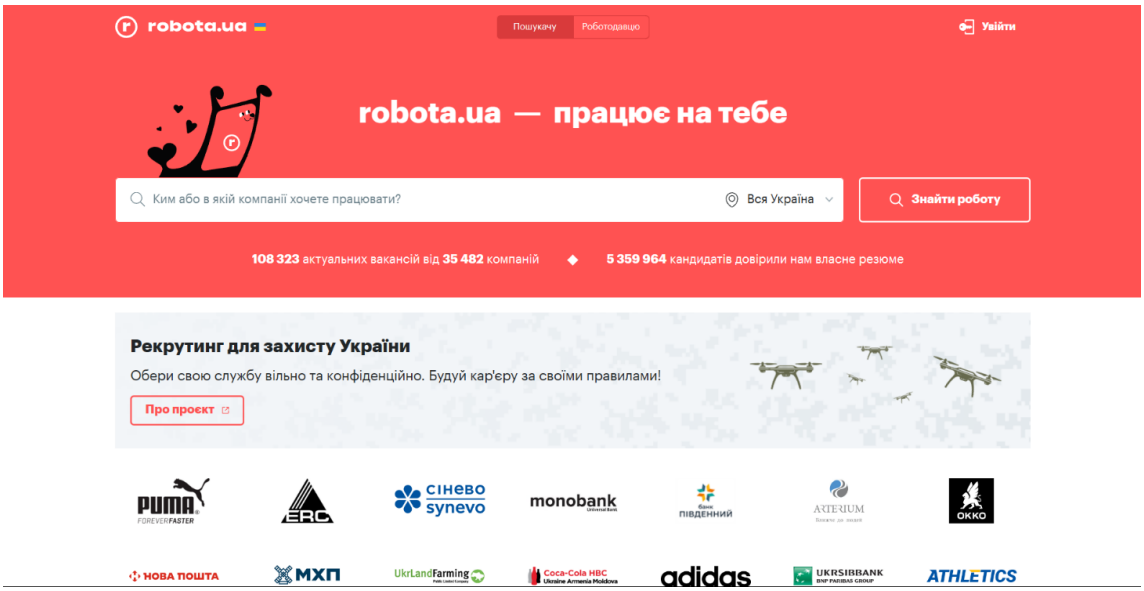

Рисунок 1.2 – Головна сторінка сайту Robota.ua

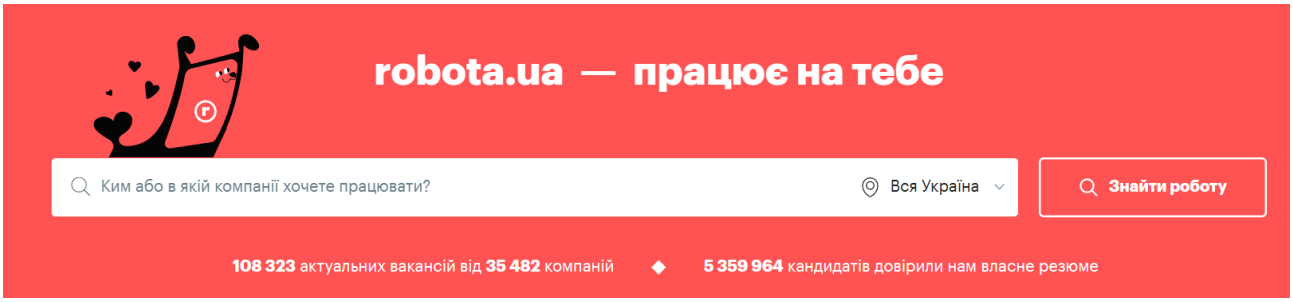

Рисунок 1.3 – Пошук за назвою та розташуванням

Рисунок 1.4 – Пошук за компаніями

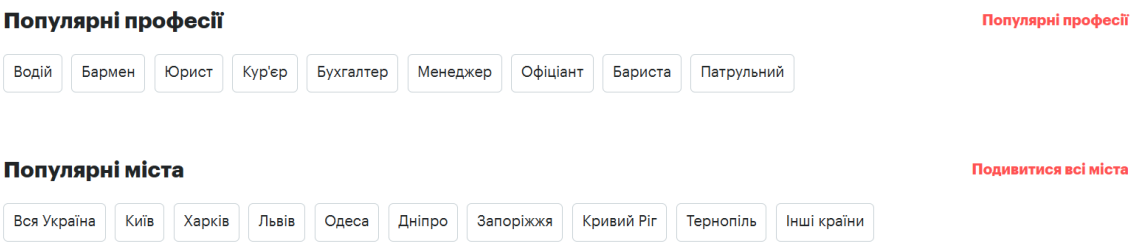

Рисунок 1.5 – Пошук за популярними містами та професіями

Work.ua – це ще один великий і популярний ресурс для пошуку роботи в Україні. На рисунку 1.6 показано вигляд головної сторінки. Сайт пропонує розширений пошук роботи за назвою посади (Рисунок 1.7), категоріями (Рисунок 1.8), місцеперебування (Рисунок 1.9) та за назвою компанії (Рисунок 1.10). Крім того, користувачі можуть створювати або завантажувати свої

резюме (Рисунок 1.11) та зберігати їх у профілі та отримувати оновлення про вакансії, які відповідають їхнім критеріям. Також Work.ua надає корисні ресурси та новини стосовно пошуку роботи (Рисунок 1.12), такі як поради щодо написання резюме, проходження співбесіди та розвитку кар'єри [28].

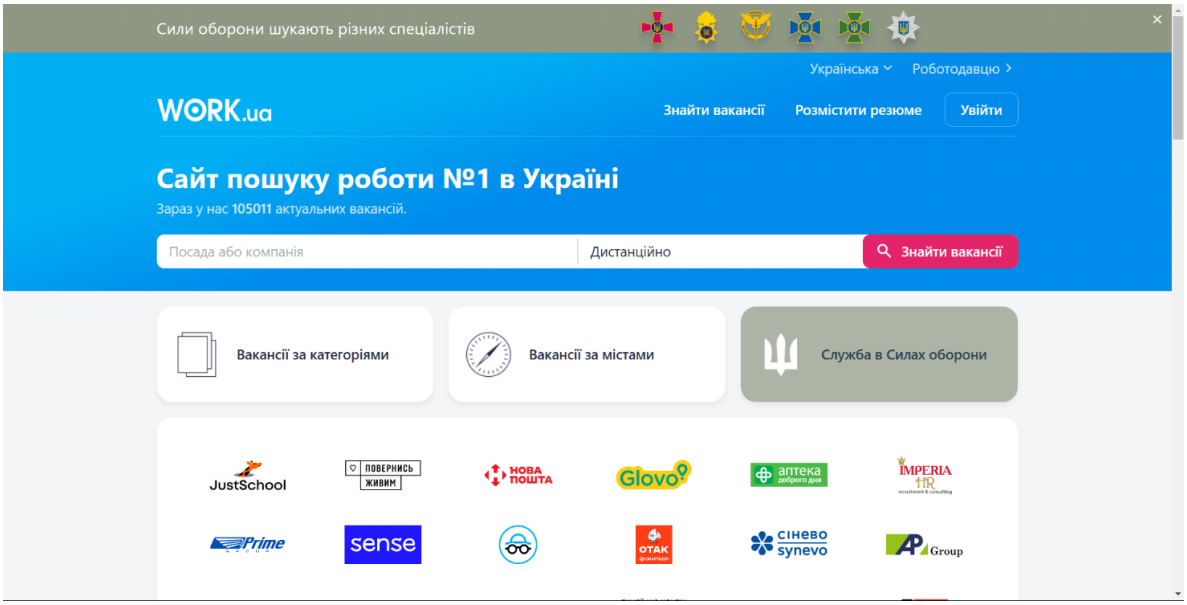

Рисунок 1.6 – Головна сторінка сайту Work.ua

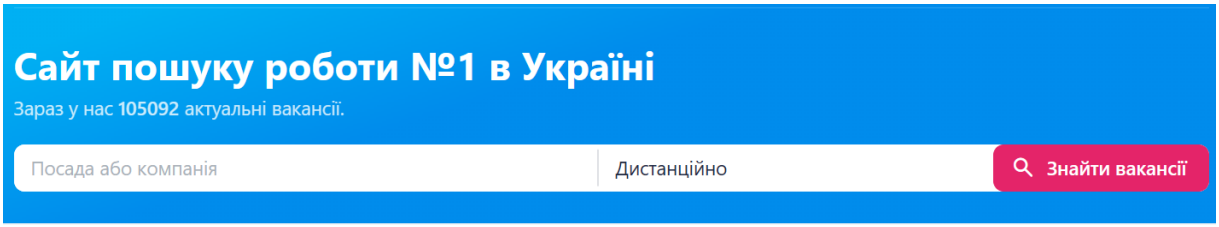

Рисунок 1.7 – Пошук вакансій за назвою

Рисунок 1.8 – Пошук за категоріями

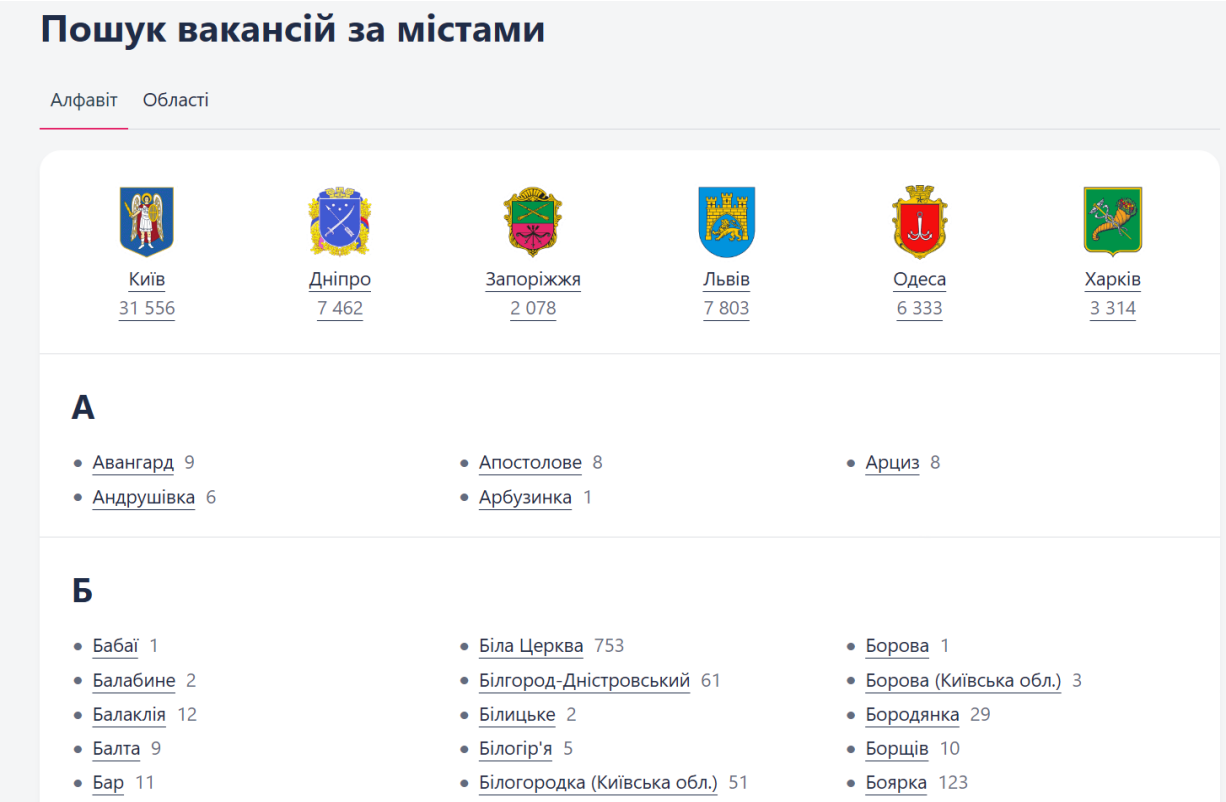

Рисунок 1.9 – Пошук за містами

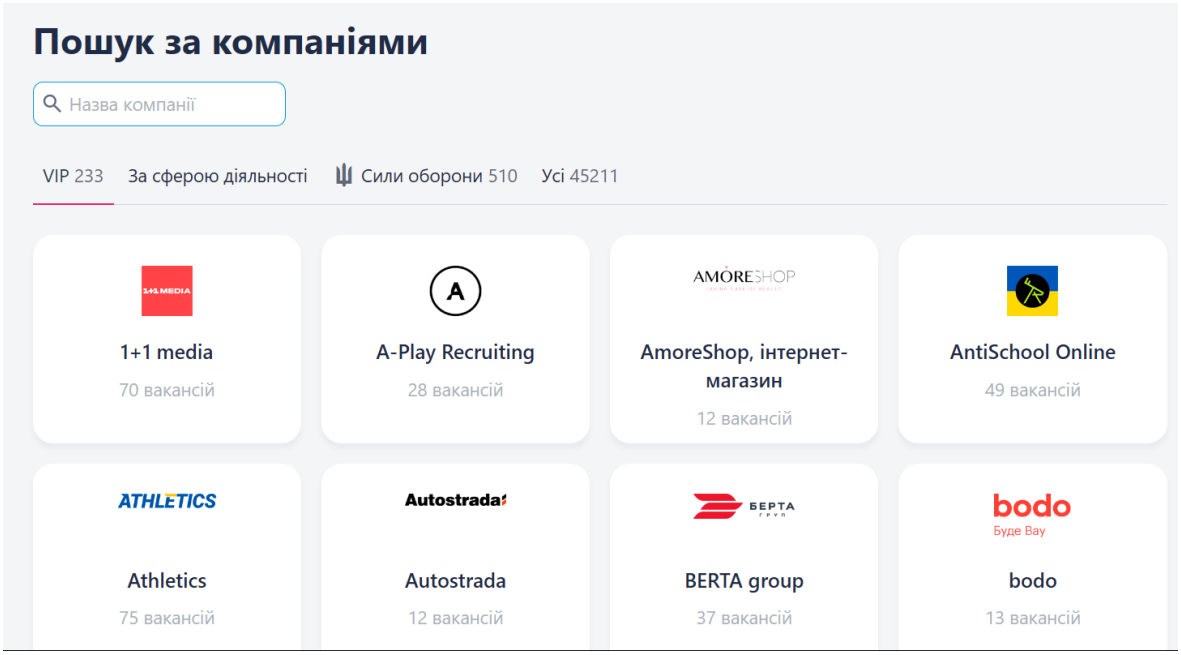

Рисунок 1.10 – Пошук за компаніями

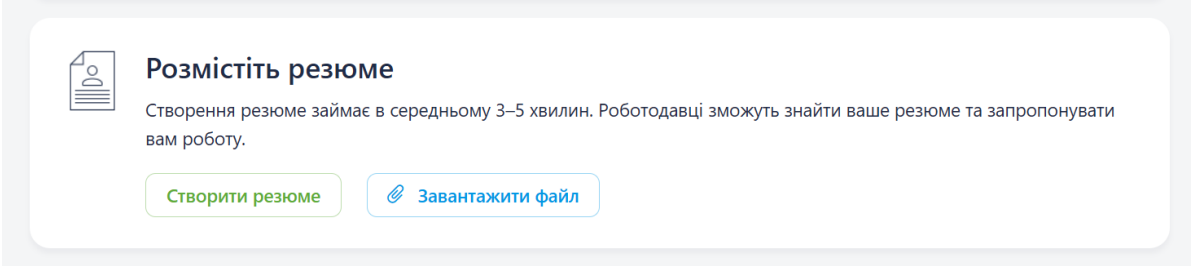

Рисунок 1.11 – Функція створення або завантаження резюме

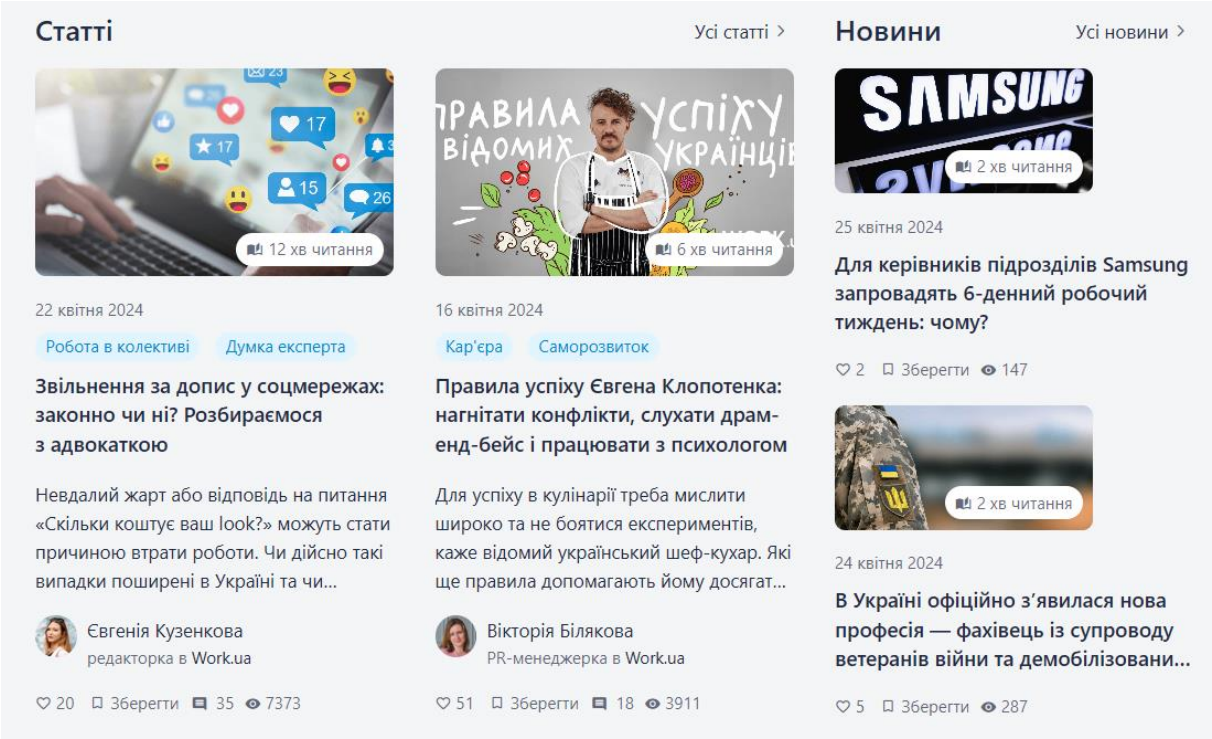

Рисунок 1.12 – Корисні статті та новини

Наступним сервісом буде Indeed.com – це міжнародний сайт із пошуку роботи з великим охопленням в Україні. На рисунку 1.13 відображено вигляд головної сторінки. За допомогою цього сайту користувачі можуть здійснювати пошук роботи за ключовими словами, місцеперебування (Рисунок 1.14) та іншими критеріями. Цікавою особливістю Indeed.com є можливість перегляду відгуків та рейтингів компаній (Рисунок 1.15), які розміщують вакансії, що дозволяє користувачам дізнатися більше інформації про роботодавця [9].

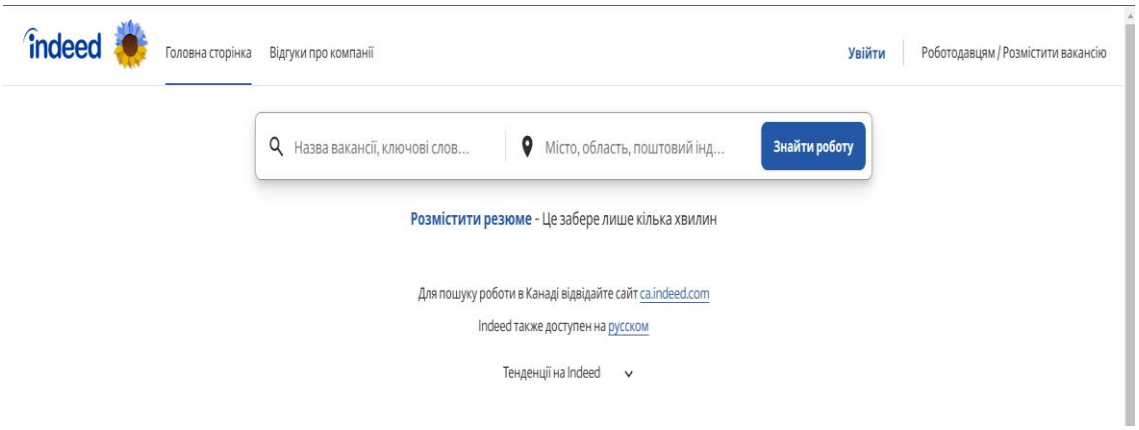

Рисунок 1.13 – Головна сторінка сайту Indeed.com

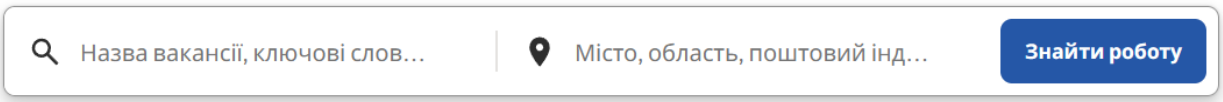

Рисунок 1.14 – Пошук вакансій за назвою, ключовим словом та містом

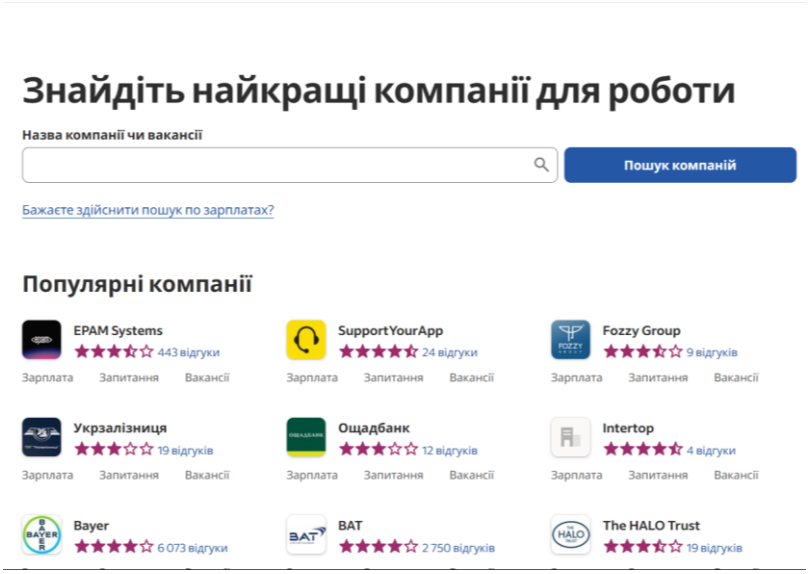

Рисунок 1.15 – Список найкращих компаній

1.3 Постановка задачі дослідження

У попередніх розділах було загально описано наявні платформи для пошуку роботи, які зазвичай потребують реєстрації та фокусуються на поданні заявок на вакансії. Однак під час даного дослідження було виявлено, що на ринку існує потреба, яка б об'єднувала вакансії з різних джерел і персоналізувала рекомендації на основі ключових слів, наданих користувачем. Наявні сайти з пошуку роботи зазвичай не мають такого функціонала і не можуть видавати нерелевантні результати пошуку [8].

Цей проєкт має на меті розробити веб-базовану систему рекомендацій для пошуку роботи з наступними ключовими функціями:

- збір вакансій з різних інтернет-джерел;
- рекомендація відповідних вакансій на основі введених користувачем ключових слів (наприклад, навички або технології);
- розробка простого та інтуїтивно зрозумілого інтерфейсу;

- дозволити користувачам отримувати повний доступ до сервісу без обов'язкової реєстрації;
- перенаправляти, у разі зацікавленості, користувачів, на оригінальну сторінку вакансії на початковому веб-сайті.

Метою цієї кваліфікаційної роботи є розробка веб-базованої рекомендаційної системи рекомендацій робочих вакансій. Для досягнення цієї мети необхідно виконати наступні завдання:

- провести аналіз предметної області;
- провести огляд та аналіз існуючих аналогів;
- визначити основні функціональні вимоги;
- розробити основні сценарії роботи системи;
- створити зрозумілий та унікальний дизайн інтерфейсу;
- розробити функціональну веб-базовану рекомендаційну систему [1].

2. АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна структура проектованої системи

Розробка веб-системи для пошуку роботи з використанням можливостей штучного інтелекту є надзвичайно актуальним і важливим завданням на сучасному ринку праці [3]. Основною метою цього підрозділу є детальний аналіз концептуальних засад, що лежать в основі створення такого веб-застосунка. Такий підхід значно полегшує процес пошуку роботи для користувачів. Використання передових технологій, таких як алгоритми машинного навчання та аналіз даних, забезпечує новий рівень автоматизації та персоналізації у сфері пошуку роботи [8].

Проектована система складається з кількох основних модулів, кожен з яких виконує свої функції, забезпечуючи загальну роботу системи. Нижче наведено детальний опис кожного з цих модулів, а також їхні функції та характеристики.

1. Модуль збору даних є одним із ключових компонентів системи, оскільки саме він забезпечує накопичення необхідної інформації для подальшої обробки та аналізу. Він виконує наступні функції:

- збір інформації про вакансії з різних джерел, таких як популярні сайти з пошуку роботи, корпоративні сайти та інші інтернет-ресурси. Цей процес включає використання технологій веб-скрапінгу, що дозволяє автоматично витягати дані з веб-сторінки;
- збір інформації про взаємодію користувача з системою для подальшого аналізу, включаючи відстеження дій користувача на сайті, таких як пошук вакансій, перегляд пропозицій або застосування фільтрів.

Модуль має наступні характеристики:

- використання технологій веб-скапінгу для автоматизованого збору даних з веб-ресурсів, що дозволяє ефективно агрегувати інформацію з багатьох джерел, забезпечуючи її актуальність і повноту;
- інтеграція з API різних платформ для отримання актуальної інформації про вакансії в режимі реального часу. API надають прямий доступ до баз даних вакансій, що значно спрощує процес збору даних [13];

- зберігання зібраних даних у структурованому вигляді для подальшої обробки, що забезпечує легкий доступ та аналіз даних;

2. Аналітичний модуль є центральною частиною системи, яка відповідає за обробку та аналіз зібраних даних. Він містить наступні функції:

- аналіз зібраних даних для виявлення тенденцій та взаємозв’язків, що дозволяє надавати персоналізовані рекомендації за допомогою алгоритмів машинного навчання для обробки великих обсягів даних;
- класифікація вакансій за різними параметрами аби структурувати дані та підвищити їхню користь для користувачів.

Він містить наступні характеристики:

- використання алгоритмів машинного навчання для обробки великих обсягів даних, що дозволяє виявляти приховані взаємозв'язки та тенденції, які важко помітити при ручному аналізі.
- застосування методів кластеризації та регресійного аналізу для визначення релевантності вакансій для конкретного користувача, що дозволяє створювати персоналізовані рекомендації.
- постійне оновлення моделей машинного навчання на основі нових даних для підвищення точності рекомендацій.

3. Модуль рекомендацій є важливим компонентом системи, який безпосередньо взаємодіє з користувачами, надаючи їм персоналізовані рекомендації щодо вакансій. Його функції:

- надання користувачам персоналізованих рекомендацій на основі введених ключових слів, історії пошуку та взаємодії з системою, що дозволяє швидко знаходити найбільш релевантні вакансії;
- генерація списку релевантних вакансій для кожного користувача на основі аналізу даних про користувача та вакансії.

Його характеристики:

- використання фільтрування контенту для аналізу ключових слів і надання рекомендацій на основі змісту вакансій.

- адаптація рекомендаційної системи до змін у поведінці користувачів і ринку праці, що забезпечує актуальність рекомендацій.

4. Інтерфейс користувача є ключовим елементом системи, який забезпечує взаємодію користувачів з системою. Його функції:

- забезпечення інтуїтивно зрозумілого та зручного доступу до функцій системи для користувачів, включаючи можливість легкої навігації по сайту, фільтрації та сортування вакансій.

Характеристики:

– реалізація на основі сучасних веб-технологій (HTML/CSS, JavaScript, React), що забезпечує швидку та ефективну роботу інтерфейсу з адаптивним дизайном для зручного доступу до системи з різних пристроїв [4, 7, 16, 26].

- інтеграція з зовнішніми сервісами для полегшення процесу пошуку та подання заявок на вакансії.

Інформаційні зв'язки між елементами системи забезпечують її узгоджену роботу та ефективний обмін даними. Основні інформаційні потоки включають:

- модуль збору даних отримує дані від зовнішніх джерел (інтернет-ресурси, API) та передає їх до аналітичного модуля для подальшого аналізу;

- аналітичний модуль обробляє отримані дані та зберігає результати в базі даних для подальшого використання модулем рекомендацій, що дозволяє створювати точні та персоналізовані рекомендації для користувачів;

- модуль рекомендацій використовує оброблені дані для генерування персоналізованих пропозицій, які виводяться у інтерфейсі користувача.

- своєю чергою інтерфейс користувача дозволяє взаємодіяти з системою, отримувати рекомендації, застосовувати фільтри та переглядати вакансії, що робить процес пошуку роботи максимально зручним та ефективним для користувачів.

Крім цього дана система взаємодіє з такими зовнішніми об'єктами та системами:

- зовнішні джерела вакансій, так як веб-сайти та API, звідки система отримує дані, що забезпечує актуальність та повноту інформації про вакансії [13];

- користувачі шо шукають роботу, які використовують систему для пошуку вакансій, що забезпечує інтерактивність та персоналізацію системи;
- сховища, де зберігаються оброблені дані та результати аналізу, що забезпечує надійне зберігання та швидкий доступ до даних.

Також для ілюстрації архітектури системи доцільно використовувати структурні схеми. В нашому випадку буде використано діаграми DFD (Data Flow Diagram) 0 рівня, DFD 1 рівня та STD (State-Transition Diagram) які відображають логіку функціонування розроблюваної системи [20, 27].

DFD 0 рівня показує основний потік даних між користувачем та системою (рисунок 2.1). Користувач взаємодіє із головною сторінкою веб-застосунка, де вводить ключові слова для пошуку вакансій. Система генерує рекомендації на основі введених даних та відображає результати. Користувач має можливість фільтрувати результати, переглядати більше вакансій, обирати конкретні вакансії та переглядати попередні запити. Ця діаграма показує взаємодію користувача з основними функціональними блоками системи [27].

Рисунок 2.1 – DFD модель 0 рівня

DFD 1 рівня (рисунок 2.2) деталізує процес збору даних з зовнішніх джерел (веб-сайтів та API), їх обробку та аналіз. Зібрані дані проходять через аналітичний модуль, який обробляє інформацію і передає її до модуля рекомендацій. Останній генерує персоналізовані рекомендації, які відображаються в інтерфейсі

користувача. Ця діаграма демонструє внутрішні процеси та потоки даних, що відбуваються у системі від моменту збору інформації до формування рекомендацій [27].

Рисунок 2.2 – DFD модель 1 рівня Веб-базованої системи

STD діаграма (рисунок 2.3) відображає стани та переходи між ними у веб-базованій системі рекомендацій робочих вакансій. Користувач взаємодіє з системою, яка збирає дані про вакансії з зовнішніх джерел. Ця діаграма ілюструє можливі стани системи під час її роботи та переходи між цими станами залежно від дій користувача та внутрішніх процесів[20].

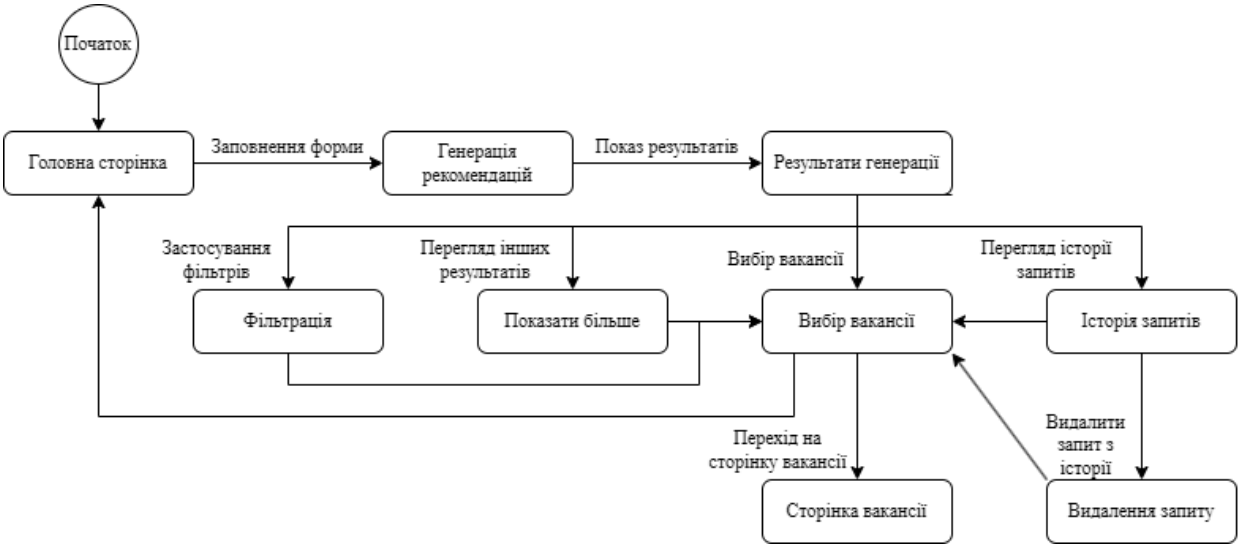

Рисунок 2.3 – STD діаграма

Використання таких структурних схем дозволяє чітко зрозуміти архітектуру системи, її компоненти та їх взаємодію, що є важливим для подальшого розвитку та вдосконалення веб-застосунка.

2.2 Алгоритмічне забезпечення

У цьому розділі ми розглянемо основи алгоритму роботи розробленої рекомендаційної системи для пошуку роботи. У системі використовуються методи обробки природної мови (NLP) та машинного навчання для аналізу текстових описів вакансій, визначення ключових характеристик та надання користувачам персоналізованих рекомендацій, що відповідають їхнім потребам та вподобанням [8, 11]. Алгоритмічне програмне забезпечення описує логіку розв'язання поставленої задачі та формування результатів, передбачаючи всі можливі ситуації, які можуть виникнути в процесі роботи системи.

Щоб краще зрозуміти принцип роботи даної рекомендаційної системи, слід розглянути узагальнений алгоритм її функціонування, що складається з декількох взаємопов'язаних кроків, які забезпечують ефективну роботу усієї системи:

1. Отримання вхідних даних. Користувач взаємодіє з інтерфейсом системи, вводячи ключові слова, які описують бажану вакансію (наприклад,

"Python розробник", "Data Scientist"), а також вказуючи бажану локацію. Ці дані є основою для формування пошукового запиту.

2. Попередня обробка даних:

- система завантажує дані про вакансії з обраного джерела, яке може бути JSON-файлом або базою даних MongoDB;
- заповнення пропущених значень у описах вакансій, щоб уникнути помилок під час подальшого аналізу;
- об'єднання ключових слів, з інформацією про локацію в єдиний пошуковий запит для порівняння з описами вакансій.

3. Векторизація тексту (TF-IDF). Система застосовує метод TF-IDF (Term Frequency-Inverse Document Frequency) для перетворення текстових описів вакансій у числові вектори [21]. TF-IDF – це статистична міра, яка оцінює важливість слова в документі (в нашому випадку, в описі вакансії) відносно колекції документів (всіх описів вакансій). Таким чином, кожна вакансія отримує свій унікальний числовий вектор, який відображає її зміст.

4. Обчислення косинусної схожості. Система обчислює косинусну схожість між вектором пошукового запиту користувача та векторами кожної вакансії. Косинусна схожість – це міра подібності між двома векторами, яка визначається косинусом кута між ними. Значення косинусної схожості варіюється від 0 (повна відмінність) до 1 (повна ідентичність).

5. Ранжування та відбір. Вакансії ранжуються за спаданням косинусної схожості з пошуковим запитом користувача. Це означає, що вакансії, описи яких найбільше схожі на запит користувача, будуть розташовані на початку списку. Система відбирає певну кількість вакансій з найвищою схожістю (наприклад, топ-10) для формування рекомендацій [8].

6. Повернення результатів. Система формує структурований список рекомендованих вакансій, який включає різноманітну інформацію про кожну вакансію: назву, компанію, місто, тип зайнятості, короткий опис та унікальний ідентифікатор. Цей список повертається користувачеві у зручному форматі JSON, який легко обробляється веб-додатком.

Основну логіку рекомендаційної системи реалізує функція get_recommendations, яка є центральним елементом алгоритмічного забезпечення. Ця функція приймає на вхід ключові слова та локацію, введені користувачем, і виконує всі необхідні кроки для формування персоналізованих рекомендацій (Рисунок 2.4):

1. Завантаження даних про вакансії з джерела та їх підготовка до обробки.
2. Створення об'єкта TF-IDF векторизатора та застосування його до описів вакансій, перетворення їх у числові вектори.
3. Обчислення косинусної схожості між вектором пошукового запиту користувача та векторами кожної вакансії, отримання міри їхньої подібності.
4. Сортування вакансій за спаданням косинусної схожості, розміщення найбільш релевантних вакансій на початку списку.
5. Відбір певної кількості вакансій з найвищою схожістю (наприклад, топ-10).
6. Створення та повернення списку рекомендованих вакансій у форматі JSON, який містить детальну інформацію про кожну вакансію.

Рисунок 2.4 – Алгоритм функціонального модуля генерації рекомендацій

Точність та достовірність рекомендацій, що надаються системою, залежать від багатьох факторів, включаючи:

- якість та репрезентативність даних, де чим більше даних про вакансії доступно системі, і чим краще вони відображають різноманітність вакансій на ринку праці, тим точнішими будуть рекомендації;
- адекватність методу TF-IDF, оскільки TF-IDF є ефективним методом для багатьох завдань обробки природної мови, але він може бути не ідеальним для всіх випадків;
- інтерпретація косинусної схожості, що є лише однією з можливих мір подібності між текстами.

Для покращення точності та достовірності рекомендацій можна застосувати такі підходи:

- застосування лематизації (приведення слів до початкової форми), стемінгу (видалення закінчень слів) та видалення стоп-слів (часто вживаних слів, які не несуть важливої інформації) може допомогти покращити якість векторизації тексту та, відповідно, точність рекомендацій [11];
- окрім ключових слів та локації, можна враховувати й інші фактори, такі як досвід роботи користувача, його освіта, навички та інтереси. Це дозволить зробити рекомендації більш персоналізованими та релевантними;
- використання складніших моделей машинного навчання, таких як нейронні мережі або градієнтний бустинг, може допомогти виявити складніші залежності між даними та покращити якість рекомендацій.

2.3 Інформаційне забезпечення

Інформаційне забезпечення є важливим елементом для успішного функціонування системи рекомендацій, а також їх структурування, зберігання та обробку. Це забезпечення відіграє важливу роль у забезпеченні точності та ефективності рекомендацій, а також у загальному функціонуванні системи.

Вхідна інформація включає дані, які користувач вводить у систему, а також дані про вакансії, які завантажуються з зовнішніх джерел. Основні категорії вхідної інформації:

Дані від користувача:

- ключові слова для пошуку вакансій: Користувач вводить ключові слова, які описують бажану посаду або навички, що допомагає звузити пошук до найбільш релевантних вакансій.
- бажана локація: Користувач вказує місто або регіон, де він хотів би знайти роботу. Це дозволяє враховувати географічні вподобання користувача при пошуку вакансій.

Дані про вакансії:

- унікальний ідентифікатор вакансії: Унікальний код, що ідентифікує вакансію у системі.
- назва вакансії: Ідентифікатор посади, що пропонується (наприклад, "Frontend Developer").
- назва компанії: Назва компанії, яка пропонує вакансію (наприклад, "NIX").
- місто або регіон: Місце розташування вакансії (наприклад, "Київ").
- тип зайнятості: Характер роботи (повна, часткова, стажування).
- опис вакансії: Детальний опис посадових обов'язків, вимог до кандидата, умов праці тощо.

Вихідна інформація включає результати обробки вхідних даних та рекомендації, які система надає користувачеві. Основні категорії вихідної інформації рекомендованих вакансій:

- унікальний ідентифікатор вакансії: Унікальний код, що ідентифікує вакансію у системі.
- назва вакансії: Назва рекомендованої посади.
- назва компанії: Назва компанії, яка пропонує рекомендовану вакансію.
- місто або регіон: Місце розташування рекомендованої вакансії.

- тип зайнятості: Тип зайнятості, що пропонується для рекомендованої вакансії.
- опис вакансії: Повний опис обов'язків та вимог до кандидата.

Концептуальне інфологічне проєктування є важливою частиною розробки інформаційних систем, оскільки воно визначає основні принципи та концепції, на яких буде базуватися система. У рамках цього процесу створюються моделі даних, які відображають структуру і взаємозв'язки між основними елементами системи. Існують дві основні моделі, що використовуються на цьому етапі: локальна і глобальна моделі.

Локальна модель фокусується на структурі даних окремих модулів або функцій системи. Це дозволяє розробникам глибоко розуміти, як кожен модуль буде працювати з даними, які дані необхідні для його функціонування та які дані він буде виробляти. У випадку нашої системи рекомендацій локальна модель включає такі сутності:

- користувач: Інформація про користувача включає cookie для збереження попередніх запитів, зокрема ключові слова для пошуку та бажану локацію.
- вакансія: Інформація про вакансію, зокрема назва, компанія, місце розташування, тип зайнятості та опис.

У свою чергу глобальна модель об'єднує всі локальні моделі в єдину цілісну структуру, що дозволяє бачити загальну картину взаємодії даних у системі. Ця модель важлива для забезпечення узгодженості даних між різними модулями системи та для виявлення потенційних проблем у взаємодії даних на ранніх етапах розробки.

Глобальна модель включає сутності "Користувач" та "Вакансія", а також зв'язки між ними, що відображають рекомендаційний процес. Наприклад, один користувач може мати кілька рекомендованих вакансій, а кожна вакансія може бути рекомендована багатьом користувачам. Це дозволяє ефективно організувати дані і забезпечити їхню доступність для всіх модулів системи .

Проєктування глобальної даталогічної моделі даних є наступним кроком після концептуального інфологічного проєктування. Цей процес включає

створення діаграм і моделей, які детально описують структуру даних у системі та їхні взаємозв'язки. Основні методи, що використовуються для цього, включають ERD (Entity-Relationship Diagram) у нотації IDEF1X та діаграми класів, а також реляційну модель даних.

ERD (Entity-Relationship Diagram) у нотації IDEF1X зображає сутності, їхні атрибути та зв'язки між ними. Це дозволяє чітко візуалізувати структуру даних і взаємодію між різними сутностями. Основні сутності та реляційна модель даних що буде використовуватися для опису структури бази даних нашої системи зображено на рисунку 2.5.

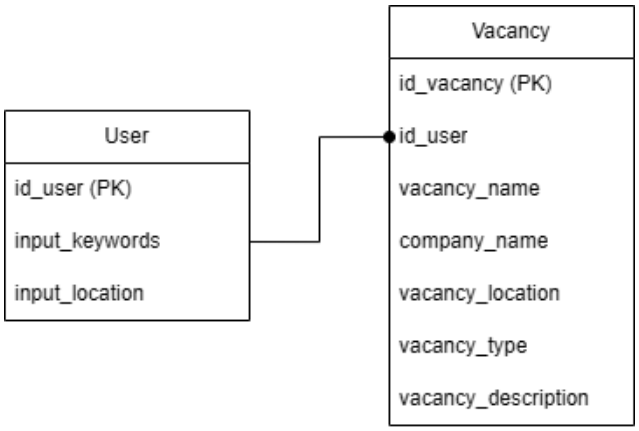

Рисунок 2.5 – Основні сутності та реляційна модель системи

Зв'язки між сутностями відображають процес рекомендацій, де один користувач може мати кілька рекомендованих вакансій, що дозволяє системі зберігати інформацію про всі можливі варіанти вакансій, які можуть бути запропоновані конкретному користувачеві.

Фізична модель даних визначає, як дані будуть фізично зберігатися в базі даних. Цей процес включає вибір конкретної бази даних, визначення структур зберігання даних та оптимізацію структури для забезпечення швидкого доступу до даних та ефективного використання ресурсів.

Для даної системи рекомендацій було обрано базу даних MongoDB. MongoDB є NoSQL базою даних, яка забезпечує гнучке зберігання даних та підтримку швидких операцій читання і запису. Це особливо важливо для системи

рекомендацій, яка повинна обробляти великі обсяги даних і надавати результати в режимі реального часу [10].

У MongoDB дані зберігаються у вигляді документів у колекціях. Кожен документ являє собою JSON-подібну структуру, яка може містити вкладені документи та масиви. Це забезпечує гнучкість у зберіганні складних структур даних [10]. Структура колекцій “User” та “Vacancy” представлено на рисунках 2.6 та 2.7 відповідно.

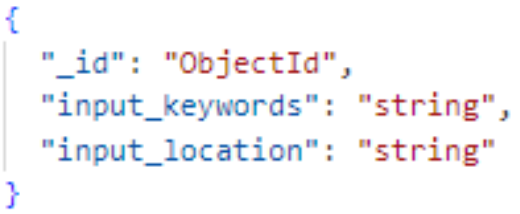

Рисунок 2.6 - Структура колекції “User”

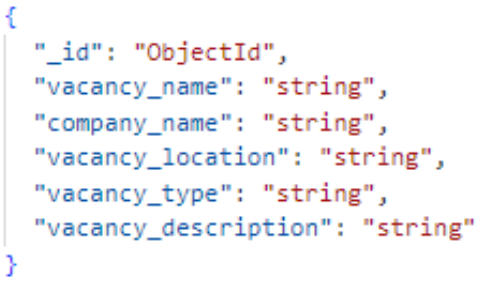

Рисунок 2.7 - Структура колекції “Vacancy”

Для забезпечення ефективного доступу до даних важливо оптимізувати структуру бази даних. Це включає створення індексів для часто використовуваних полів, таких як ключові слова для пошуку та локація. Індекси дозволяють швидко знаходити потрібні документи, що значно прискорює процес пошуку і рекомендацій.

Програмна реалізація бази даних включає налаштування MongoDB та інтеграцію з веб-додатком для забезпечення ефективного зберігання та доступу до даних [10]. Веб-додаток реалізовано за допомогою Flask, мікрофреймворку для Python, який забезпечує простий та гнучкий спосіб створення веб-додатків та інтеграції з базами даних [5].

Після встановлення MongoDB необхідно створити колекції для зберігання даних про користувачів та вакансії. Це можна зробити за допомогою команд в командному рядку або через інтерфейс MongoDBCompass.

Flask є легким мікрофреймворком для Python, який дозволяє швидко створювати веб-додатки. Для нашого веб-додатка Flask взаємодіє з MongoDB за допомогою бібліотеки pymongo, яка забезпечує зручний інтерфейс для роботи з базою даних [5, 14]. На рисунку 2.8 показано приклад коду для взаємодії з базою даних.

```
from flask import Flask, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

# Налаштування підключення до бази даних MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['job_recommendation']
user_collection = db['user']
job_collection = db['job']

# Завантаження даних з колекції вакансій
jobs_data = pd.DataFrame(list(job_collection.find()))

@app.route('/recommendations', methods=['POST'])
def recommendations():
    data = request.get_json()
    keywords = data['keywords']
    location = data['location']

    recommended_jobs = get_recommendations(keywords, location)
    return jsonify(recommended_jobs)

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

Рисунок 2.8 – Приклад коду для взаємодії з базою даних

3. ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Реалізація програмного забезпечення

Структурна схема програмної системи відіграє важливу роль у відображенні компонентів системи та їх взаємодії. Вона надає розробникам та інженерам чітке уявлення про архітектуру системи, дозволяючи їм зрозуміти, як окремі модулі працюють разом для досягнення спільної мети. У нашому випадку веб-додаток для рекомендації робочих вакансій з можливостями штучного інтелекту складається з кількох ключових модулів, кожен з яких виконує певну функцію. На рисунку 3.1 зображено структурну схему програмної системи.

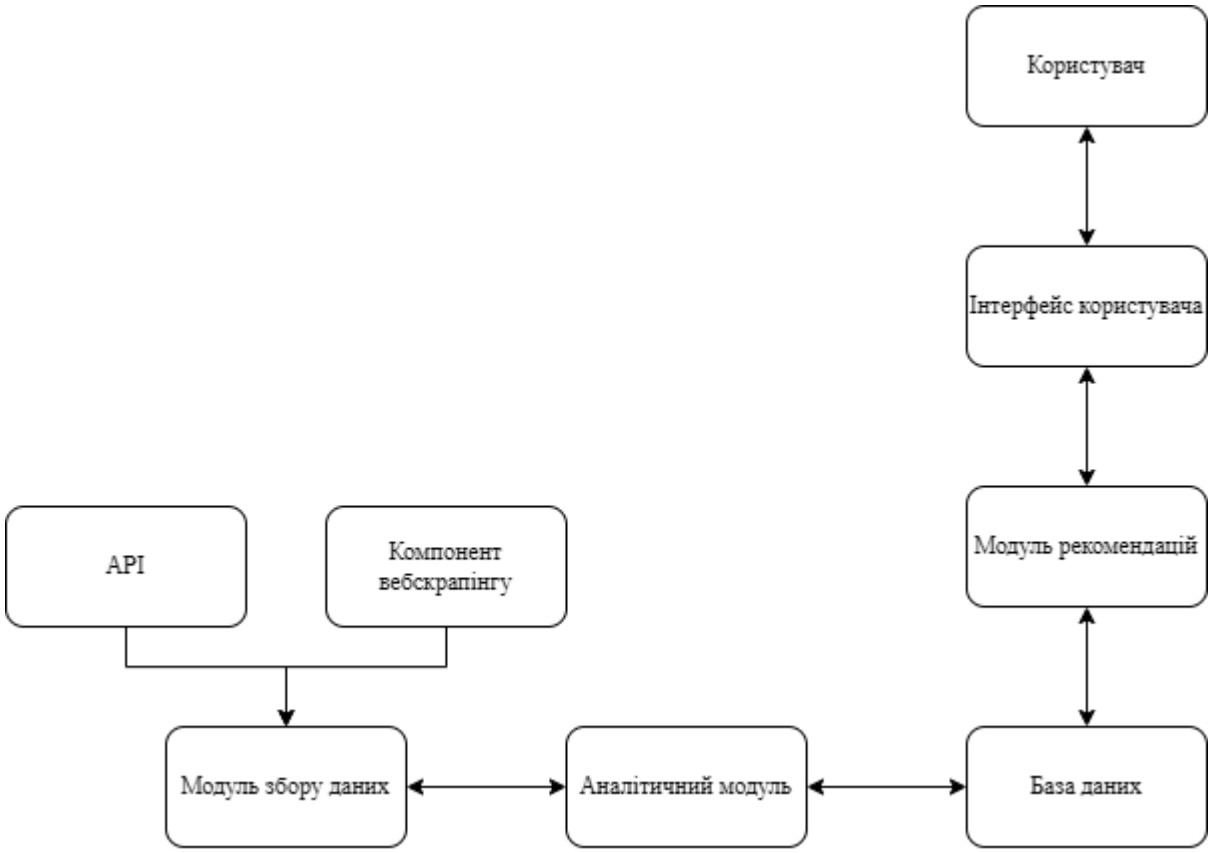

Рисунок 3.1 – Структурна схема програмної системи

Інтерфейс користувача є ключовим елементом системи, що забезпечує взаємодію користувачів з системою. Його основні функції включають:

- забезпечення інтуїтивно зрозумілого доступу до функцій системи: інтерфейс користувача дозволяє користувачам легко знаходити та фільтрувати вакансії, переглядати рекомендації та взаємодіяти з системою;
- адаптивний дизайн: Реалізація інтерфейсу на основі сучасних веб-технологій (HTML/CSS, JavaScript, React) забезпечує швидку та ефективну роботу з адаптивним дизайном для зручного доступу з різних пристроїв [4, 7, 16, 26].

Модуль рекомендацій безпосередньо взаємодіє з користувачами, надаючи їм персоналізовані рекомендації щодо вакансій. Його основні завдання включають:

- надання персоналізованих рекомендацій: Використовуючи введені користувачем ключові слова, історію пошуку та взаємодії з системою, модуль рекомендацій генерує список релевантних вакансій для кожного користувача;
- генерація списку релевантних вакансій: Аналізуючи дані про користувачів та вакансії, модуль рекомендацій створює персоналізовані пропозиції, які відображаються через інтерфейс користувача.

Модуль рекомендацій використовує фільтрування контенту для аналізу ключових слів і надання рекомендацій на основі змісту вакансій. Він адаптується до змін у поведінці користувачів і ринку праці, що забезпечує актуальність рекомендацій [8].

Аналітичний модуль є центральною частиною системи, яка відповідає за обробку та аналіз зібраних даних. Його основні функції включають:

- аналіз даних: Аналітичний модуль використовує алгоритми машинного навчання для обробки великих обсягів даних, виявлення тенденцій та взаємозв’язків. Це дозволяє створювати персоналізовані рекомендації для користувачів;
- класифікація вакансій: Вакансії класифікуються за різними параметрами, такими як галузь, тип зайнятості та місце розташування. Це допомагає структурувати дані та підвищити їхню корисність для користувачів.

Аналітичний модуль постійно оновлює моделі машинного навчання на основі нових даних, що дозволяє покращувати точність рекомендацій з часом.

Застосування методів кластеризації та регресійного аналізу забезпечує релевантність вакансій для конкретних користувачів[8].

Модуль збору даних є фундаментальною частиною системи, оскільки він забезпечує накопичення необхідної інформації для подальшої обробки та аналізу. Його основні завдання включають збір інформації про вакансії. Дані про вакансії збираються з різних джерел, таких як популярні сайти з пошуку роботи та корпоративні сайти компаній. Для цього використовуються технології веб-скрапінгу, які дозволяють автоматично витягнути дані з веб-сторінок. Також даний модуль інтегрується з API різних платформ для отримання актуальної інформації про вакансії в режимі реального часу. Це забезпечує ефективне агрегування інформації з багатьох джерел, підтримуючи її актуальність і повноту [13].

UML-діаграма класів відіграє важливу роль у моделюванні об'єктноорієнтованої архітектури програмної системи. Вона надає структурований спосіб візуалізації класів, їх атрибутів, методів та взаємозв'язків між ними. Це допомагає розробникам краще зрозуміти бізнес-логіку системи та забезпечує основу для подальшої реалізації програмного забезпечення. На рисунку 3.2 можна розглянути UML-діаграму класів.

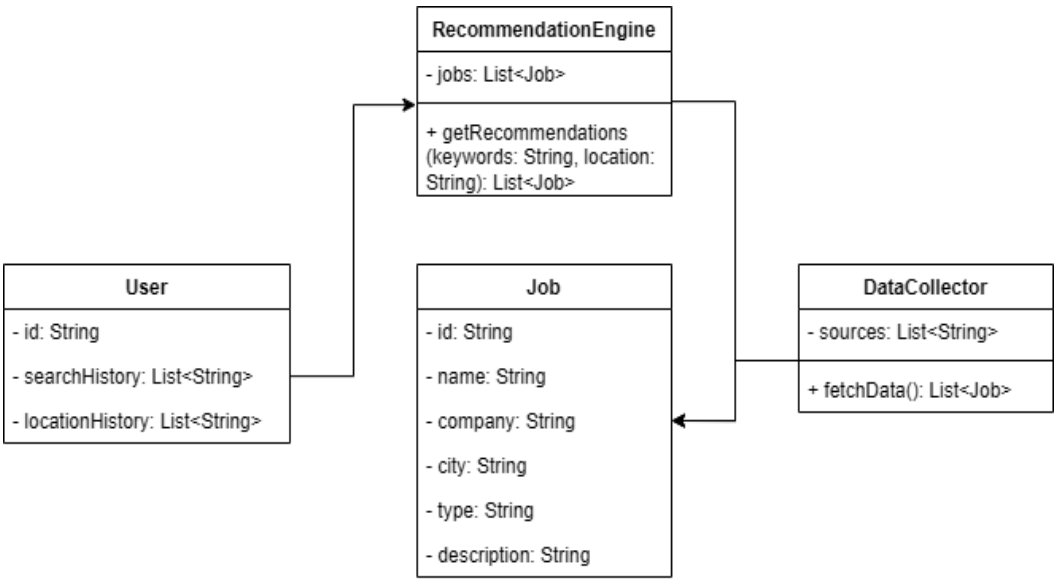

Рисунок 3.2 – UML-діаграма класів

Основні класи, що реалізують бізнес-логіку програмної системи, включають: Job, User, RecommendationEngine та DataCollector.

Клас Job представляє вакансію і містить наступні атрибути:

- id: Унікальний ідентифікатор вакансії (тип String);
- name: Назва вакансії (тип String);
- companyName: Назва компанії, що пропонує вакансію (тип String);
- city: Місто, де розташована вакансія (тип String);
- type: Тип зайнятості (тип String), наприклад, повна зайнятість, часткова зайнятість, стажування;
- description: Опис вакансії (тип String), що містить інформацію про обов'язки та вимоги до кандидата.

Клас User представляє користувача системи і містить наступні атрибути:

- id: Унікальний ідентифікатор користувача (тип String);
- searchHistory: Історія пошукових запитів користувача (тип List<String>), що містить ключові слова для пошуку вакансій;
- locationHistory: Історія пошукових запитів за локаціями (тип List<String>), що містить назви міст або регіонів.

Клас RecommendationEngine відповідає за генерацію рекомендацій на основі введених користувачем ключових слів та локації. Він містить наступні атрибути та методи:

- jobs: Список вакансій (тип List<Job>), що доступні для аналізу;
- getRecommendations(keywords: String, location: String): Метод, що приймає ключові слова та локацію, і повертає список рекомендованих вакансій (тип List<Job>).

Клас DataCollector відповідає за збір даних з різних джерел. Він містить наступні атрибути та методи:

- sources: Список джерел даних (тип List<String>), звідки збираються вакансії;
- fetchData(): Метод, що збирає дані з джерел та повертає список вакансій (тип List<Job>).

Опис взаємозв'язків:

RecommendationEngine: Взаємодіє з класом Job для отримання даних про вакансії та генерації рекомендацій для користувача. Використовує метод getRecommendations, щоб аналізувати ключові слова та локацію, введені користувачем, і знаходити найбільш релевантні вакансії.

DataCollector: Взаємодіє з класом Job для збору даних про вакансії з різних джерел. Використовує метод fetchData, щоб отримати актуальні вакансії та зберігати їх для подальшого аналізу.

User: Взаємодіє з класом RecommendationEngine для отримання персоналізованих рекомендацій на основі пошукових запитів та історії пошуку. Інформація про користувача, така як searchHistory та locationHistory, використовується для поліпшення точності рекомендацій. RecommendationEngine аналізує дані користувача та надає відповідні вакансії.

Загалом UML-діаграма класів надає чітке уявлення про структуру та взаємозв'язки між основними класами програмної системи. Вона допомагає розробникам зрозуміти, як кожен клас працює окремо та як вони взаємодіють між собою для забезпечення функціональності системи. Така візуалізація є ключовою для успішного проєктування та реалізації складних програмних систем.

UML-діаграма станів являє собою потужний інструмент для моделювання поведінки системи. Вона відображає можливі стани об'єкта під час його життєвого циклу та переходи між цими станами. У рамках нашої розробленої системи UML-діаграма станів допомагає візуалізувати взаємодію користувача з графічним інтерфейсом користувача (GUI) і показує, як система реагує на різні дії користувача. На рисунку 3.3 показана UML-діаграма станів.

Start - це початковий стан де користувач відкриває веб-додаток та основні елементи інтерфейсу, відображаються на екрані. Переходи з цього стану можливі після введення користувачем ключових слів для пошуку вакансій.

Search - користувач вводить ключові слова та локацію для пошуку вакансій. Після введення цієї інформації користувач натискає кнопку пошуку, що приводить до переходу в стан "Search Results", де відбувається пошук.

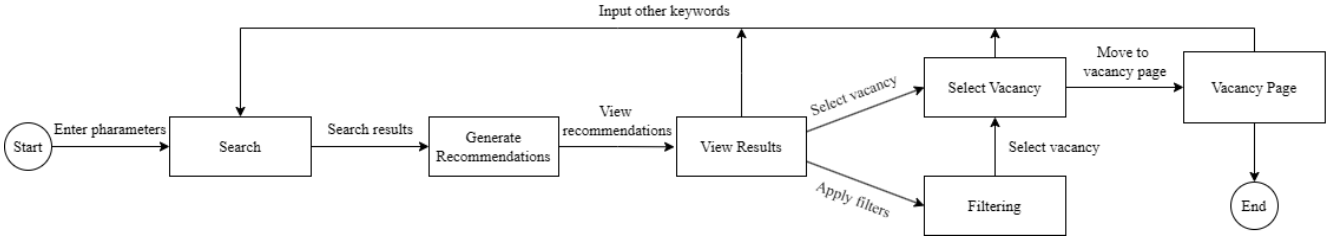

Рисунок 3.3 – UML-діаграма станів

Generate Recommendations - система генерує персоналізовані рекомендації на основі введених користувачем ключових слів та локації, а також на основі попередніх пошукових запитів та взаємодії з системою.

View Results - користувач переглядає результати пошуку або рекомендацій. Після цього він може обрати конкретну вакансію для перегляду деталей або застосувати фільтри для уточнення результатів.

Filtering - користувач застосовує фільтри до результатів пошуку для уточнення результатів. Після застосування фільтрів результати оновлюються, і користувач може обрати конкретну вакансію.

Select Vacancy - користувач обирає конкретну вакансію для перегляду деталей.

Vacancy Page - після вибору вакансії користувача перенаправляє на сторінку вакансії для перегляду повної інформації. Користувач може повернутися до списку результатів пошуку або до рекомендацій.

End - завершення процесу, коли користувач завершив взаємодію з системою.

UML-діаграма станів дає чітке розуміння поведінки системи в різних станах і переходах між ними. Це дозволяє розробникам і дизайнерам користувацького інтерфейсу зрозуміти, як система повинна реагувати на дії користувача, забезпечуючи інтуїтивно зрозумілий і ефективний інтерфейс. Це також допомагає виявити можливі проблеми та вдосконалення на етапі проєктування системи.

При розробці веб-застосунків важливо вибрати правильні програмні засоби реалізації, які можуть ефективно реалізувати функціональність, високу продуктивність і простоту використання. Зараз буде розглянуто обґрунтування вибору основних технологій для реалізації системи.

HTML5 (Hyper Text Markup Language) та CSS3 (Cascading Style Sheets) є базовими технологіями для створення веб-сторінок. HTML забезпечує структуру веб-сторінок, використовуючи теги для визначення різних елементів, таких як заголовки, абзаци, списки, форми та інші. HTML5 надає семантичні елементи, що допомагають поліпшити доступність і SEO-оптимізацію веб-додатків. CSS, своєю чергою, використовується для стилізації HTML-елементів, забезпечуючи привабливий та професійний вигляд веб-сторінок. CSS дозволяє створювати складні макети за допомогою Flexbox та Grid Layout, що робить інтерфейс адаптивним і зручним для користувачів. Використання цих технологій забезпечує основи для створення інтуїтивно зрозумілого та естетично привабливого інтерфейсу користувача [4, 7].

JavaScript є однією з основних мов програмування для веб-розробки, яка дозволяє створювати динамічні та інтерактивні веб-сторінки. Використання JavaScript у поєднанні з HTML і CSS дозволяє створювати багатофункціональні веб-додатки, які реагують на дії користувачів у режимі реального часу. Зокрема, JavaScript використовується для обробки подій, маніпулювання DOM (Document Object Model), валідації форм, а також для інтеграції з API і виконання асинхронних запитів. Це робить веб-додаток більш інтерактивним і забезпечує кращий користувацький досвід. Завдяки своїй гнучкості та потужності, JavaScript є невід'ємною частиною сучасної веб-розробки [4, 7, 13, 26].

ReactJS – це одна з найпопулярніших бібліотек JavaScript для створення користувацьких інтерфейсів, розроблена компанією Facebook. Вона дозволяє створювати динамічні та інтерактивні веб-сторінки з мінімальними зусиллями. Компонентний підхід ReactJS сприяє підвищенню читабельності та повторного використання коду, що спрощує підтримку та розвиток проєкт. Використання віртуального DOM значно покращує продуктивність додатка, оскільки оновлення компонентів відбувається швидше та ефективніше. Крім того, ReactJS має велику спільноту розробників та безліч ресурсів, що дозволяє швидко знаходити рішення для різних проблем [16, 26].

Python є високорівневою мовою програмування, яка відома своєю простотою, читабельністю та широкими можливостями. Він використовується для створення серверної частини додатка, забезпечуючи обробку даних, управління базами даних, а також реалізацію алгоритмів машинного навчання. Python має багато бібліотек та фреймворків, таких як Flask для створення веб-додатків, та Scikit-learn для реалізації алгоритмів машинного навчання. Завдяки своїй гнучкості та великій кількості інструментів, Python є ідеальним вибором для розробки складних веб-додатків з підтримкою штучного інтелекту [5, 15, 18].

Flask – це мікрофреймворк для Python, що забезпечує гнучкість та простоту у створенні веб-додатків. Його легкість та мінімалізм дозволяють розробникам швидко створювати веб-додатки без зайвих ускладнень, залишаючи повний контроль над архітектурою проекту. Flask легко інтегрується з різними бібліотеками та інструментами, що дозволяє розширювати функціональність додатка відповідно до потреб проекту. Добре задокументовані ресурси та активна спільнота сприяють швидкому розв'язанню проблем і допомагають розробникам отримувати необхідну підтримку [5].

Scikit-learn – це бібліотека для машинного навчання в Python, яка містить алгоритми для векторизації тексту та обчислення косинусної схожості. Вона надає широкий спектр алгоритмів для класифікації, регресії, кластеризації та зниження розмірності, що дозволяє розв'язувати різні задачі машинного навчання. Простий та інтуїтивно зрозумілий API Scikit-learn спрощує процес розробки моделей машинного навчання, а також добре інтегрується з іншими бібліотеками Python, такими як NumPy та Pandas, що дозволяє ефективно обробляти дані. Добре задокументовані ресурси та активна спільнота допомагають швидко розв'язувати проблеми та знаходити необхідну інформацію для реалізації алгоритмів машинного навчання [12, 18].

Pymongo – це офіційний клієнт MongoDB для Python, який забезпечує зручний та ефективний інтерфейс для взаємодії з базою даних MongoDB. Використання pymongo дозволяє розробникам легко підключатися до MongoDB, виконувати CRUD операції (створення, читання, оновлення, видалення) та

виконувати складні запити. Pymongo підтримує асинхронні операції, що покращує продуктивність додатка шляхом одночасного виконання декількох запитів. Інтеграція pymongo з Flask дозволяє створювати потужні та гнучкі серверні додатки, що забезпечують ефективне управління даними [14].

MongoDB – це документно-орієнтована NoSQL база даних, яка дозволяє ефективно зберігати та обробляти великі обсяги даних. Її гнучка структура даних у форматі BSON (бінарний JSON) дозволяє легко змінювати структуру даних без необхідності модифікації схем. MongoDB підтримує горизонтальне масштабування, що забезпечує обробку великих обсягів даних із високою продуктивністю. Крім того, підтримка реплікації та відмовостійкості забезпечує доступність даних навіть у випадку збою одного з серверів. Це робить MongoDB ідеальним вибором для зберігання та управління даними у веб-додатках, що потребують високої продуктивності та надійності [10].

Visual Studio Code (VSCode) – це безплатний редактор коду, розроблений компанією Microsoft, який став одним з найпопулярніших інструментів для розробників завдяки своїй гнучкості, продуктивності та багатофункціональності. VSCode підтримує безліч мов програмування та фреймворків завдяки розширенням, що дозволяє легко адаптувати його під будь-який проєкт. Вбудовані інструменти для налагодження, інтеграція з Git, а також підтримка IntelliSense (інтелектуальні підказки коду) значно покращують ефективність розробки. Завдяки своєму багатому набору функцій і розширень, VSCode є незамінним інструментом для розробників, що працюють над складними веб-додатками [25].

Git – це розподілена система керування версіями, яка дозволяє розробникам відстежувати зміни в коді, працювати над проєктами в команді та ефективно керувати розробкою програмного забезпечення. Git забезпечує можливість створення гілок для роботи над новими функціями або виправленням помилок без впливу на основний код, що дозволяє легко об'єднувати зміни після завершення роботи. Використання Git значно покращує процес розробки, дозволяючи зберігати історію змін, повертатися до попередніх версій коду та спільно працювати над проєктом у розподіленій команді. Інтеграція з платформами для керування

проєктами, такими як GitHub або GitLab, надає додаткові інструменти для спільної роботи, відстеження задач і автоматизації процесів [6].

Вибір програмних засобів для реалізації системи базувався на їхній здатності забезпечити високу продуктивність, гнучкість та зручність у використанні. HTML5 та CSS3 надають основи для створення інтуїтивно зрозумілого та привабливого інтерфейсу користувача. ReactJS додає динамічності та інтерактивності. Python забезпечує потужну основу для обробки даних та реалізації алгоритмів машинного навчання. Flask надає гнучкий та легкий бекенд, MongoDB дозволяє ефективно зберігати та обробляти дані, а Scikit-learn забезпечує потужні засоби для машинного навчання та аналізу даних. Pymongo надає зручний інтерфейс для взаємодії з базою даних, Visual Studio Code значно покращує процес розробки завдяки своїм багатофункціональним можливостям, а Git забезпечує ефективне керування версіями та спільну роботу над проєктом. Комбінація цих технологій дозволяє створити ефективну та надійну систему.

3.2 Інтерфейс користувача

Користувацький інтерфейс є своєрідним комунікаційним каналом, завдяки якому здійснюється взаємодія користувача й комп’ютера. Щоб створити ефективний інтерфейс, потрібно розуміти, які завдання вирішуватимуть користувачі за допомогою цієї програми та які вимоги до інтерфейсу можуть виникнути в користувачів.

Програма повинна допомагати користувачам виконувати свої завдання. Це означає, що інтерфейс має бути легким для освоєння й не створювати користувачеві перешкод. Інтерфейс повинен бути інтуїтивно зрозумілим, з чіткою навігацією та логічно організованим розташуванням елементів. Користувач має змогу швидко знайти потрібну функцію або інформацію без тривалого пошуку.

При роботі з програмою користувач не повинен відчувати дискомфорту. Для цього необхідно забезпечити перевірку результатів якомога більшої кількості "некоректних" дій користувача, але не робити це повсюдно. Це допоможе уникнути

помилок та неправильного використання програми, але не буде набридливим для користувача. Користувач повинен отримувати чіткі інструкції щодо своїх дій і отримувати інформаційні повідомлення лише тоді, коли це справді необхідно. Важливо також надати досвідченим користувачам можливість відключення виведення інформаційних повідомлень, щоб вони могли працювати ефективніше без зайвих відволікань.

Однією з основних функцій інтерфейсу є пошук вакансій. Дана функція зображена на рисунку 3.4.

```
const handleSearch = () => {
    const keywords = document.querySelector('.mainwords-input').value;
    const location = document.querySelector('.location-input').value;

    const request = { keywords, location };
    const updatedRequests = [...recentRequests, request];
    setRecentRequests(updatedRequests);
    localStorage.setItem('recentRequests', JSON.stringify(updatedRequests)); // Зберігаємо у localStorage

    axios.post('http://localhost:5000/recommendations', request)
        .then(response => {
            const recommendedJobs = response.data;
            setRecommendedJobs(recommendedJobs);
            setFilteredJobs(recommendedJobs);
            setShowJobsList(true); // Показуємо список вакансій після отримання рекомендацій
            if (recommendedJobs.length > 0) {
                setSelectedJob(recommendedJobs[0]); // Встановлюємо першу вакансію як вибрану після отримання рекомендацій
            }
        })
        .catch(error => {
            console.error('Error fetching recommendations:', error);
        });
};
```

Рисунок 3.4 – Функція пошуку вакансій

На рисунку 3.5 представлено як виглядає інтерфейс для введення ключових слів та місця розташування. Користувач вводить необхідні дані в поля пошуку і натискає кнопку для початку пошуку. Ця функція збирає ключові слова та місце розташування, створює запит, зберігає його в localStorage та надсилає на сервер для отримання рекомендацій. Отримані рекомендації встановлюються у стан, і відображається список вакансій.

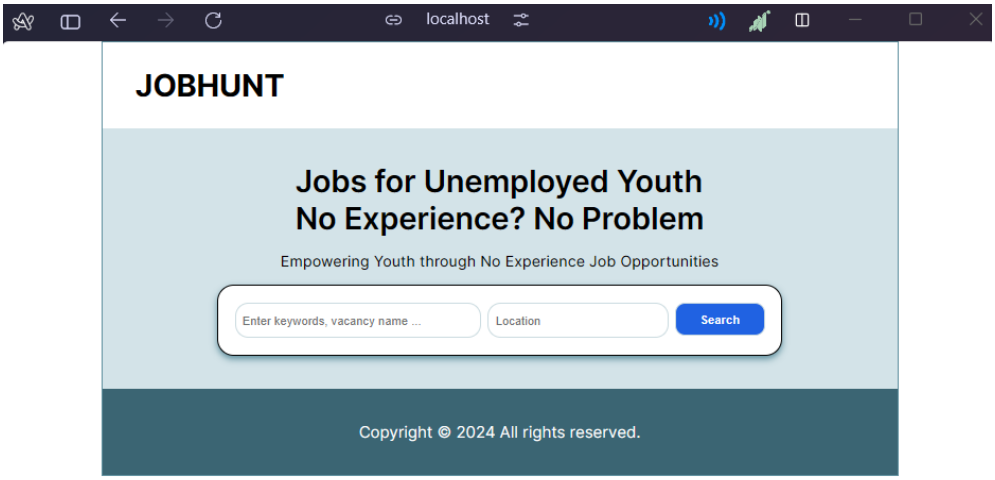

Рисунок 3.5 – Інтерфейс для введення ключових слів та місця розташування

На рисунку 3.6 представлено як сторінка виглядає після генерації рекомендацій.

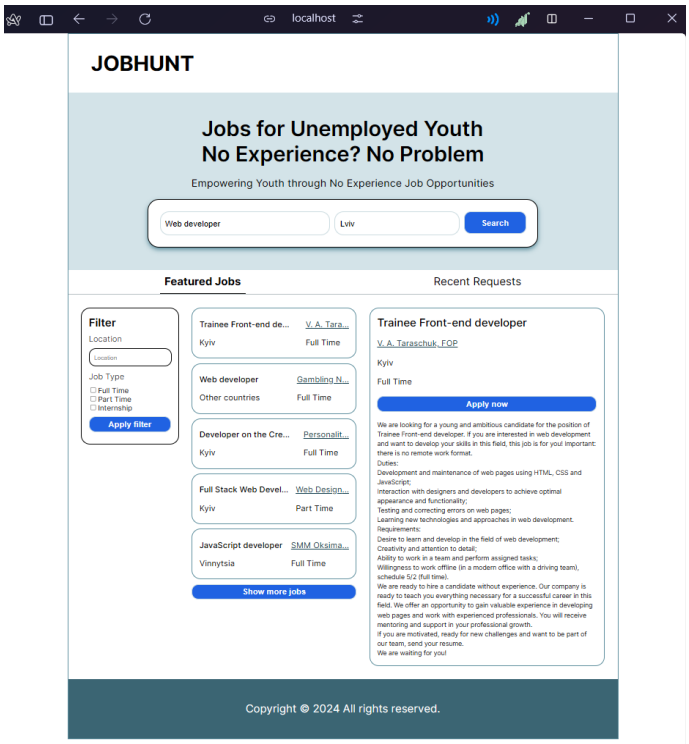

Рисунок 3.6 – Вигляд сторінки після генерації рекомендацій

Після отримання результатів пошуку користувач може переглянути "Recent Requests" (Останні запити). Це реалізовано за допомогою наступного коду (Рисунок 3.7), який дозволяє перемикатися між вкладками:

```
const handleTabClick = (tab) => {
    setActiveTab(tab);
};
```

Рисунок 3.7 – Функція перемикання між активними вкладками

На Рисунку 3.8 представлено вкладку "Recent Requests". Користувач може переглянути та за бажанням видалити попередні запити.

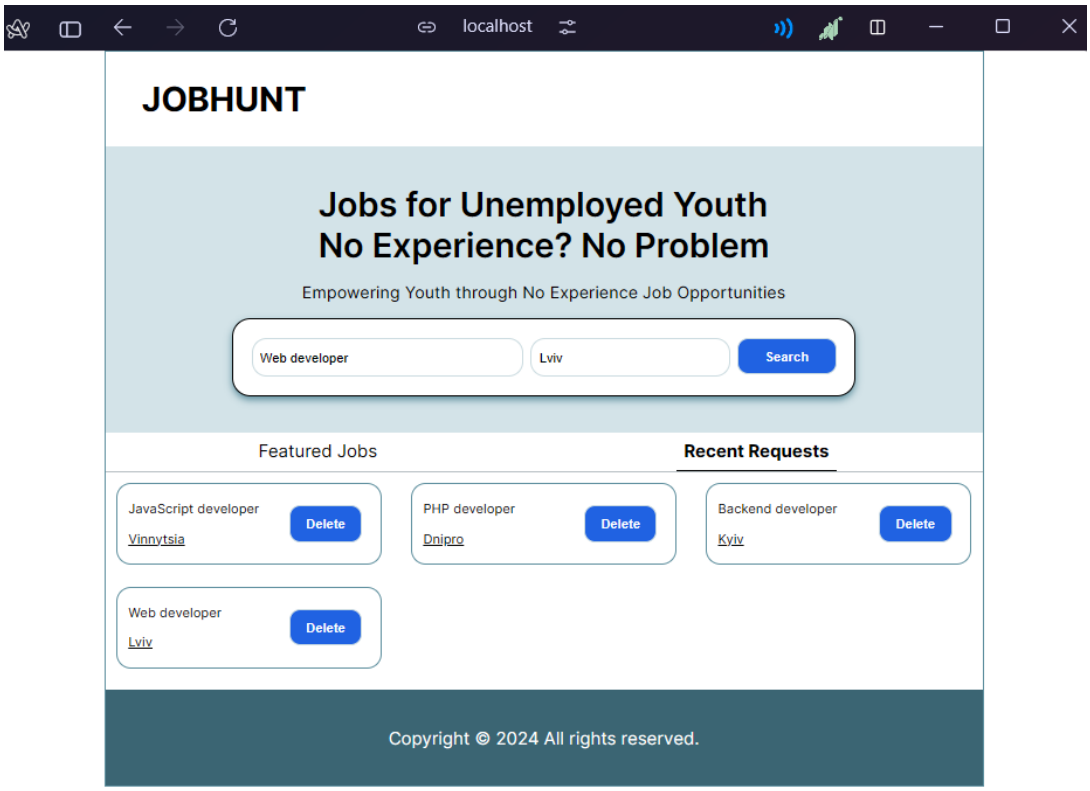

Рисунок 3.8 – Вигляд вкладки Recent Requests

Наступною важливою функцією є застосування фільтрів до списку вакансій. Функція реалізована за допомогою коду на рисунку 3.9.

```
const applyFilter = () => {
    const jobsToFilter = recommendedJobs.length > 0 ? recommendedJobs : jobsData;
    const filtered = jobsToFilter.filter(job => {
        // Фільтрація за типом роботи
        if (selectedJobTypes.length > 0 && !selectedJobTypes.includes(job.type)) {
            return false;
        }
        // Перевірка, чи було введено місто та фільтрація за ним
        if (selectedCity && selectedCity.trim() !== '' && job.city && job.city.toLowerCase() !== selectedCity.toLowerCase()) {
            return false;
        }
        return true;
    });
    setFilteredJobs(filtered);
    if (filtered.length > 0) {
        setSelectedJob(filtered[0]); // Встановлюємо першу вакансію як вибрану після фільтрації
    }
};
```

Рисунок 3.9 – Функція фільтрації вакансій

На рисунку 3.10 представлено блок для застосування фільтрів. Користувач може вибрати тип роботи та ввести місто для уточнення результатів пошуку. Ця функція фільтрує вакансії за типом роботи та містом, встановлюючи відфільтровані вакансії у стан. Якщо після фільтрації залишаються вакансії, перша з них встановлюється як вибрана.

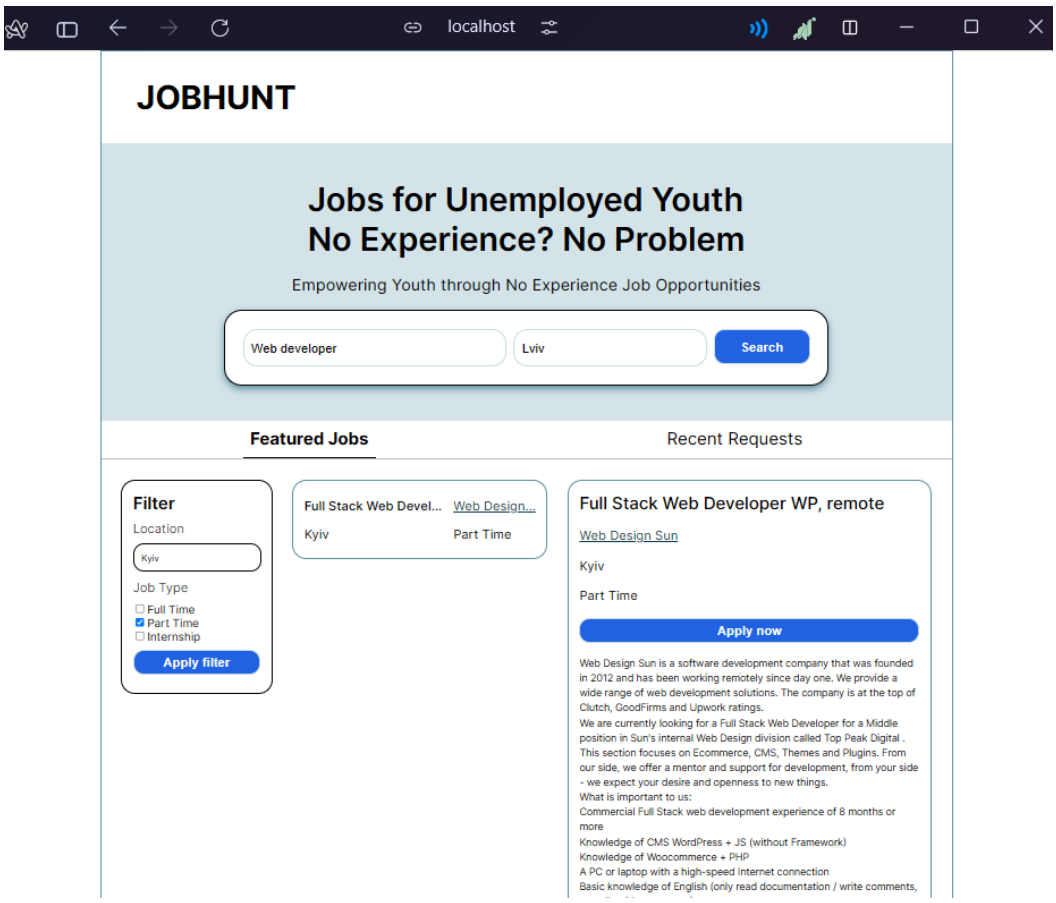

Рисунок 3.10 – Вигляд відфільтрованих вакансій

Також є функція переходу на сторінку вакансії, яка реалізована за допомогою наступного коду, який зображений на рисунку 3.11 для того, щоб користувач повноцінно ознайомився з вакансією.

```
const handleApplyNow = () => {
    if (selectedJob && selectedJob.url) {
        window.open(selectedJob.url, '_blank'); // Відкриває URL вакансії в новій вкладці
    }
};
```

Рисунок 3.11 – Функція переходу на початкову сторінку вакансії

На рисунку 3.12 представлено як користувач натиснув на кнопку "Apply now" для переходу на сторінку вакансії. Це відкриває URL вакансії в новій вкладці браузера, що дозволяє користувачу подати заявку на обрану вакансію.

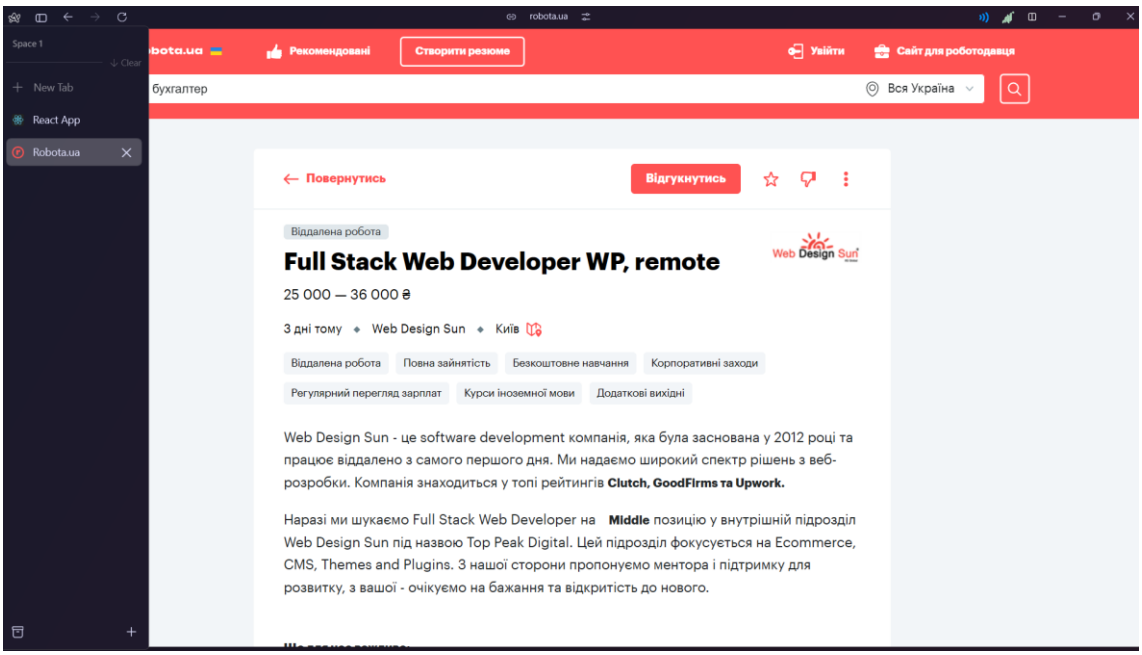

Рисунок 3.12 – Вигляд початкової сторінки вакансії

Загалом, система надає користувачеві можливість ефективно здійснювати пошук вакансій, застосовувати фільтри для уточнення результатів, переглядати деталі вакансій та отримувати персоналізовані рекомендації. Інтерфейс є інтуїтивно зрозумілим та забезпечує зручну взаємодію з системою.

3.3 Тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом у розробці будь-якої системи, оскільки воно дозволяє виявити та виправити помилки до випуску продукту. Для розробленої системи було обрано три основні типи тестування: тестування адаптивності інтерфейсу користувача та функціональне тестування [19, 24]. Ці методи забезпечують комплексну перевірку системи, дозволяючи переконатися в її надійності, функціональності та зручності використання. Нижче наведено детальний опис процедур тестування.

Тестування адаптивності інтерфейсу користувача забезпечує перевірку зручності використання та відповідності інтерфейсу дизайн-специфікаціям на різних пристроях і екранах, таких як мобільні телефони з шириною екрана 425px (рисунок 3.13), планшети з шириною екрана 768px (рисунок 3.14) та комп'ютери з шириною екрана від 1024px (рисунок 3.15). Це гарантує, що користувачі можуть зручно взаємодіяти з системою незалежно від пристрою, який вони використовують [24].

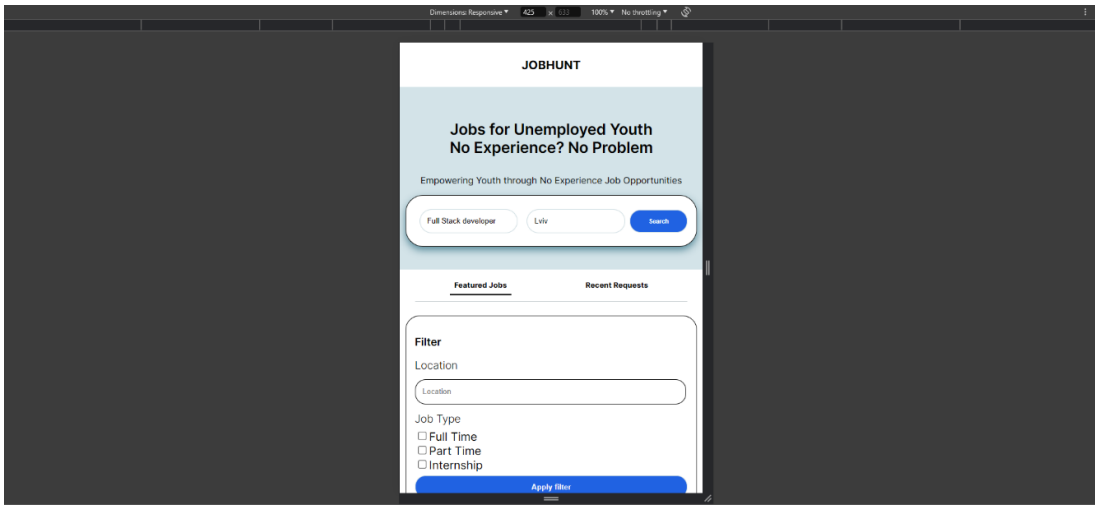

Рисунок 3.13 – вигляд додатка при ширині екрана 425px

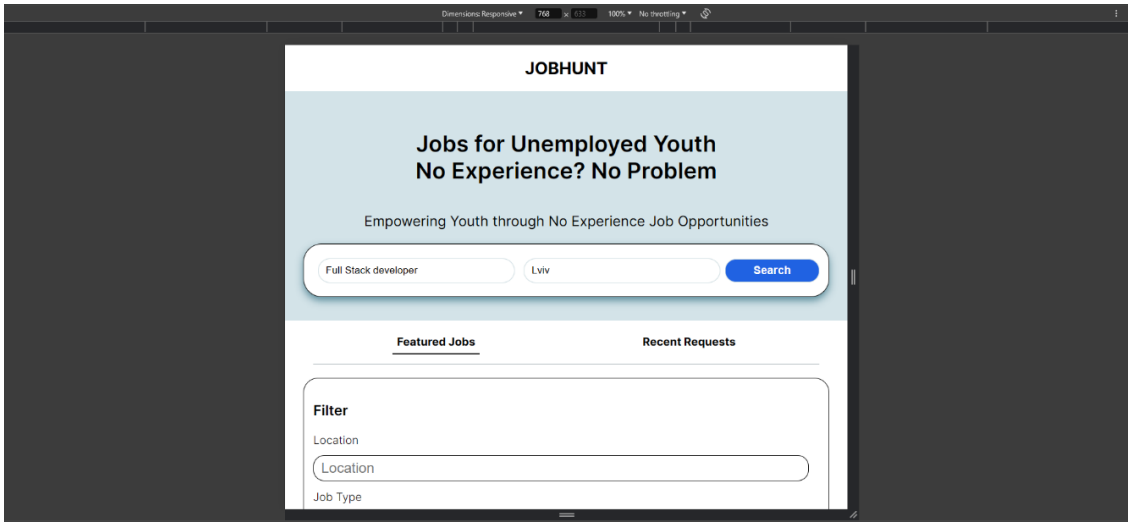

Рисунок 3.14 – вигляд додатка при ширині екрана 768px

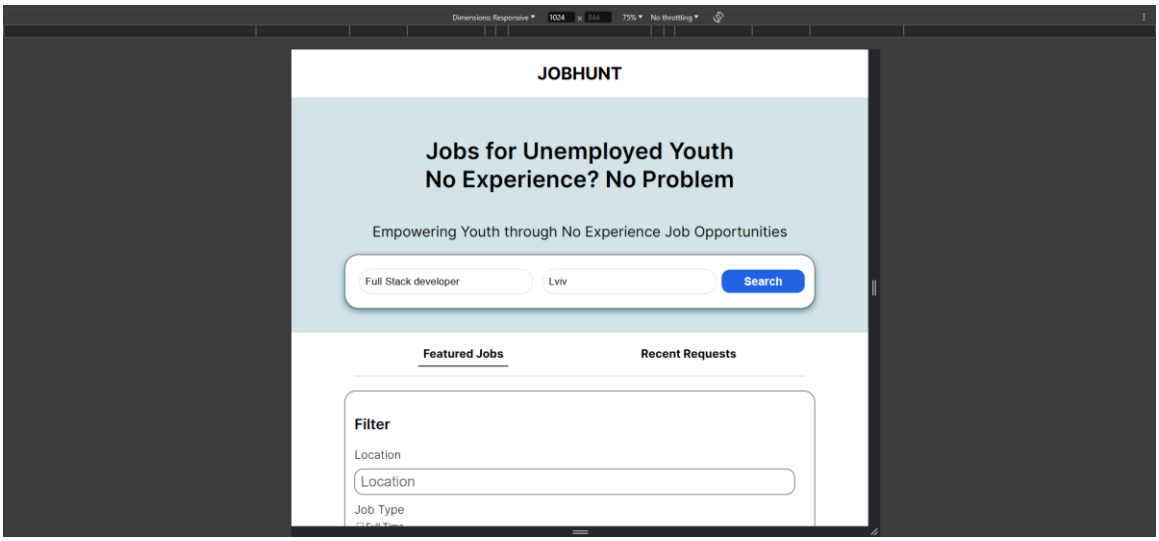

Рисунок 3.15 – вигляд додатка при ширині екрана 1024px

Як можемо бачити з даних рисунків, однакових параметрів та при зміні ширини екрана, всі елементи відображаються правильно без жодних помилок.

Функціональне тестування зосереджене на перевірці відповідності системи її функціональним вимогам [19]. Основні функції системи, сценарії тестування, очікувані результати та фактичні результати представлені у вигляді таблиці та підтверджено рисунками з результатами.

В таблиці 3.1 представлено перелік функціональних тестів, проведених для перевірки основних функцій. Тут міститься опис кожної функції, сценарій тестування, очікуваний результат та фактичний результат. Такий підхід дозволяє детально оцінити відповідність системи її функціональним вимогам.

Таблиця 3.1 – Перелік функціональних тестів

Функція	Сценарій тестування	Очікуваний результат	Фактичний результат
Генерація рекомендацій	Ввести ключові слова та локацію, натиснути кнопку "Search"	Система відображає список рекомендованих вакансій на основі введених даних та історії пошуку	Відповідає очікуваному результату
Застосування фільтрів	Обрати параметри фільтра (тип зайнятості, місто) та натиснути кнопку "Застосувати фільтр"	Система відображає відфільтрований список вакансій відповідно до обраних параметрів	Відповідає очікуваному результату (Рисунок 3.17)
Перегляд детальної інформації про вакансію	Натиснути на вакансію зі списку результатів пошуку або рекомендацій	Система відображає детальну інформацію про обрану вакансію	Відповідає очікуваному результату (Рисунок 3.18)
Перехід на початкову сторінку вакансії	Натиснути на кнопку “Apply now” на сторінці з детальною інформацією	Користувача переносить на початковий сайт, де ця вакансія була розміщена	Відповідає очікуваному результату (Рисунок 3.19)

На рисунку 3.16 зображено процес генерації рекомендацій. Користувач вводить ключові слова та місце розташування, натискає кнопку "Search", після чого система аналізує введені дані та історію пошуку, відображаючи список рекомендованих вакансій.

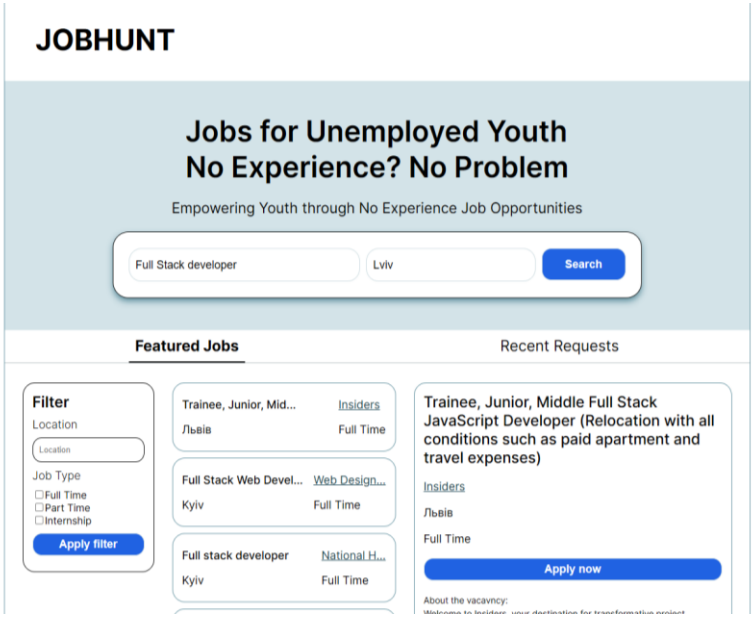

Рисунок 3.16 – Генерація рекомендацій

На рисунку 3.17 зображено застосування фільтрів. Користувач обирає параметри фільтра (тип зайнятості, місто) та натискає кнопку "Apply filter". Система відображає відфільтрований список вакансій, що відповідає обраним параметрам.

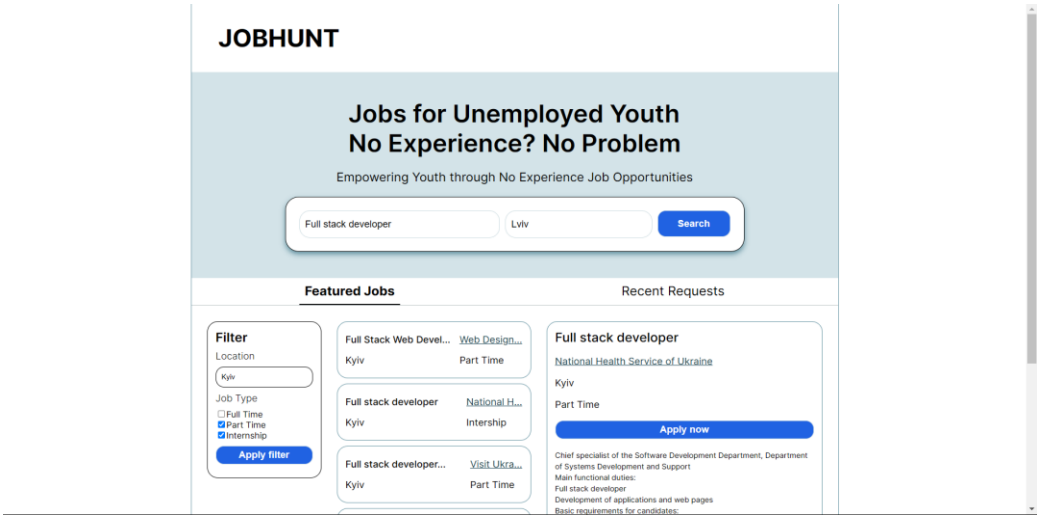

Рисунок 3.17 – Застосування фільтрів

На рисунку 3.18 відображено процес перегляду детальної інформації про вакансію. Користувач натискає на вакансію зі списку результатів, і система

відображає опис обраної вакансії, включаючи назву компанії, місто, тип зайнятості та інші деталі.

На рисунку 3.19 зображено функцію переходу на початкову сторінку вакансії. Після натискання на кнопку “Apply now” на сторінці з детальною інформацією про вакансію, користувач перенаправляється на оригінальний сайт, де ця вакансія була розміщена. Це дозволяє користувачу подати заявку на обрану вакансію безпосередньо на веб-сайті роботодавця.

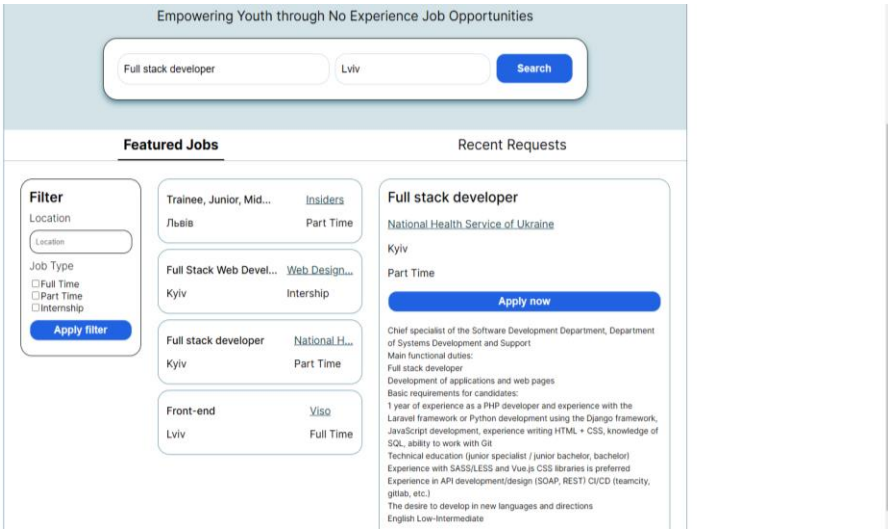

Рисунок 3.18 – Перегляд детальної інформації про вакансію

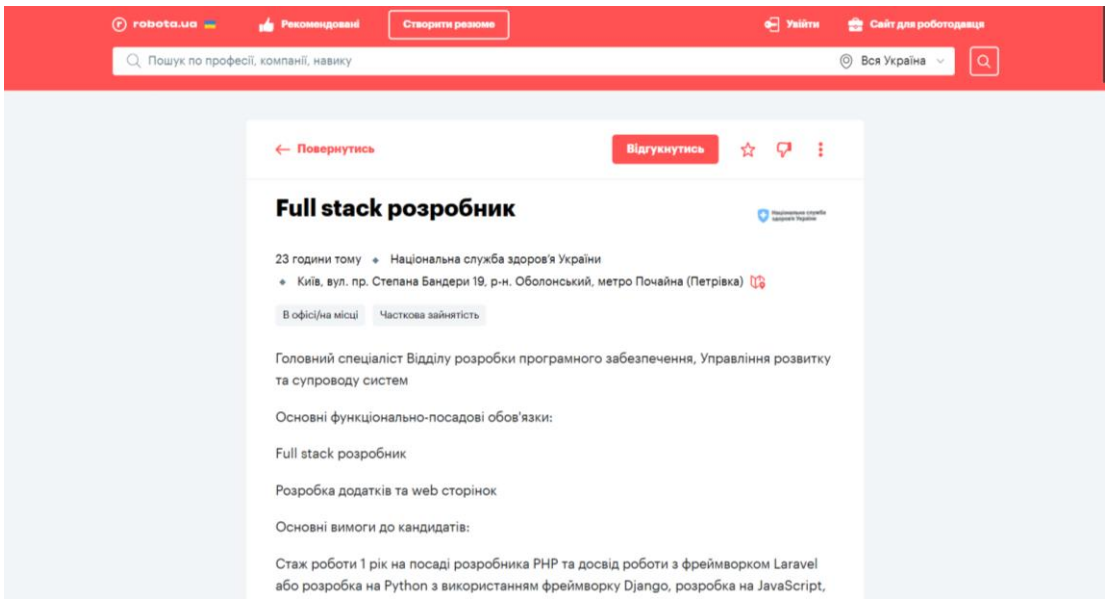

Рисунок 3.19 – Перехід на початкову сторінку вакансії

ВИСНОВКИ

Ця робота присвячена розробці веб-базованої системи рекомендації робочих вакансій, яка використовує можливості штучного інтелекту для підвищення ефективності та зручності процесу. Проведено аналіз існуючих систем, створено алгоритмічну та інформаційну підтримку, реалізовано програмно-технологічне забезпечення та проведено тестування розробленої системи.

У першому розділі було проведено детальний аналіз предметної області та огляд існуючих аналогів систем рекомендацій робочих вакансій. Основна увага приділялася розумінню поточних тенденцій та викликів у пошуку роботи онлайн. Було виявлено, що багато існуючих систем мають певні обмеження, такі як відсутність персоналізованих рекомендацій та інтеграції з різними джерелами вакансій. Це підкреслює необхідність створення нової системи, яка б об'єднувала ці функціональні можливості.

Другий розділ роботи присвячений розробці алгоритмічної та інформаційної підтримки системи. Було створено загальну структуру проектованої системи, розроблено алгоритми для персоналізованого підбору вакансій на основі ключових слів та інших параметрів. Також було реалізовано інформаційне забезпечення, включаючи опис вхідної та вихідної інформації, концептуальне інфологічне проектування та проектування глобальної датологічної моделі даних. Це забезпечило основу для ефективного функціонування системи.

У третьому розділі роботи розглянуто програмно-технологічне забезпечення системи, включаючи реалізацію програмного забезпечення та інтерфейс користувача. Було створено структурну схему програмної системи, UML-діаграми класів та станів, а також обґрунтовано вибір програмних засобів. Веб-система була реалізована з використанням сучасних технологій, таких як HTML/CSS, JavaScript, ReactJS, Python, Flask та MongoDB, що забезпечило його високу продуктивність та зручність використання.

Розроблена система продемонструвала свою ефективність та релевантність в умовах сучасного ринку праці. Проведені тестування підтвердили її здатність

надавати персоналізовані рекомендації та забезпечувати користувачам зручний інтерфейс для пошуку роботи. Ця робота має потенціал для подальшого розвитку, зокрема інтеграції додаткових функцій, таких як аналіз резюме та автоматичне заповнення заявок на вакансії. Пропонована система може стати важливим інструментом для шукачів роботи та роботодавців, підвищуючи ефективність та результативність процесу працевлаштування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні рекомендації до виконання кваліфікаційної роботи / [Комар М. П., Саченко А. О., Васильків Н. М. та ін.]. – Тернопіль: ЗУНУ, 2024. – 52 с.
2. Старих О. В. Веб-базована система рекомендацій робочих вакансій. *Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика. Математичні методи, моделі та інформаційні технології в економіці*: тези доп. Міжнар. наук.-практ. конф, м. Ізмаїл, 18 травня 2024 р. м. Ізмаїл: ЦФЕНД, 2024. С. 33-34.
3. Як працює ШІ? Принципи роботи сучасного AI [Електронний ресурс]. – URL: https://itproger.com/ua/news/kak-rabotaet-ii-printsipi-raboti-ovremennogo-ai (дата звернення: 20.05.2024)
4. CSS Introduction [Електронний ресурс]. – URL: https://www.w3schools.com/css/default.asp (дата звернення: 20.05.2024)
5. Flask Documentation [Електронний ресурс]. – URL: https://flask.palletsprojects.com/ (дата звернення: 20.05.2024)
6. Git Documentation [Електронний ресурс]. – URL: https://git-scm.com/doc (дата звернення: 20.05.2024)
7. HTML Introduction [Електронний ресурс]. – URL: https://www.w3schools.com/html/html_intro.asp (дата звернення: 20.05.2024)
8. How to Build an AI-based Recommendation System [Електронний ресурс]. – URL: https://www.leewayhertz.com/build-recommendation-system/ (дата звернення: 20.05.2024)
9. Indeed.com [Електронний ресурс]. – URL: https://www.indeed.com/ (дата звернення: 20.05.2024)
10. MongoDB Documentation [Електронний ресурс]. – URL: https://www.mongodb.com/docs/ (дата звернення: 20.05.2024)
11. Natural Language Processing (NLP) [A Complete Guide] [Електронний ресурс]. – URL: https://www.deeplearning.ai/resources/natural-language-processing/ (дата звернення: 20.05.2024)

12.Pandas Documentation [Електронний ресурс]. – URL: https://pandas.pydata.org/pandas-docs/stable/ (дата звернення: 20.05.2024)

13.Postman API Documentation [Електронний ресурс]. – URL: https://www.postman.com/api-platform/api-documentation/ (дата звернення: 20.05.2024)

14.Pymongo Documentation [Електронний ресурс]. – URL: https://pymongo.readthedocs.io/en/stable/ (дата звернення: 20.05.2024)

15.Python Documentation [Електронний ресурс]. – URL: https://docs.python.org/3/ (дата звернення: 20.05.2024)

16.React Documentation [Електронний ресурс]. – URL: https://react.dev/ (дата звернення: 20.05.2024)

17.Robota.ua [Електронний ресурс]. – URL: https://robota.ua/ (дата звернення: 20.05.2024)

18.Scikit-learn Documentation [Електронний ресурс]. – URL: https://scikit-learn.org/stable/ (дата звернення: 20.05.2024)

19.Software Testing: Functional Testing [Електронний ресурс]. – URL: https://www.geeksforgeeks.org/software-testing-functional-testing/ (дата звернення: 20.05.2024)

20.State Transition Diagrams. [Електронний ресурс] – Режим доступу: https://qamania.org/blog/state-transition/ (дата звернення: 20.05.2024)

21.TF-IDF [Електронний ресурс]. – URL: https://uk.wikipedia.org/wiki/TF-IDF (дата звернення: 20.05.2024)

22.The Art of Micro Frontends: Build websites using compositional UIs that grow naturally as your application / Florian Rappl. – Birmingham: Packt Publishing, 2021. – 356 с.

23.The Role of AI in Content Recommendation [Електронний ресурс]. – URL: https://aicontentfy.com/en/blog/role-of-ai-in-content-recommendation (дата звернення: 20.05.2024)

24.UI Testing Guide [Електронний ресурс]. – URL: https://www.browserstack.com/guide/ui-testing-guide (дата звернення: 20.05.2024)

25. Visual Studio Code Documentation [Електронний ресурс]. – URL: https://code.visualstudio.com/docs (дата звернення: 20.05.2024)

26. Web JavaScript Documentation [Електронний ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 20.05.2024)

27. What is a Data Flow Diagram. [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram (дата звернення: 20.05.2024)

28. Work.ua [Електронний ресурс]. – URL: https://work.ua/ (дата звернення: 20.05.2024)

ДОДАТОК А

Програмний код

Головний файл для відображення сторінки (index.js)

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import Header from './Header';
import Main from './Main';
import Footer from './Footer';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Header />
    <Main />
    <Footer />
  </React.StrictMode>
);

reportWebVitals();
```

Файл для відображення основного функціоналу (Main.js)

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './index.css';
import jobsData from './vacancy.json';

const Main = () => {
    const [visibleJobs, setVisibleJobs] = useState(5);
    const [jobs, setJobs] = useState([]);
    const [filteredJobs, setFilteredJobs] = useState([]);
```

```
    const [selectedJob, setSelectedJob] = useState(null);
    const [selectedJobTypes, setSelectedJobTypes] = useState([]);
    const [selectedCity, setSelectedCity] = useState('');
    const [recommendedJobs, setRecommendedJobs] = useState([]);
    const [showJobsList, setShowJobsList] = useState(false);
    const [activeTab, setActiveTab] = useState('featured');
    const [recentRequests, setRecentRequests] = useState([]);

    useEffect(() => {
        setJobs(jobsData);
        setFilteredJobs(jobsData);

        const savedRequests =
localStorage.getItem('recentRequests');
        if (savedRequests) {
            setRecentRequests(JSON.parse(savedRequests));
        }
    }, []);

    const handleShowMore = () => {
        setVisibleJobs(prevVisibleJobs => Math.min(prevVisibleJobs +
10, filteredJobs.length));
    };

    const handleJobClick = (jobId) => {
        const selectedJob = filteredJobs.find(job => job._id ===
jobId);
        setSelectedJob(selectedJob);
    };

    const handleApplyNow = () => {
        if (selectedJob && selectedJob.url) {
            window.open(selectedJob.url, '_blank');
        }
    };
```

```
    const applyFilter = () => {
        const jobsToFilter = recommendedJobs.length > 0 ?
recommendedJobs : jobsData;
        const filtered = jobsToFilter.filter(job => {
            if (selectedJobTypes.length > 0 &&
!selectedJobTypes.includes(job.type)) {
                return false;
            }
            if (selectedCity && selectedCity.trim() !== '' &&
job.city && job.city.toLowerCase() !== selectedCity.toLowerCase()) {
                return false;
            }
            return true;
        });
        setFilteredJobs(filtered);
        if (filtered.length > 0) {
            setSelectedJob(filtered[0]);
        }
    };

    const handleJobTypeChange = (e) => {
        const jobType = e.target.name;
        if (e.target.checked) {
            setSelectedJobTypes(prevTypes => [...prevTypes,
jobType]);
        } else {
            setSelectedJobTypes(prevTypes => prevTypes.filter(type
=> type !== jobType));
        }
    };

    const handleCityChange = (e) => {
        setSelectedCity(e.target.value);
    };

    const handleSearch = () => {
```

```
        const keywords = document.querySelector('.mainwords-
input').value;
        const location = document.querySelector('.location-
input').value;

        const request = { keywords, location };
        const updatedRequests = [...recentRequests, request];
        setRecentRequests(updatedRequests);
        localStorage.setItem('recentRequests',
JSON.stringify(updatedRequests));

        axios.post('http://localhost:5000/recommendations', request)
            .then(response => {
                const recommendedJobs = response.data;
                setRecommendedJobs(recommendedJobs);
                setFilteredJobs(recommendedJobs);
                setShowJobsList(true);
                if (recommendedJobs.length > 0) {
                    setSelectedJob(recommendedJobs[0]);
                }
            })
            .catch(error => {
                console.error('Error fetching recommendations:',
error);
            });
    };

    const handleTabClick = (tab) => {
        setActiveTab(tab);
    };

    const handleDeleteRequest = (index) => {
        const updatedRequests = recentRequests.filter((_, i) => i
!== index);
        setRecentRequests(updatedRequests);
```

```
        localStorage.setItem('recentRequests',
JSON.stringify(updatedRequests));
    };

    return (
        <main className="main">
            <div className="search-job-div">
                <h1 className="hero-title">Jobs for Unemployed Youth
<br />No Experience? No Problem</h1>
                <p className="hero-subtitle">Empowering Youth
through No Experience Job Opportunities</p>
                <div className="search-bar">
                    <input className="mainwords-input" type="text"
placeholder="Enter keywords, vacancy name ..." />
                    <input className="location-input" type="text"
name="location" id="location" placeholder="Location" />
                    <button className="search-button" type="button"
onClick={handleSearch}>Search</button>
                </div>
            </div>
            {showJobsList && Array.isArray(filteredJobs) && (
                <div className="jobs-list">
                    <div className="choose-job-list">
                        <h2 className={`tab ${activeTab ===
'featured' ? 'active' : ''}`} onClick={() =>
handleTabClick('featured')}>Featured Jobs</h2>
                        <h2 className={`tab ${activeTab === 'recent'
? 'active' : ''}`} onClick={() => handleTabClick('recent')}>Recent
Requests</h2>
                    </div>

                    {activeTab === 'featured' && (
                        <div className="featured-jobs">
                            <div className="filter">
                                <h3 className="filter-
text">Filter</h3>
```

```
                                <div className="location">
                                    <p className="filter-
subtext">Location</p>
                                    <input
                                        className="location-input-
bar"
                                        type="text"
                                        name="location"
                                        id="location"
                                        placeholder="Location"
                                        value={selectedCity}
                                        onChange={handleCityChange}
                                    />
                                </div>
                                <div className="job-type">
                                    <p className="filter-
subtext">Job Type</p>
                                    <div>
                                        <input
                                            className="filter-add-
text"
                                            type="checkbox"
                                            id="full-time"
                                            name="Full Time"

onChange={handleJobTypeChange}
                                        />
                                        <label htmlFor="full-
time">Full Time</label>
                                    </div>
                                    <div>
                                        <input
                                            className="filter-add-
text"
                                            type="checkbox"
                                            id="part-time"
```

```
                                        name="Part Time"
onChange={handleJobTypeChange}
                                    />
                                    <label htmlFor="part-
time">Part Time</label>
                                </div>
                                <div>
                                    <input
                                        className="filter-add-
text"
                                        type="checkbox"
                                        id="internship"
                                        name="Internship"
onChange={handleJobTypeChange}
                                    />
                                    <label
htmlFor="internship">Internship</label>
                                </div>
                            </div>
                            <button className="filter-apply-
button" onClick={applyFilter}>Apply filter</button>
                        </div>
                        <div className="jobs">
                            {filteredJobs.slice(0,
visibleJobs).map((job, index) => (
                                <div key={index} className="job"
onClick={() => handleJobClick(job._id)}>
                                    <div>
                                        <p className="vacancy-
name">{job.name.substring(0, 20)}{job.name.length > 20 ? '...' :
''}</p>
                                        <p className="vacancy-
location">{job.city}</p>
                                    </div>
```

```
                                        <div>
                                            <p className="company-
name">{job.companyName.substring(0, 10)}{job.companyName.length > 10
? '...' : ''}</p>
                                            <p className="vacancy-
type">{job.type}</p>
                                        </div>
                                    </div>
                                ))}
                                {visibleJobs < filteredJobs.length
&& (
                                    <button className="show-more"
onClick={handleShowMore}>Show more jobs</button>
                                )}
                            </div>
                            {selectedJob && (
                                <div className="job-details">
                                    <p className="details-
name">{selectedJob.name}</p>
                                    <p className="details-
company">{selectedJob.companyName}</p>
                                    <p className="details-
location">{selectedJob.city}</p>
                                    <p className="details-
type">{selectedJob.type}</p>
                                    <button className="details-
apply" onClick={handleApplyNow}>Apply now</button>
                                    <p className="details-
description">{selectedJob.description}</p>
                                </div>
                            )}
                        </div>
                    )}

                    {activeTab === 'recent' && (
                        <div className="recent-requests">
```

```
                            <div className="requests">
                                {recentRequests.map((request, index)
=> (
                                    <div key={index}
className="request">
                                        <div>
                                            <p className="request-
keywords">{request.keywords}</p>
                                            <p className="request-
location">{request.location}</p>
                                        </div>
                                        <button className="delete-
request" onClick={() => handleDeleteRequest(index)}>Delete</button>
                                    </div>
                                ))}
                            </div>
                        </div>
                    )}
                </div>
            )}
        </main>
    );
};

export default Main;
```

Функція збереження даних в Local storage

```
const updatedRequests = [...recentRequests, request];
setRecentRequests(updatedRequests);
localStorage.setItem('recentRequests',
JSON.stringify(updatedRequests));
```

Файл серверної частини додатку (server.py)

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

```
from flask_cors import CORS
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

app = Flask(__name__)
CORS(app)
jobs_data = pd.read_json('src/vacancy.json')

def get_recommendations(keywords, location):

    jobs_data['description'].fillna('', inplace=True)

    search_query = keywords + ' ' + location

    tfidf_vectorizer = TfidfVectorizer()

    tfidf_matrix =
tfidf_vectorizer.fit_transform(jobs_data['description'])

    cosine_similarities =
cosine_similarity(tfidf_vectorizer.transform([search_query]),
tfidf_matrix)

    df_cosine_sim = pd.DataFrame(cosine_similarities[0],
columns=['cosine_similarity'])
    df_cosine_sim = pd.concat([df_cosine_sim, jobs_data], axis=1)
    df_cosine_sim_sorted =
df_cosine_sim.sort_values(by='cosine_similarity', ascending=False)

    recommended_jobs = df_cosine_sim_sorted[['url', 'name',
'companyName', 'city', 'type', 'description',
'_id']].head(10).to_dict(orient='records')

    return recommended_jobs
```

```
@app.route('/recommendations', methods=['POST'])
def recommendations():
    data = request.get_json()
    keywords = data['keywords']
    location = data['location']

    recommended_job_ids = get_recommendations(keywords, location)
    print("Recommended job IDs:", recommended_job_ids)
    return jsonify(recommended_job_ids)

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

Файл для парсингу сайтів з роботою (vacancyParser.py)

```
import requests
import pymongo
from pymongo import MongoClient
import json

client = MongoClient('mongodb://localhost:27017/')
db = client['job_db']
collection = db['job_vacancies']

api_url = 'https://api.robota.ua/vacancies'

params = {
    'key': 'your_api_key',
    'count': 100
}

def fetch_and_store_vacancies():
    try:
        response = requests.get(api_url, params=params)
        response.raise_for_status()
        data = response.json()
```

```
        for vacancy in data['items']:
            job_info = {
                'name': vacancy.get('name'),
                'url': vacancy.get('link'),
                'companyName': vacancy.get('company',
{}).get('name'),
                'description': vacancy.get('description'),
                'category': vacancy.get('category', {}).get('name')
            }
            collection.update_one({'url': job_info['url']}, {'$set':
job_info}, upsert=True)

        print(f'Successfully stored {len(data["items"])} job
vacancies')

    except requests.exceptions.RequestException as e:
        print(f"Error fetching data from Robota.ua API: {e}")

fetch_and_store_vacancies()
```

ДОДАТОК Б

Апробація отриманих результатів

ЦФЕНД
ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
INTERNATIONAL SCIENTIFIC-PRACTICAL CONFERENCE

ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ ЕКОНОМІКИ, ОБЛІКУ, УПРАВЛІННЯ ТА ПРАВА: ТЕОРІЯ І ПРАКТИКА

PERSPECTIVE DIRECTIONS FOR THE DEVELOPMENT OF ECONOMICS, ACCOUNTING, MANAGEMENT AND LAW: THEORY AND PRACTICE

Збірник тез доповідей
Book of abstracts

18 травня 2024 р.
May 18, 2024

м. Ізмаїл, Україна
Izmail, Ukraine

ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
INTERNATIONAL SCIENTIFIC-PRACTICAL CONFERENCE

ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ ЕКОНОМІКИ, ОБЛІКУ, УПРАВЛІННЯ ТА ПРАВА: ТЕОРІЯ І ПРАКТИКА

PERSPECTIVE DIRECTIONS FOR THE DEVELOPMENT OF ECONOMICS, ACCOUNTING, MANAGEMENT AND LAW: THEORY AND PRACTICE

Збірник тез доповідей
Book of abstracts

18 травня 2024 р.
May 18, 2024

м. Ізмаїл, Україна
Izmail, Ukraine

The proposed IoT weather monitoring system aims to provide accurate, real-time weather data. Using data processing and visualization techniques the proposed system will offer valuable insights to users, contributing to better decision-making in different sectors like agriculture, transportation, and disaster management.

References

1. Holst A. IoT in Environmental Monitoring: Advancements and Applications // Journal of IoT and Sensor Networks. 2023.
2. Kumar V., Jain A. Machine Learning for Weather Prediction: An Overview // International Journal of Meteorology. 2022.
3. National Center for Atmospheric Research (NCAR). Weather Research and Forecasting (WRF) Model.
4. IBM Weather Company. Hyperlocal Weather Forecasting with IoT and AI // IBM Research. 2023.

УДК 004.8

Старих О. В.
студент IV курсу
Західноукраїнський національний університет

ВЕБ-БАЗОВАНА СИСТЕМА РЕКОМЕНДАЦІЙ РОБОЧИХ ВАКАНСІЙ

З розвитком комп'ютерних технологій та Інтернету пошук роботи стає все більш динамічним і складним процесом. Сучасний ринок праці характеризується величезною кількістю вакансій, що створює інформаційне перевантаження для шукачів роботи. Водночас рекрутери стикаються з проблемою відсіювання численних заявок, щоб знайти ідеальних кандидатів. У цих умовах інтеграція штучного інтелекту (ШІ) у веб-додатки для пошуку роботи стає вкрай актуальною [1, 2].

Метою цього дослідження є розробка веб-застосунка для пошуку роботи, доповненого можливостями ШІ. Використовуючи алгоритми ШІ, цей додаток має на меті спростити процес пошуку роботи як для шукачів. Завдяки використанню алгоритмів підбору, персоналізованих рекомендацій та розширених функцій пошуку, система спрямована на підвищення ефективності та результативності процесу пошуку роботи[3].

Основними завданнями цього проекту є розробка інтуїтивно зрозумілого і зручного інтерфейсу для користувачів, створення алгоритмів штучного інтелекту для підбору вакансій та надання персоналізованих рекомендацій, розширення можливостей пошуку для забезпечення точної фільтрації та сортування списків вакансій, а також оптимізація системи за показниками продуктивності, масштабованості та надійності.

Об'єктом дослідження є веб-додаток, який охоплює всі його функціональні можливості та взаємодію з користувачем. Предмет дослідження зосереджується на оптимізації процесів пошуку роботи за допомогою рішень на

основі ШІ. Для реалізації цього проекту будуть використані сучасні інструменти розробки, включаючи мови програмування HTML/CSS, Python та JavaScript, фреймворк React, середовище розробки VS Code, а також базу даних MongoDB.

Серед популярних веб-ресурсів для пошуку роботи в Україні можна виділити такі платформи, як Robota.ua [4], Work.ua [5] та міжнародний сайт Indeed.com [6]. Кожен з них має свої особливості, але жоден не пропонує повноцінної інтеграції штучного інтелекту для персоналізованих рекомендацій.

Веб-додаток, розроблений у рамках цього проекту, не лише допоможе користувачам ефективно знаходити вакансії, але й аналізуватиме їх взаємодію з системою, що дозволить покращувати алгоритми рекомендацій. Завдяки інтеграції ШІ, система буде здатна надавати персоналізовані рекомендації, що значно підвищить ефективність пошуку роботи як для шукачів, так і для роботодавців.

Список літератури

1. How to Build an AI-based Recommendation System [Електронний ресурс]. – URL: https://www.leewayhertz.com/build-recommendation-system/

2. The Role of AI in Content Recommendation [Електронний ресурс]. – URL: https://aicontentfy.com/en/blog/role-of-ai-in-content-recommendation

3. Як працює ШІ? Принципи роботи сучасного AI [Електронний ресурс]. – URL: https://itproger.com/ua/news/kak-rabotaet-ii-printsipi-raboti-sovremennogo-ai

4. Robota.ua. [Електронний ресурс]. – URL: https://robota.ua/.

5. Work.ua. [Електронний ресурс]. – URL: https://work.ua/.

6. Indeed.com. [Електронний ресурс]. – URL: https://www.indeed.com/.