Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка методу підбору та розподілу рекомендованих курсів працівникам IT-компанії
(тема)

Виконав:
студент 2 курсу, групи СПРм-22-1
Батраченко В.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Освітня програма Системне проєктування
(повна назва освітньої програми)

Керівник проф. Колесник Л.В
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Гребеннік І.В
(підпис) (прізвище, ініціали)

2024 р.

Кваліфікаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.

Матеріали кваліфікаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.

Попередній захист проведено 03 червня 2024 року.

Керівник кваліфікаційної роботи  *Л.В. Колесник*

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова

Освітня програма Системне проєктування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Батраченку Владиславу Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка методу підбору та розподілу рекомендованих курсів працівникам IT-компанії*
затверджена наказом університету від 01 квітня 2024 р. № 259 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 11 червня 2024 р.
3. Вихідні дані до роботи *Функція: Розробка методу підбору та розподілу рекомендованих курсів працівникам IT-компанії. Форма діалогу: веб-застосунок. Перелік використовуваних програмних засобів: ОС Windows, інтегроване середовище розробки IntellijIdea, Java, Spring, SpringBoot, Thymeleaf, MongoDB. Технічне забезпечення: комп'ютер з веб-браузером Google Chrome.*
4. Перелік питань, що потрібно опрацювати в роботі ***4.1*** *Аналіз предметної області.* ***4.2*** *Дослідження сфери.* ***4.3*** *Поняття рекомендаційної системи.* ***4.4*** *Аналіз задач багатокритеріальної оптимізації.* ***4.5*** *Аналіз аналогів систем з інтегрованим методом підбору та розподілу рекомендованих курсів працівникам IT-компанії.* ***4.6*** *Визначення сфери застосування інформаційної системи дистанційних курсів з програмування IT-компанії.* ***4.7*** *Постановка задачі.* ***4.8*** *Розробка схеми алгоритму та вимог до його етапів.* ***4.9*** *Розробка структури алгоритму.* ***4.10*** *Розробка етапів алгоритму.* ***4.11*** *Підготовка тестових наборів даних.* ***4.12*** *Проведення дослідження та отримання результату.* ***4.13*** *Аналіз отриманих результатів.* ***4.14*** *Проектування системи.* ***4.15*** *Розробка архітектури.* ***4.16*** *Розробка серверу бази даних.* ***4.17*** *Розробка карти сайту* ***4.18*** *Висновки.* ***4.19*** *Перелік джерел посилання.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *кресленики, схеми, плакати та/або комп'ютерні ілюстрації (слайди) на аркушах формату А4, що включаються до тексту пояснювальної записки або складу додатків: структура алгоритму; схема алгоритму; графік функцій; діаграма варіантів використання; діаграма класів; діаграма послідовності дій для прецеденту «перегляд рекомендаційних курсів»; реалізована архітектура; схема даних бази даних, що реалізована на платформі mongodb; логічна*

структура (карта) вебсторінок сайту; схеми навігації у додатках; схема моделі бази даних; екранні форми розроблених компонентів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	01.04.2024	Виконано
2	Аналіз завдання, літератури та аналогів з теми атестаційної роботи	03.04.2024	Виконано
3	Опрацювання літератури та аналіз об'єкту дослідження	04.04.2024	Виконано
4	Постановка та формалізація задачі	08.04.2024	Виконано
5	Дослідження методів та алгоритмів вирішення задачі. Розробка моделі.	09.04.2024	Виконано
6	Розробка архітектури взаємодії користувача та веб-додатку	15.04.2024	Виконано
7	Тестування розроблення програмного забезпечення та порівняння моделей	20.04.2024	Виконано
8	Оформлення пояснювальної записки	25.04.2024	Виконано
9	Подача кваліфікаційної роботи на допуск до захисту	01.06.2024	Виконано
10	Підготовка докладу до захисту кваліфікаційної роботи	03.06.2024	Виконано
11	Подання кваліфікаційної роботи	11.06.2024	Виконано

Дата видачі завдання _1_ ____квітня___2024 р.

Студент _____Батраченко В.О.__ _____
(підпис)

Керівник роботи _____ ____проф. Колесник Л.В_____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 80 с., 1 табл., 24 рис., 25 джерел інформації.

РЕКОМЕНДАЦІЙНА СИСТЕМА, МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ, МЕТОД ІДЕАЛЬНОЇ ТОЧКИ, JAVA, SPRING, SPRING BOOT, THYMELEAF

Об’єкт дослідження – процес підбору та розподілу курсів для підвищення кваліфікації співробітникам ІТ-компанії за допомогою рекомендаційних систем.

Предмет дослідження – методи рекомендаційних систем та методи прийняття рішень, такі як метод ідеальної точки, для підвищення ефективності підбору та розподілу курсів серед співробітників ІТ-компанії.

Мета роботи – використовуючи наявні методи рекомендаційних систем та прийняття рішень, розробити метод підбору та розподілу рекомендованих курсів для підвищення ефективності навчання співробітників.

Методи дослідження – методи рекомендаційних систем, методи прийняття рішень, розробка методу підбору та розподілу рекомендованих курсів, проведення тестування та оцінювання результатів, створення рекомендаційної системи з інтегрованим методом.

Наукова новизна – розроблений та інтегрований метод підбору та розподілу рекомендованих курсів працівникам ІТ-компанії, що підвищує якість та ефективність навчання співробітникам. Він дозволяє використовуючи дані про навики співробітника, дані проектів та курсів підібрати найбільш важливі курси. Також можна використовувати і частину алгоритму, наприклад перші 2 етапів для визначення найбільш необхідних навичок, що потребують покращення ІТ-працівнику. Крім різнобічного використання, в подальшому цей алгоритм можна змінити, додавши додаткові умови, що залежать від конкретної предметної області і оптимізувати для автоматизування будь-якого навчання, що робить цей метод універсальним та корисним.

ABSTRACT

Master’s Thesis: 80 pages, 1 tables, 24 figures, 25 sources.

RECOMMENDATION SYSTEM, DECISION-MAKING METHODS, IDEAL POINT METHOD, JAVA, SPRING, SPRING BOOT, THYMELEAF

The object of the study is the process of selection and distribution of courses for improving the skills of employees of an IT company using recommendation systems.

The subject of the study is the methods of recommender systems and decision-making methods, such as the ideal point method, to increase the efficiency of selection and distribution of courses among employees of an IT company.

The purpose of the work is to develop a method of selecting and distributing recommended courses to improve the effectiveness of employee training using existing methods of recommendation systems and decision-making.

Research methods - methods of recommender systems, decision-making methods, development of a method of selection and distribution of recommended courses, testing and evaluation of results, creation of a recommender system with an integrated method.

The scientific novelty of the result provides a developed and integrated method of selecting and distributing recommended courses to employees of an IT company, which increases the quality and effectiveness of training for employees. It allows you to select the most important courses using data on employee skills, project and course data. You can also use a part of the algorithm, for example, the first 2 stages to determine the most necessary skills that need to be improved for an IT employee. In addition to versatile use, in the future this algorithm can be changed by adding additional conditions depending on a specific subject area and optimized to automate any training, which makes this method universal and useful.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

РС –рекомендаційна система.

ПО – предметна область.

UML – Unified Modeling Language.

MVC – Model-View-Controller.

SQL – Structured Query Language.

NoSQL – Not Only SQL.

REST – Representational State Transfer.

HTTPS – Hypertext Transfer Protocol Secure.

JSON – JavaScript Object Notation.

ВСТУП

Світ кожного дня розвивається настільки швидко, що легко загубитися серед різноманітної кількості технологій. Для кожної IT-компанії важливо мати широкий спектр спеціалістів, які володіють достатнім рівнем знань у всіх актуальних технологіях на цей час. Враховуючи ці фактори, всі IT-компанії намагаються організувати курси для підвищення кваліфікації та надання необхідних знань для проєктів, щоб утримувати лідерство на ринку.

Зростає важливість підбору лише тих курсів, що є найбільш необхідними для співробітників. Серед основних критеріїв для аналізу необхідно виділити вимоги до проєктів на основі аналізу світового ринку, резюме працівника, дані курсу, особливо, кількість вільних місць. Алгоритм створюється на основі рекомендаційних систем. За допомогою цієї системи, IT-компанії зможуть мати кваліфікованих співробітників та велику кількість проектів, зайнявши першість на ринку IT та є актуальною задачею для розв’язання.

Метою даної роботи є розробка методу підбору та розподілу рекомендованих курсів працівникам IT-компанії.

Завданнями розробки є:

– аналіз предметної області;

– аналіз існуючих методів рекомендаційних систем та прийняття рішень;

– аналіз реалізованих аналогів;

– розробка методу підбору та розподілу рекомендованих курсів;

– тестування розробленого методу;

– проектування та розробка рекомендаційної системи з інтегрованим методом підбору та розподілу курсів між співробітниками IT-компанії.

На основі результатів розробки очікується створення ефективної та функціональної рекомендаційної системи з інтегрованим методом підбору та розподілу курсів, яка буде готова до тестування та впровадження в робоче середовище IT-компанії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження сфери

З кожним роком роль діджиталізації в житті людей має все більш вагомий вплив. Усе більше технологій інтегрується в різноманітні сфери життя та бізнесу. За рахунок своїх значних переваг, діджиталізація привертає увагу і стає стратегічним пріоритетом для багатьох компаній. Разом з використанням цифрових технологій створюється різноманітна кількість проєктів в IT-компаніях для яких потребується все більше спеціалістів. У той же час з розвитком технологій вимоги до IT-працівників зростають кожного року. Для отримання переваги у конкурентному середовищі, IT-компанії повинні залучати більше проєктів, але для цього необхідно мати компетентних працівників, що будуть відповідати вимогам за рівнем знань та володінням технологіями. Для цього більшість компаній мають свої ресурси для підвищення кваліфікації своїх співробітників. Але серед потужного потоку інформації та технологій в сучасному світі легко загубитись та тяжко визначити пріоритети, які ж технології необхідно вивчати працівникам у першу чергу. Через неправильний підхід, люди не матимуть достатньо навичок для залучення на проєкти, що приведе до втрати компаніями проєктів та прибутку [1]. Тому основною задачею постає створення рекомендаційної системи для підбору лише необхідних курсів своїм співробітникам для розвитку навичок на основі резюме та вимог до проєктів.

1.2 Поняття рекомендаційної системи

Для того, щоб рекомендувати курси працівникам, необхідно використовувати в алгоритмі критерії, які засновані на його резюме, та світовий ринок проєктів, для більш ефективного підбору.

Враховуючи тему дослідження, а саме розробка алгоритмів підбору та розподілу рекомендованих курсів, то рекомендаційні системи можна описати як,

інформаційні технології, що використовуючи аналіз даних та алгоритми, надають персоналізовані рекомендації курсів працівникам для найбільш оптимального покращення навичок [2]. За допомогою цієї системи, IT-компанії зможуть мати кваліфікованих співробітників та велику кількість проєктів, зайнявши першість на ринку IT.

Для аналізу розглянемо популярні онлайн ресурси, що використовують рекомендаційні системи:

– eBay (рис.1.1) – протягом багатьох років eBay, великий гравець у торговельній сфері, активно використовує інструменти машинного навчання та методи штучного інтелекту на різних рівнях свого бізнесу. Ці технології застосовуються для різних цілей, таких як персоналізований підбір товарів на головній сторінці, спеціальні пропозиції, оптимізація пошукової видачі та реклама. Завдяки алгоритмам рекомендацій вони досягають значних обсягів додаткових продажів, приносячи близько $1 млрд. щокварталу.

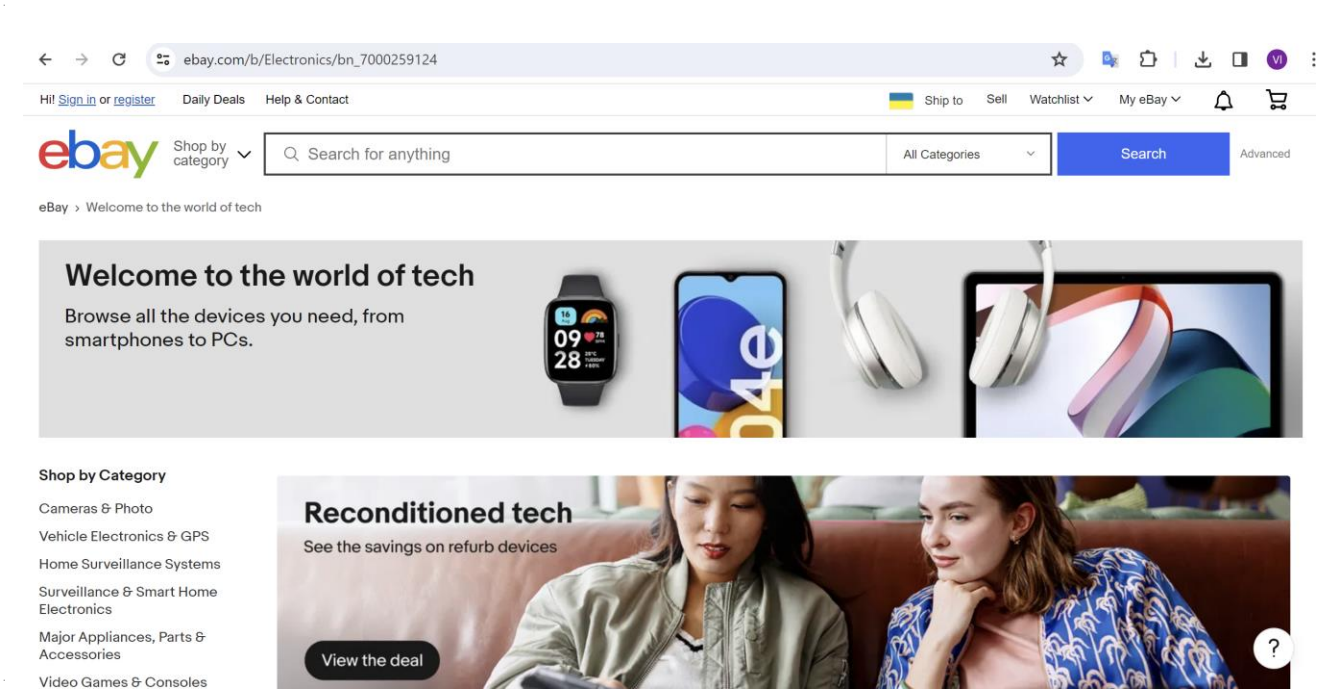

Рисунок 1.1 – Рекомендаційна система онлайн ресурсу «EBay»

Рекомендаційна система аналізує актуальний асортимент товарів на сайті та прогнозує, які товари найбільше сподобаються користувачу. Під час покупки

система враховує, щоб пропозиції не повторювалися та не перекривали товари, які вже знаходяться в корзині, застосовуючи для цього, серед іншого, технології розпізнавання зображень. Важливо, щоб рекомендації були пов'язані з поточними покупками настільки ефективно, щоб стимулювати інтерес користувача та спонукати його додати їх у корзину;

– LinkedIn (рис. 1.2) – платформа, орієнтована на бізнес, яка діє як соціальна мережа. Вона надає користувачеві рекомендації щодо знайомих осіб, робочих місць та компаній, що відповідають його професійним інтересам. Для формування цих рекомендацій використовується система колаборативного фільтрування, побудована на технології Apache Hadoop.

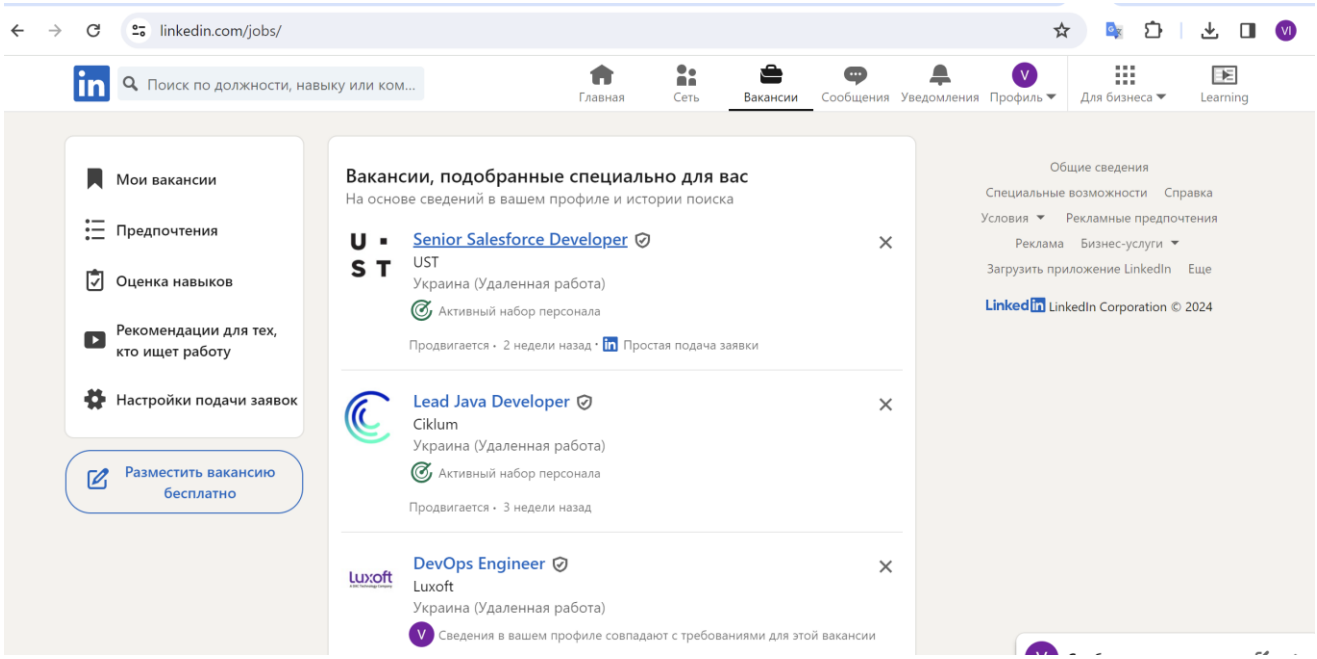

Рисунок 1.2 – Рекомендаційна система онлайн ресурсу «LinkedIn»

Основною ціллю рекомендаційних систем є надання користувачеві актуального контенту на основі його даних та збільшення прибутку компаній через підвищення лояльності клієнтів [3].

Рекомендаційні системи класифікуються наступним чином [4]:

– методи рекомендації – використовуються різні підходи, включаючи аналіз контенту, колаборативний фільтр та гібридні методи;

– джерела даних користувачів – враховуються як явні, так і неявні вподобання користувачів. Явні вподобання визначаються на основі дій, таких як відгук, оцінка, лайки. Неявні використовуються на основі історії перегляду, покупок;

– тип рекомендованих продуктів – наприклад контент (музика, відео, зображення, в таких соціальних мережах як Instagram, TikTok, YouTube), соціальні контакти (наприклад друзі в LinkedIn, Facebook), товари (наприклад в онлайн магазинах eBay, Comfy);

– динамічність рекомендації – бувають статичні та динамічні. Статичні рекомендації є сталими для всіх користувачів та не змінюються з часом. Динамічні оновлюються в реальному часі або періодично відповідно до актуальних інтересів користувача. Цей тип рекомендації можуть використовувати дані користувача лише для поточного сеансу (на основі сесії) або постійно (за допомогою авторизації користувача);

– масштаб системи – бувають спрямовані на одного користувача та колективні, які спрямовані на групу користувачів;

– мета рекомендації – підвищення продаж товарів, надання допомоги користувачу або інформації;

– тип системи – використовуються м’які та жорсткі системи. В м’яких рекомендації надаються з урахуванням нечіткості або невизначеності у вхідних даних. В жорстких, рекомендації надаються на основі чітких або точних вхідних даних.

1.3 Аналіз методів надання персоналізованих рекомендацій

Надання персоналізованих рекомендацій – це процес, який полягає у пропозиції користувачам вмісту, товарів або послуг, які відповідають їхнім індивідуальним потребам, вподобанням та інтересам. Виділяють три основі методи надання персоналізованих рекомендацій.

Перший підхід на підставі ознакових описів, називається content-based filtering. Він використовує інформацію про властивості об'єктів, наприклад, контент (музика, відео, зображення, в таких соціальних мережах як Instagram, TikTok, YouTube), соціальні контакти (наприклад друзі в LinkedIn, Facebook), товари (наприклад в онлайн магазинах eBay, Comfy) та вподобання користувачів, щоб зробити персоналізовані рекомендації. Основна ідея полягає у тому, щоб рекомендувати користувачам об'єкти, які подібні до тих, що їм сподобалися раніше, з урахуванням їхніх властивостей та характеристик.

Цей підхід складається з декількох етапів. Спочатку система аналізує характеристики об'єктів, наприклад такі дані, як назва, опис, метадані, ключові навички, категорія. Одним із прикладом аналізу, є представлення об’єкта у векторному просторі, де кожна координата відповідає певній характеристиці [5].

Наведемо приклад. Нехай є N об'єктів, які можуть бути представлені за допомогою M характеристик. Кожен об’єкт може бути представлений у векторному просторі O_i розмірності M, як $O_i = (o_{i1}, o_{i2}, \dots, o_{iM})$.

Схожість між двома об'єктами або користувачами може бути виміряна за допомогою різних метрик, таких як косинусна схожість (1.1) або евклідова відстань (1.2).

$$similarity(O_i, O_j) = \frac{O_i \cdot O_j}{\|O_i\| \cdot \|O_j\|}, \tag{1.1}$$

де $O_i \cdot O_j$ – скалярний добуток векторів;

$\|O_i\|$ та $\|O_j\|$ – їхні довжини (норми).

$$distance(O_i, O_j) = \sqrt{\sum_{k=1}^{M} (O_{ik} - O_{jk})}. \tag{1.2}$$

Далі збирається інформація про вподобання користувачів, наприклад, лайки, перегляд товарів, оцінки, корзина та покупки. Після аналізу

характеристик об'єктів і вподобань користувачів створюється їх профіль, який використовується для порівняння та надання рекомендацій. З математичної точки зору створення профілю це поєднання властивостей об'єктів у векторному просторі та розрахунок вагових характеристик об'єктів та користувачів, щоб визначити їхню відповідність.

Нехай O_i – вектор характеристик об’єкта i, а w_j – вага j-ої характеристики. Тоді профіль об'єкта i, позначений як P_i, можна обчислити як зважену суму його характеристик

$$P_k = \sum_{j=1}^{M} w_j * o_{ij},$$

де M – кількість характеристик об’єкта;

o_{ij} – значення j-ої характеристики для об’єкта i.

Вагові коефіцієнти w_j можуть бути встановлені вручну або обчислені за допомогою різних методів, таких як інформаційна важливість характеристик, заснована на частоті вживання, чи ж визначення на основі аналізу даних.

Потім обираються товари, що відповідають профілю користувача. Для цього необхідно визначити метод визначення схожості між об'єктами для подальшого вибору. Метод визначення схожості між двома об'єктами або користувачами може бути виміряна за допомогою різних метрик, таких як косинусна схожість (1.1) або евклідова відстань (1.2). Після обчислення схожості між користувачем та об'єктами (або між об'єктами), можна вибрати ті об'єкти, що мають найбільшу схожість з користувачем для рекомендації. Наприклад, можна вибрати k найбільш схожих об'єктів з користувачем за допомогою сортування за значенням схожості та вибрати перші l об'єктів для рекомендації.

У результаті користувачеві рекомендуються об'єкти, які максимально відповідають його вподобанням та інтересам на основі аналізу властивостей об'єктів та профілювання користувача.

Переваги контент-базованих методів включають простоту реалізації та здатність до роботи з невеликою кількістю даних. Однак ці методи можуть мати

обмежену потужність в порівнянні з іншими підходами, такими як колаборативна фільтрація, особливо коли недостатньо інформації про властивості об'єктів або вподобання користувачів.

Другий підхід називається колаборативною фільтрацією. Цей метод використовує інформацію про взаємодію користувачів з об'єктами для генерації персоналізованих рекомендацій. Основна ідея полягає у використанні історії взаємодії користувачів з об'єктами (наприклад перегляд товару, лайки, оцінки, додавання в кошик та покупки) для знаходження схожих користувачів або об'єктів та генерації рекомендацій на основі цих схожостей.

Основні підходи до колаборативної фільтрації включають:

– модель спільної фільтрації на основі користувачів (user-based collaborative filtering). Визначення схожості між користувачами на основі їхніх взаємодій з об'єктами, після чого йде рекомендація об'єктів, що сподобалися схожим користувачам, але ще не були переглянуті поточним користувачем;

– модель спільної фільтрації на основі об'єктів (item-based collaborative filtering). Визначення схожості між об'єктами на основі взаємодій користувачів з ними і рекомендація об'єктів, що схожі з тими, які вже сподобалися поточному користувачеві;

– засновані на побудові моделі вподобання – передбачається побудова моделі машинного навчання, яка враховує латентні (скриті) параметри користувачів та об’єктів.

Колаборативна фільтрація складається з декількох етапів. Спочатку йде створення матриці, де кожен елемент відображає взаємодію користувача з об'єктом (наприклад, рейтинг або відсутність взаємодії). Формалізуємо задачу колаборативної фільтрації. Нехай U — множина користувачів (users), I — множина об’єктів (items).

Інформація про відомі вподобання представлена у вигляді набору трійок:

$$D = \{(u, i, r_{ui})\}, (u, i) \in R,$$

де r_{ui} – кількісна величина вподобання об’єкта i користувачем u;

$R \subseteq U \times I$ – множина пар (користувач, об’єкт), про які відома ступінь вподобання.

Для подальшої зручності, введемо також позначення:

$R(u) = \{i : (u,i) \in R\}$ – множина об’єктів, суміжних з користувачем u;

$R(i) = \{u : (u,i) \in R\}$ – множина користувачів, суміжних з об’єктом i.

Приклад поданий в таблиці 1.1

Таблиця 1.1 – Матриця корисності

	Продукт 1	Продукт 2	Продукт 3	Продукт 4	Продукт 5
Користувач 1	1	2	3	5	
Користувач 2	2	4	4	1	
Користувач 3	1	5			5
Користувач 4		1	1	5	
Користувач 5	2		3		4

Далі йде розрахунок схожості між користувачами або об'єктами на основі їхніх взаємодій. Це може бути зроблено за допомогою різних метрик схожості, таких як косинусна схожість, Пірсона чи Жаккара.

Косинусна схожість була детально розібрана вище (1.1). Розглянемо коефіцієнт кореляції Пірсона. Цей коефіцієнт вимірює лінійну залежність між двома величинами. У контексті колаборативної фільтрації, кореляція Пірсона використовується для визначення ступеня схожості між взаємодіями користувачів або об'єктів. Формула кореляції Пірсона:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\ \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}.$$

Індекс Жаккара вимірює схожість між двома множинами, виражаючи

спільну кількість елементів відносно загальної кількості елементів у множинах. У контексті колаборативної фільтрації, індекс Жаккара може бути використаний для визначення ступеня схожості між взаємодіями користувачів або об'єктів. Він обчислюється як відношення кількості спільних взаємодій до загальної кількості унікальних взаємодій

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1.3}$$

де $|A \cap B|$ – кількість спільних елементів між множинами A та B;

$|A \cup B|$ – загальна кількість унікальних елементів у множинах A та B.

Індекс приймає значення від 0 до 1. Чим ближче він до 1, тим більше схожість між множинами A і B. Якщо обидві множини ідентичні, індекс Жаккара буде рівний 1. Якщо ж обидві множини не мають спільних елементів, індекс буде дорівнювати 0.

І на останньому етапі, використовується отримані значення схожості для прогнозування взаємодії користувача з новими об'єктами. Наприклад, можна вибрати об'єкти, що мають найвищу схожість з тими, які користувач вже взаємодіяв.

Третій підхід називається гібридним методом і поєднує в собі колаборативну фільтрацію та контент-базовані методи. Розділяють декілька видів гібридних методів:

– гібридна комбінація колаборативної фільтрації та контент-базованих методів. Об'єднання рекомендацій, отриманих з колаборативної фільтрації та контент-базованих методів, для покращення точності системи рекомендацій. Наприклад, можна використовувати колаборативну фільтрацію для ранжування рекомендацій, отриманих від контент-базованих методів, або навпаки;

– гібридна комбінація моделей. Використання кількох моделей машинного навчання або алгоритмів рекомендацій для створення комбінованої системи рекомендацій. Наприклад, можна використовувати нейронні мережі для

пошуку складних залежностей між користувачами та об'єктами, а потім використовувати ці результати разом з традиційними алгоритмами рекомендацій.

– гібридна комбінація фільтрації на основі змісту та заснованих на співробітництві. Комбінування контент-базованих методів (які використовують характеристики об'єктів) з методами колаборативної фільтрації (які використовують взаємодії користувачів та об'єктів). Наприклад, можна використовувати контент-базовані методи для побудови профілювання користувачів та об'єктів, а потім використовувати ці профілі для підбору рекомендацій засобами колаборативної фільтрації;

– гібридна комбінація експертних та автоматизованих методів. Використання як експертних, так і автоматизованих методів для генерації рекомендацій. Наприклад, система може використовувати експертні знання або правила для вибору певних категорій рекомендацій, а потім використовувати автоматизовані методи для персоналізації цих рекомендацій для кожного користувача.

Гібридні методи рекомендацій дозволяють враховувати різні аспекти та взаємозв'язки між користувачами та об'єктами, що може призвести до покращення точності, різноманітності та задоволення користувачів.

1.4 Проблеми рекомендаційних систем

Рекомендаційні системи допомагають користувачам знаходити вміст або продукти, в яких вони можуть бути зацікавлені, але вони також стикаються з рядом проблем, які можуть впливати на їхню ефективність та користувацький досвід. Є декілька найбільш поширених проблем, наприклад, проблема холодного старту, розрідженості, персоналізації, фільтрів пузирів, ефект статичності та проблема врахування контексту.

Проблема холодного старту виникає, коли рекомендаційна система не може зробити адекватні рекомендації для нових користувачів або нових об'єктів,

для яких немає достатньої кількості даних. Це може призвести до поганого персоналізованого досвіду для нових користувачів.

Для розв'язання проблеми холодного старту рекомендаційних систем можна використовувати стратегії, які включають надання базових рекомендацій на основі популярних об'єктів або категорій, залучення додаткової інформації від нових користувачів, використання контекстуальної інформації для адекватних рекомендацій, застосування гібридних методів, які поєднують різні підходи до рекомендацій, а також залучення досвідчених користувачів до надання рекомендацій для нових користувачів.

Рекомендаційні системи часто працюють з великими обсягами даних, і деякі користувачі можуть взаємодіяти лише з обмеженою кількістю об'єктів, що призводить до розрідженості матриці взаємодії. Це ускладнює розрахунок схожості між користувачами або об'єктами та знижує точність рекомендацій.

Для вирішення проблеми розрідженості можна застосовувати різні стратегії. Використання методів матричного розкладання для зменшення розмірності матриці взаємодії та отримання приблизних значень для нульових елементів. Це може допомогти у виявленні прихованих залежностей та покращенні точності рекомендацій. Додавання додаткової інформації про об'єкти (наприклад, описи, ключові слова, категорії) для зменшення розрідженості та покращення точності рекомендацій за допомогою контент-базованих методів.

Ідеальні рекомендації повинні бути індивідуалізованими для кожного користувача, але іноді системи можуть виявити складності у забезпеченні відповідних рекомендацій через обмежену інформацію про користувачів або об'єкти.

Використання алгоритмів машинного навчання, таких як класифікація або кластеризація, може допомогти виявити патерни в поведінці користувачів та створити персоналізовані моделі рекомендацій.

Рекомендаційні системи можуть піддаватися утворенню "пузирів фільтрів", де користувачі бачать лише контент або продукти, які відповідають

їхнім попереднім вподобанням. Це може призвести до обмеження диверсифікації інформації, яку бачать користувачі.

Забезпечення різноманітності в рекомендаціях, шляхом включення різних типів контенту або продуктів, які можуть бути цікавими для користувача. Це допомагає зменшити ефект "пузиркового фільтра" і розширює горизонти інформації, яку бачать користувачі.

Рекомендаційні системи можуть стати статичними в часі, не враховуючи динаміку інтересів користувачів та нових трендів у вмісті чи продуктах.

Для вирішення цієї проблеми необхідне постійне оновлення моделей рекомендацій за допомогою нових даних. Це може включати періодичне перетренування моделей або використання алгоритмів, які автоматично навчаються на нових даних з часом.

Багато рекомендаційних систем не враховують контекстуальні фактори, такі як місце, час, настрій, що може призвести до неправильних рекомендацій в залежності від конкретного контексту.

1.5 Аналіз задач багатокритеріальної оптимізації

Мета задачі прийняття рішень полягає у характеризації її частковими властивостями, а рівень досягнення цієї мети вимірюється їхніми кількісними значеннями. Таким чином, можна порівнювати рішення за досягнутим рівнем цих часткових властивостей. Формально часткові критерії представляються скалярними або векторними відображеннями $k(x): X \to Y,$, де X – множина прийнятних рішень, а Y – множина критеріїв. Вибір системи часткових критеріїв є неформалізованою, евристичною задачею. Її розв’язання ускладнюється необхідністю виконання таких, у деякому розумінні суперечливих, умов: з одного боку, набір критеріїв повинен достатньо повно характеризувати рішення; з іншого – містити якомога менше критеріїв. Крім того, різні критерії повинні враховувати різні якості. Узагальнені вимоги, в загальному випадку, суперечливі та не можуть бути задоволені одночасно. Тому, при формуванні

набору критеріїв у реальних задачах, доводиться йти на компроміси.

Математичну модель багатокритеріальної задачі прийняття рішень можна подати у вигляді такої оптимізаційної задачі [6]:

$$X^0 = A\,rg\,\underset{x\in X}{extr}\{k_1(x), k_2(x), \dots, k_n(x)\},$$

де $k_i(x), i = 1,2, \dots, n$ – частинні критерії;

X – множина допустимих рішень.

Нормалізація часткових критеріїв включає перетворення їх у однаковий масштаб та інтервал значень, щоб забезпечити їхню порівняльну придатність та стабільність у відношенні до екстремальних значень (максимуму або мінімуму). Формально це вирішується шляхом визначення єдиної шкали для всіх часткових критеріїв, що використовуються.

Розглянемо мінімаксне нормалізування. В цьому методі перетворюються значення критеріїв таким чином, щоб всі вони знаходилися в одному і тому ж діапазоні, зазвичай від 0 до 1. Цей метод використовується для того, щоб забезпечити рівність вагомості кожного критерію при прийнятті рішення в мультикритеріальних аналізах.

Процес мінімаксного нормалізування полягає у визначенні найкращого та найгіршого значень кожного критерію у наборі даних. Потім кожне значення критерію нормалізується за допомогою наступної формули [7]:

$$p_i(x) = \frac{k_i(x) - k_i^-}{k_i^+ - k_i^-},$$

де $k_i(x)$ – значення частинного критерію;

k_i^+, k_i^- – відповідно найкраще й найгірше значення частинного критерію $k_i(x)$ на області допустимих рішень $x \in X$.

Залежно від виду екстремуму (напряму домінування) маємо

$$k_i^+ = \begin{cases} \max k_i(x), & \text{якщо } k_i(x) \to max, \\ \min k_i(x), & \text{якщо } k_i(x) \to min; \end{cases}$$

$$k_i^- = \begin{cases} \min k_i(x), & \text{якщо } k_i(x) \to max, \\ \max k_i(x), & \text{якщо } k_i(x) \to min. \end{cases}$$

Цей процес забезпечує, що всі критерії будуть мати значення в діапазоні від 0 до 1, що полегшує порівняння їх між собою та прийняття рішень на основі цих даних.

Метод максимізації корисності з коефіцієнтами використовується для знаходження оптимального рішення з урахуванням вагових коефіцієнтів для кожного критерію. Нехай є адитивна функція корисності:

$$U(x) = \sum_{i=1}^{n} w_i * p_i(x),$$

де w_i – ваги критерію;

$p_i(x)$ – нормована корисність за критерієм.

Для максимізації цієї функції можна використовувати метод оптимізації, наприклад, метод градієнтного підйому чи метод Нелдера-Міда.

Класичний метод ідеальної точки передбачає побудову і використання ідеальної точки x, яка визначається як точка, що максимізує кожен критерій окремо. Оптимальне рішення x^0 обирається найближчим у вибраній метриці до ідеальної точки. Нехай $\rho(y, \hat{y}) = \{\sum_{i=1}^{n} |p_i(x) - \hat{p_i}|^s\}^{\frac{1}{s}}$, $s \geq 1$, тоді оптимальне рішення x^0 знаходиться як $Arg\min_{x \in X} \rho(y, \hat{y})$.

Для отримання кращих результатів, необхідно поєднати два методи - максимізацію корисності з ваговими коефіцієнтами та метод ідеальної точки. Спочатку визначаються критерії та їх вагові коефіцієнти, після чого формується функція корисності для кожного критерію, яка враховує відстань до ідеальної точки. Ці функції об'єднуються в адитивну функцію корисності з ваговими коефіцієнтами. Далі використовується метод мінімізації цієї адитивної функції корисності для отримання оптимального рішення. Комбінація обох методів

дозволяє краще враховувати важливість кожного критерію та забезпечити більш точне визначення оптимального рішення. Тоді отримає таку формулу:

$$\rho(y, \hat{y}) = \{\sum_{i=1}^{n} w_i |p_i(x) - \widehat{p_i}|\}, s \geq 1,$$

оптимальне рішення x^0 знаходиться як $Arg\min_{x \in X} \rho(y, \hat{y})$.

1.6 Аналіз аналогів систем з інтегрованою системою для підбору та розподілу рекомендованих курсів працівникам IT-компанії

Було проаналізовано існуючі системи з інтегрованою системою для підбору та розподілу рекомендованих курсів працівникам IT-компанії. Серед компаній, які пропонують свої курси підвищення кваліфікації для робітників є такі компанії-гіганти як Epam [8], GlobalLogic, SoftServe, Luxsoft. Усі перелічені компанії є одними із найбільших IT-компаній та складають основу IT-ринку праці на території України. Тому аналіз саме даних систем, як кращих, дозволить сформувати загальне враження про системи даного плану.

Інтерфейс застосунку IT-компанії «Epam Systems» зображений на рис. 1.4.

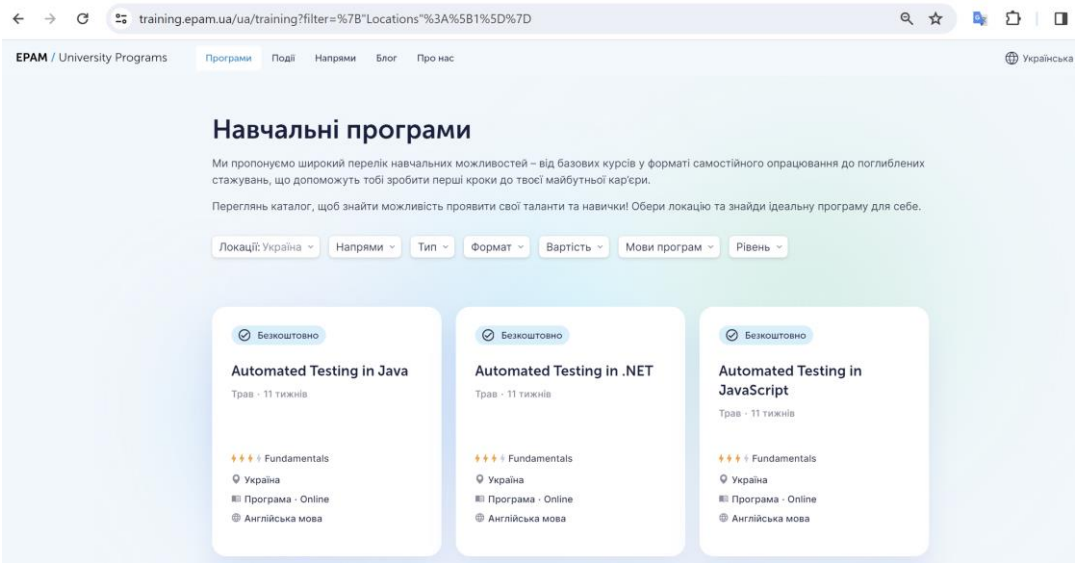

Рисунок 1.3 – Інтерфейс застосунку IT-компанії «Epam Systems»

На рисунку 1.3 зображено навчальні програми для старту в IT-компанії EPAM, не тільки для співробітників, а і для всіх бажаючих. Так як ніякі дані не було представлено, то був виведений список всіх доступних курсів. Це звичайна фільтрація без використання рекомендаційних систем, через що користувач повинен переглянути кожний курс, і сам визначитись, на що може потребуватися багато часу. В компанії є тест (рис. 1.4), після проходження якого буде результат (рис. 1.5).

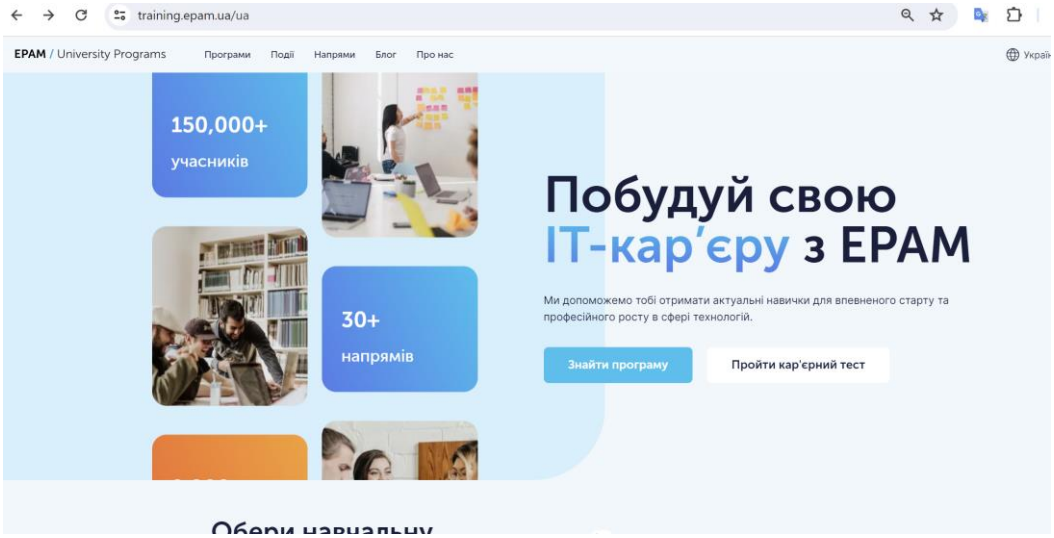

Рисунок 1.4 – Інтерфейс застосунку для проходження тесту IT-компанії «Epam Systems»

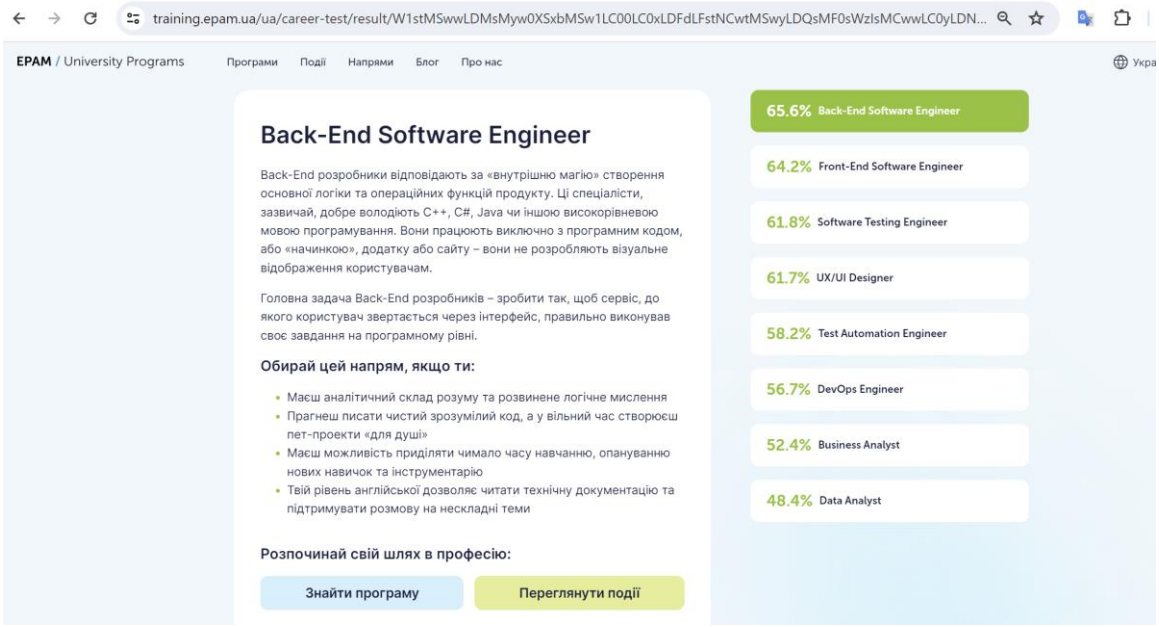

Рисунок 1.5 – Результат проходження тесту

Але це ніяк не відноситься до рекомендаційної системи і не допоможе оптимізувати. Тому розглянемо ще одну систему, яку пропонує компанія «Epam» (рис. 1.6)

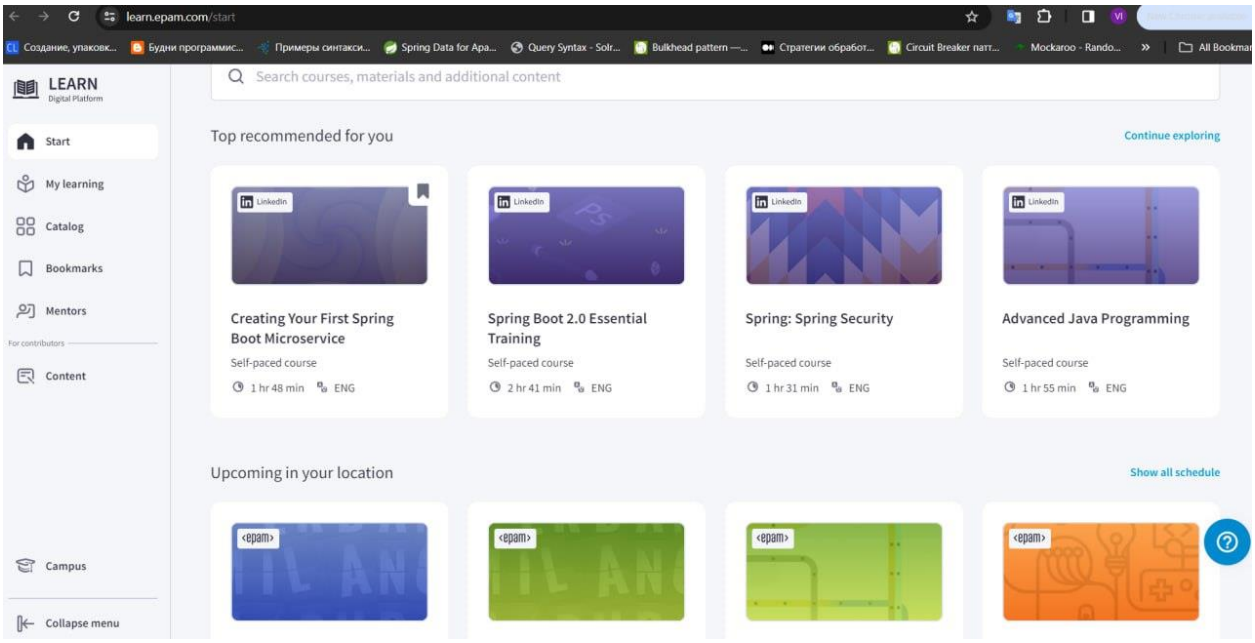

Рисунок 1.6 – Інтерфейс застосунку IT-компанії «Epam Systems» з рекомендаційною системою

У цій рекомендаційній системі, яка аналізує резюме працівника та вибірку курсів, для підбору навчальних матеріалів, основним недоліком є відсутність врахування вимог проєктів до працівників, які постійно змінюються. Це призводить до того, що процес вибору та розподілу курсів для підвищення кваліфікації не є автоматизованим, і співробітникам доводиться самостійно шукати та аналізувати інформацію. Для вирішення цих проблем необхідно переглянути підхід та вдосконалити алгоритм рекомендацій.

Переглянемо інші функції існуючої системи. На рисунках 1.7 та 1.8 зображено ручна фільтрація для пошуку.

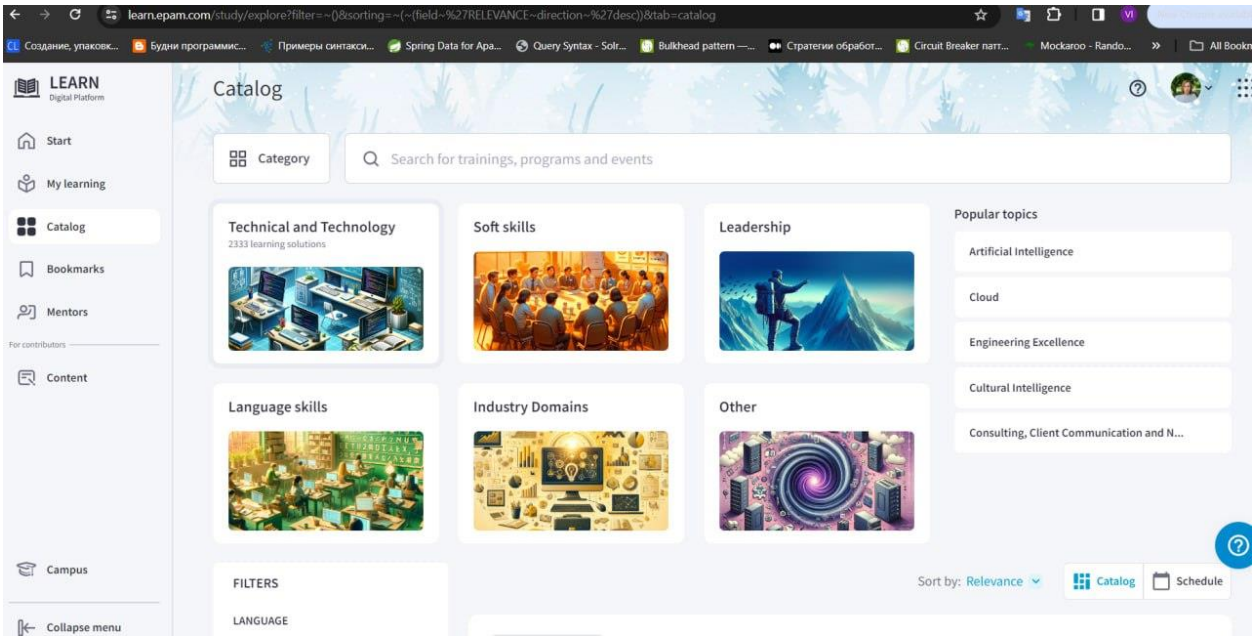

Рисунок 1.7 – Інтерфейс застосунку IT-компанії «Epam Systems» з рекомендаційною системою

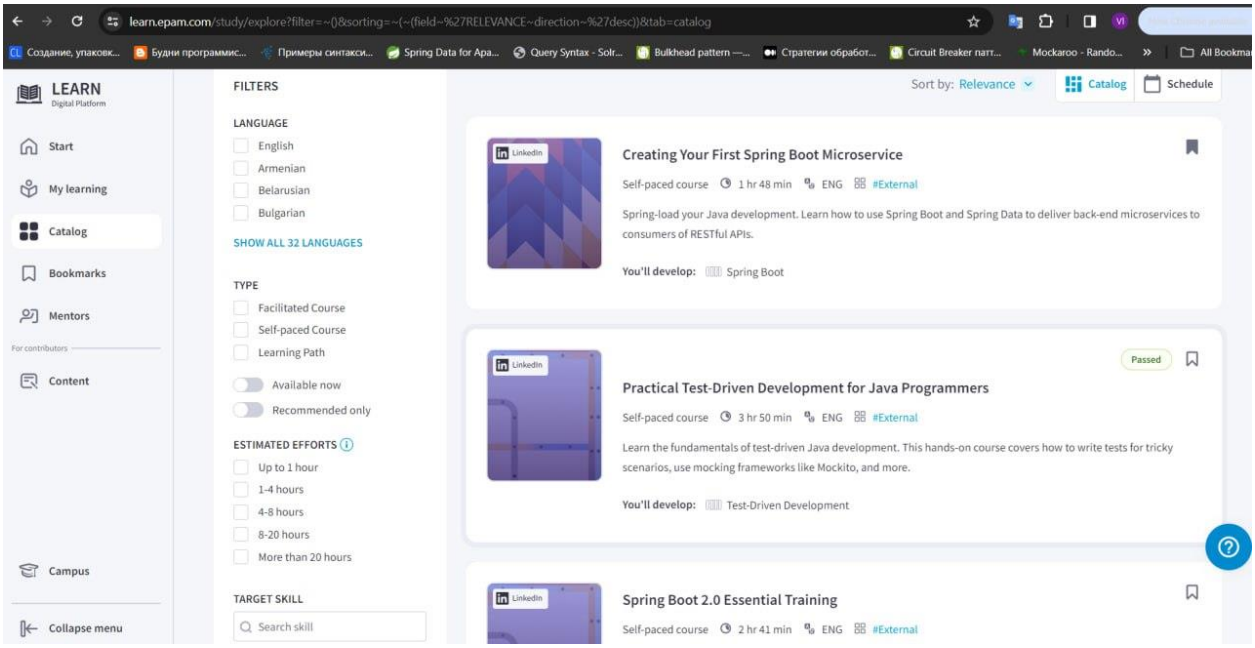

Рисунок 1.8 – Інтерфейс застосунку IT-компанії «Epam Systems» з рекомендаційною системою

На рисунку 1.9 зображено курси, які пропонує рекомендаційна система.

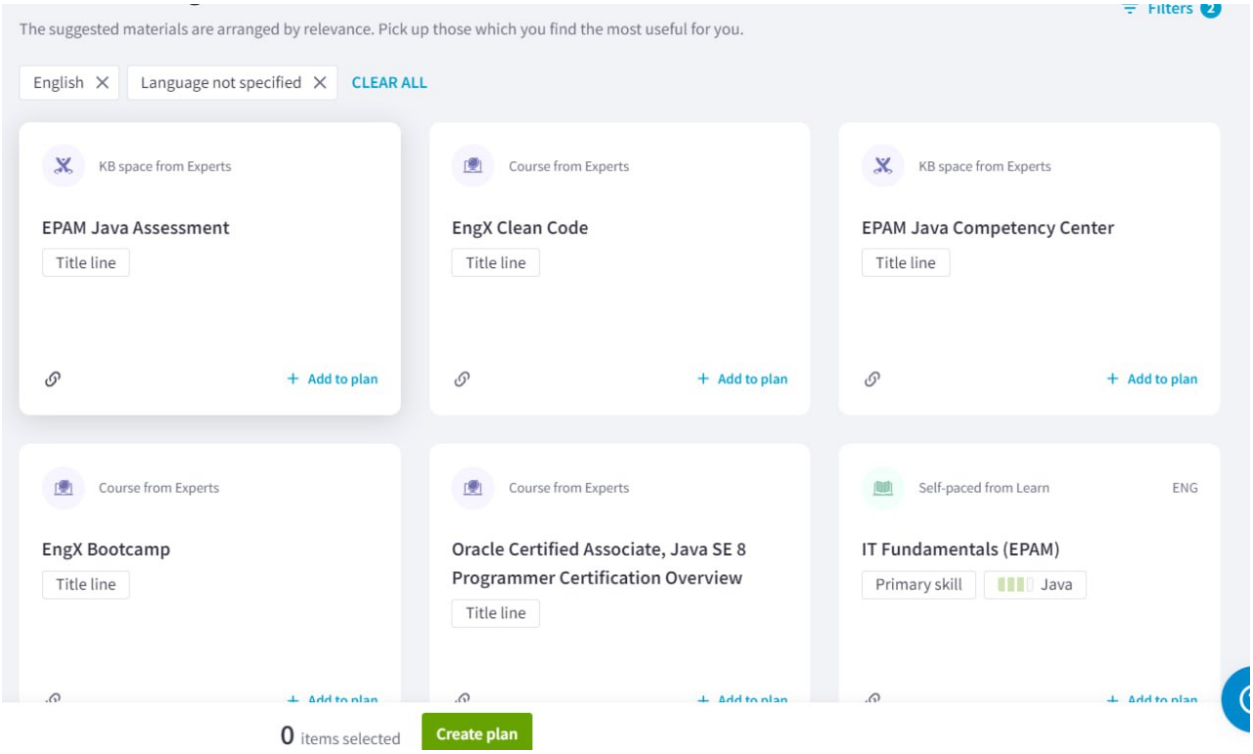

Рисунок 1.9 – Інтерфейс застосунку IT-компанії «Epam Systems» з рекомендаційною системою

Однією зі значущих проблем існуючої системи є використання для відбору курсів внутрішніх ресурсів компанії на підставі лише введених необхідних навичок, не враховуючи поточної ситуації на українському та світовому ринку праці, а також поточних вимог проєктів. Це призводить до того, що більш актуальні курси не вивчаються, а працівники не отримують необхідних навичок для ефективної роботи. Для з'ясування більш детальних аспектів проблеми, розглянемо алгоритм існуючої системи та окреслимо етапи, які потрібно покращити і чому це необхідно.

Рисунок 1.10 – Інтерфейс профілю працівника

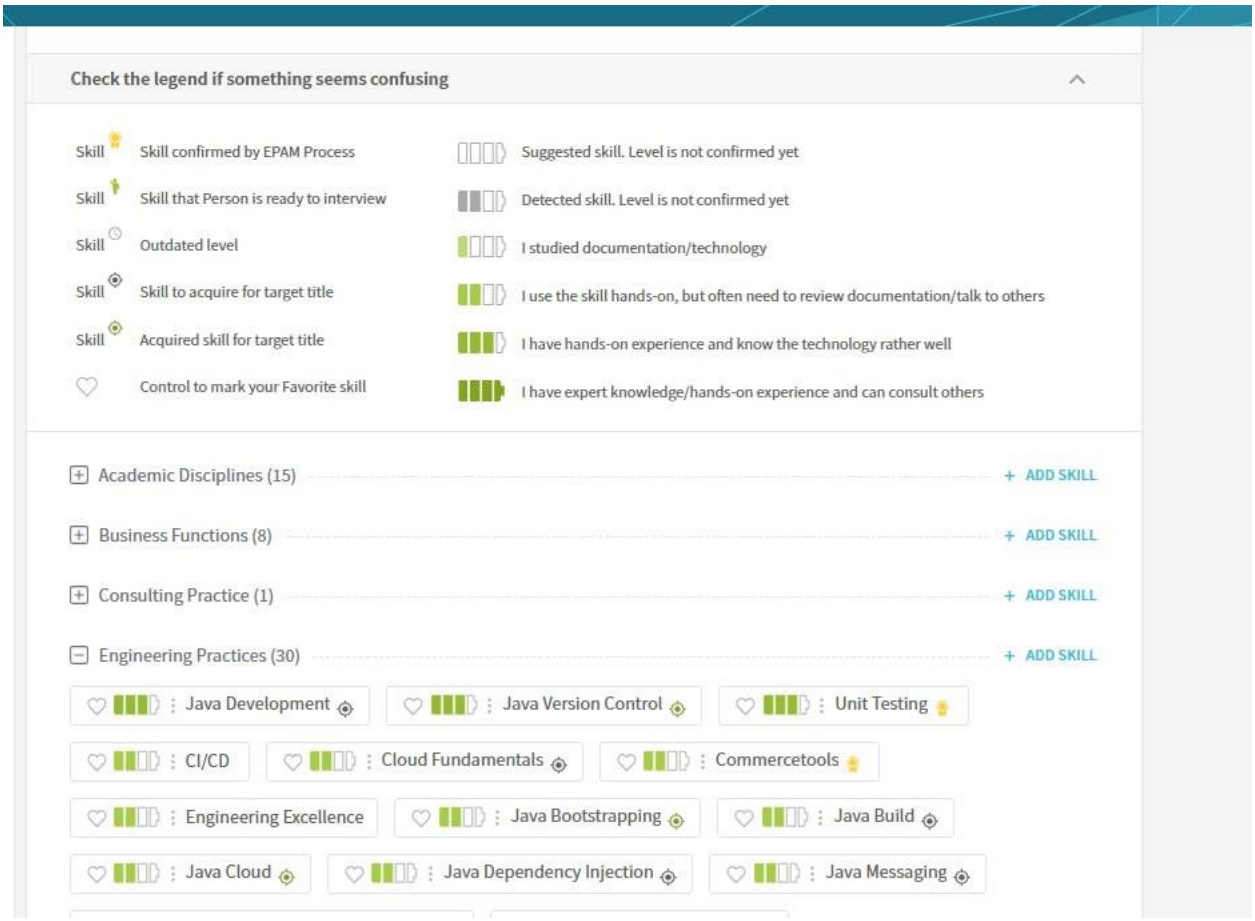

Рисунок 1.11 – Інтерфейс заповнених навичок співробітника

Для існуючої системи, користувач повинен бути співробітником IT-компанії та заповнити свій профіль та навички, указавши їх рівень, який може приймати значення від 1 до 4. Приклад профілю зображений на рисунку 1.10. На рисунку 1.11 зображено приклад заповнених навичок.

Після цього рекомендаційна система IT-компанії підбирає курси на основі введених даних. Однак серед рекомендованих можуть бути некоректні курси, що містять навички, які вже неактуальні для проєктів. Також проблемою є те, що деякі рекомендовані курси можуть мати обмежену кількість місць або вже набрали достатню кількість учасників, тому працівники, яким потрібен цей навчальний матеріал, можуть не мати можливості записатися на курс. Крім того, деякі рекомендовані курси можуть початися через деякий час, що робить їх неактуальними для співробітників.

1.7 Визначення сфери застосування інформаційної системи дистанційних курсів з програмування IT-компанії

Основним підходом до підбору курсів працівникам компанії повинно бути використання аналізу даних про навички робітника, вимоги до навичок для посади на проєкті, а також аналіз спрямованості курсів та встановлення відповідності поточних навичок працівника посадам, які він може обіймати та курсам, які необхідно пройти для того, щоб підвищити рівень знань для необхідної посади. Так, для покращення системи підбору проєктів та курсів працівнику, було вирішено розширити основний алгоритм, який буде функціонувати у декілька етапів та буде корисним та ефективним для IT-компанії для виконання описаних задач. Основною перевагою використання описаного алгоритму є те, що такий алгоритм комплексно враховує поточний рівень навичок працівника, поточні вимоги до вакансій на проєктах та курси, які необхідні для того, щоб підвищити рівень знань працівника для обіймання доступної вакансії. Урахування усіх перелічених факторів підбору курсів робить отриманий перелік рекомендацій у найбільшій мірі релевантним для працівників.

1.8 Постановка задачі

Нехай $R-$ резюме працівника, що містить основні дані, такі як сфера розробки, рівень та навички з їх рівнями. $U = \{u_1, u_2, \dots, u_n\}$ – множина наявних проєктів в системі, які можуть використовуватися для аналізу вимог, $V = \{v_1, v_2, \dots, v_m\}$ – множина наявних курсів в системі, які можуть бути запропоновані співробітнику. Кількість проєктів n та кількість курсів m в системі можуть бути достатньо великими (декілька тисяч або мільйонів).

Кожен проєкт u_i представлений у вигляді вектора текстових характеристик $u_i^t = \{u_1^t, u_2^t, \dots, u_l^t\}$.

Кожен курс v_j представлений у вигляді вектора текстових характеристик $v_j^t = \{v_1^t, v_2^t, \dots, v_p^t\}$.

Нехай h – функція, яка відображає корисність проєктів $u \in U$ та курсів $v \in V$ для працівника на основі його резюме R:

$$\mathrm{h}: \mathrm{R} \times \mathrm{U} \times \mathrm{V} \rightarrow \mathrm{E},$$

де E – повністю впорядкована множина (наприклад, цілі невід'ємні чи дійсні числа в певному діапазоні).

Тоді, для користувача R необхідно знайти такі курси $v \in V$, які дозволили би отримати максимальну корисність при їх вивченні для покращення резюме працівника R.

Математично задачу знаходження найбільш релевантних курсів можна описати у вигляді:

$$\forall\, v \in V, v'_u = arg\, max_{v \in V}\, h(R, u, v).$$

У простих системах рекомендацій, які використовують явний зворотний зв'язок, функція корисності h виражається як оцінка, яку користувач ставить певному продукту за фіксованою шкалою. Проте значення функції h в нашому випадку залежить від трьох параметрів та повинна враховувати предметну область та обмеження що випливають на її основі, наприклад вимоги проєктів.

Тому задачею дослідження є виявлення такої функції корисності, яка б виявилася ефективнішою за наявні аналоги та задовільнила б вимоги.

2 РОЗРОБКА МЕТОДА ПІДБОРУ ТА РОЗПОДІЛУ РЕКОМЕНДОВАНИХ КУРСІВ ПРАЦІВНИКАМ ІТ-КОМПАНІЇ

2.1 Розробка схеми алгоритму та вимог до його етапів

Розробку алгоритму необхідно почати зі структури. Для проектування, необхідно поставити вимоги до кожного етапу [9]. Спочатку виділимо етапи та схематично зобразимо їх. Результат зображений на рисунку 2.1.

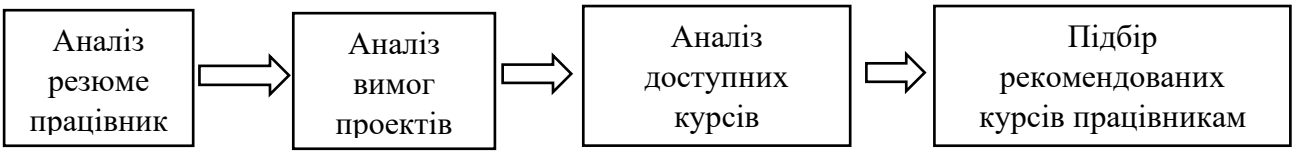

Рисунок 2.1 – Схема алгоритму

Отже, алгоритм складається з чотирьох етапів:

– аналіз резюме працівника. На даному етапі збираються дані працівника. Основними параметрами є його сфера діяльності, наприклад, веб-розробка або мобільна розробка, рівень працівника, що є одним з найважливіших критеріїв при підборі рекомендаційних курсів та навички та їх рівень (від 1 до 4). Існує 4 рівня працівника: Junior, Middle, Senior, Lead;

– аналіз вимог проєктів. На даному етапі збираються дані проектів, такі як сфера, рівень працівника, необхідні та бажані вимоги до працівника. Вимоги проєктів необхідні для визначення навичок необхідних для вивчення, наприклад підвищити рівень навичку «Java» або вивчення нового навичку «Spring Framework»;

– аналіз доступних курсів ІТ-компанії. На даному етапі аналізуються курси ІТ-компанії для визначення доступності та отримання даних для коректного розподілу курсів;

– підбір рекомендованих курсів працівникам ІТ-компанії. На даному етапі використовуючи дані з попередніх, підбирається курси працівнику за допомогою алгоритму розподілу.

Далі необхідно поставити цілі та вимоги до кожного етапу, на основі яких буде спроектована структура алгоритму. Структура повинна включати етапи, вимоги та потік даних.

Перший етап: Аналіз резюме працівника. Основна ціль – це збір даних про працівника для подальшого підбору рекомендаційних курсів.

Вимоги:

– на вхід подається резюме працівника;

– на виході є дані про сферу діяльності працівника, наприклад, Web development, Mobile Development;

– на виході є дані про рівень працівника (Junior, Middle, Senior, Lead);

– на виході є список основних навичок та їх рівень, наприклад Java – 4, Spring – 3, SpringBoot – 2, JSP- 3, Angular – 1;

– потік даних повинен перейти на наступний етап (сфера діяльності, рівень працівника, рівень навичок).

Другий етап: Аналіз вимог проєктів. Основна ціль – це збір вимог до працівника для визначення необхідних навичок для проєктів.

Вимоги:

– на вхід подається дані з першого етапу, а саме опрацьовані дані працівника;

– на вхід подаються дані проєктів;

– дані проєктів повинні включати:

а) одну чи декілька сфер розробки проекту. Наприклад, Web development, Web Design, Mobile Development;

б) вимоги до рівня працівника. Наприклад, Junior, Middle, Senior або Lead;

в) вимоги до необхідних навичок. Наприклад, Java – 3, Spring – 2, це означає, що для проєкта працівнику необхідно мати навичок «Java» на рівні 3;

г) вимоги до бажаних навичок. Наприклад, Angular – 2, це означає,

що для проєкту працівнику бажано мати навичок Angular на рівні 2.

– список відібраних проєктів для аналізу вимог повинен:

а) відноситися до тієї ж сфери розробку проєкту;

б) бути відповідним до рівня співробітника, або відрізнятися не більше ніж на 1. Якщо рівень працівника нижче необхідного на один, то проєкт має менший пріоритет в списку. Наприклад, якщо працівник має рівень «Junior», тоді всі проекти де необхідні працівники рівня «Senior» і вище прибираються зі списку, а де необхідні працівники рівня «Middle», мають менший пріоритет у списку. Якщо проєкт має вимогу до рівня «Junior», то їх пріоритет в списку найбільший;

в) загальне задоволення навичками працівника вимог проєкту хоча б 50%. Детальніше розрахунки загального задоволення буде далі, в розділі розробки другого етапу.

– на основі списку відібраних проєктів повинен на вихід бути поданий список навичок та їх оцінка, що служить пріоритетом для вивчення.

Третій етап: Аналіз доступних курсів IT-компанії. Основна ціль – це визначення доступних курсів для навчання працівників.

Вимоги:

– на вхід подаються дані з другого етапу, а саме список навичок та їх оцінка, що служить пріоритетом для навчання;

– на вхід подаються дані курсів з платформи навчання IT-компанії;

– на вихід подається список навиків із оцінками. До кожного навичка є список курсів для вивчення;

– кожен курс повинен містити наступні дані:

а) назва курсу;

б) час вивчення курсу;

в) максимальна кількість студентів, що може записатися на проходження курсу. В подальшому буде використовуватися для нормального розподілу курсів;

ґ) кількість студентів, які вже записані на курс;

г) кількість працівників кому був рекомендований курс. Ці дані необхідні, щоб нормально розподілити курси серед всіх працівників;

д) дата початку вивчення;

є) сфери розробки;

е) навички. Обов’язково назва, і хоча б один із наступних атрибутів: необхідний рівень навичка для вивчення курсу або рівень до якого збільшиться навичок після вивчення курсу.

– навички до яких не має доступних курсів, повинні бути виключені зі списку;

– курси повинні мати ідентичну до працівника сферу розробки.

Четвертий етап: Підбір рекомендованих курсів працівникам IT-компанії. Основна ціль – це підбір курсів для навчання працівників на основі їхніх навичок, вимог проєктів та доступних курсів.

Вимоги:

– використання даних з попередніх етапів для підбору курсів, а саме дані працівника, список навичок та курсів для вивчення та дані курсів;

– вихідні дані складаються зі списку курсів рекомендованих для вивчення працівнику, та запропонований пріоритет вивчення;

– алгоритм повинен нормально розподілити рекомендації курсів, та забезпечити такі рекомендації, щоб працівники рівномірно записувалися на курси. Тобто відсоток записаних від максимально можливої кількості повинен бути максимально однаковим на курсах.

2.2 Розробка структури алгоритму

Структура алгоритму [10] починається зі старту та переходу на перший етап, аналіз резюме працівника. Для цього етапу необхідні дані зі сховища даних «Employee Data Storage», де зберігаються дані резюме працівників. На виході

отримуємо блок даних «Employee Resume Data», що складається з «Area» (сфера розробки), «Level» (рівень співробітника) та «Skills and their level» (список навичок та їх рівні, від 1 до 4). Ці дані передаються на другий етап, «Project Requirements Analysis» для аналізу даних проєктів, таких як необхідні та бажані вимоги до працівника. Також для цього етапу необхідні дані зі сховища даних «Project Data Storage», де зберігаються вимоги проєктів, які необхідні для визначення навичок необхідних для вивчення. Після аналізу, отримаємо список навичок з їх пріоритетами.

Для третього етапу, аналізу доступності курсів в репозиторію IT-компанії, необхідні дані другого етапу «List of skills with priority». Також на вхід подаються дані зі сховища даних «Courses Data Storage», що складаються із наступних складових: назва курсу, час вивчення курсу, максимальна кількість студентів, що може записатися на проходження курсу, кількість студентів, які вже записані на курс, кількість працівників кому був рекомендований курс, дата початку вивчення, сфери розробки та навички. В результаті на виході отримуємо дані «List of skills with priority and list of course for each skill», що складаються із списків навичок із пріоритетами та курсів.

Структура алгоритму зображена на рисунку 2.2.

На останньому етапі виконується підбір рекомендованих курсів працівникам IT-компанії. Для цього на вхід подається блок даних «List of skills with priority and list of course for each skill» та «Employee Resume Data». В результаті отримуємо блок даних «Recommended Courses», що складається зі списку рекомендованих курсів та їх пріоритет вивчення.

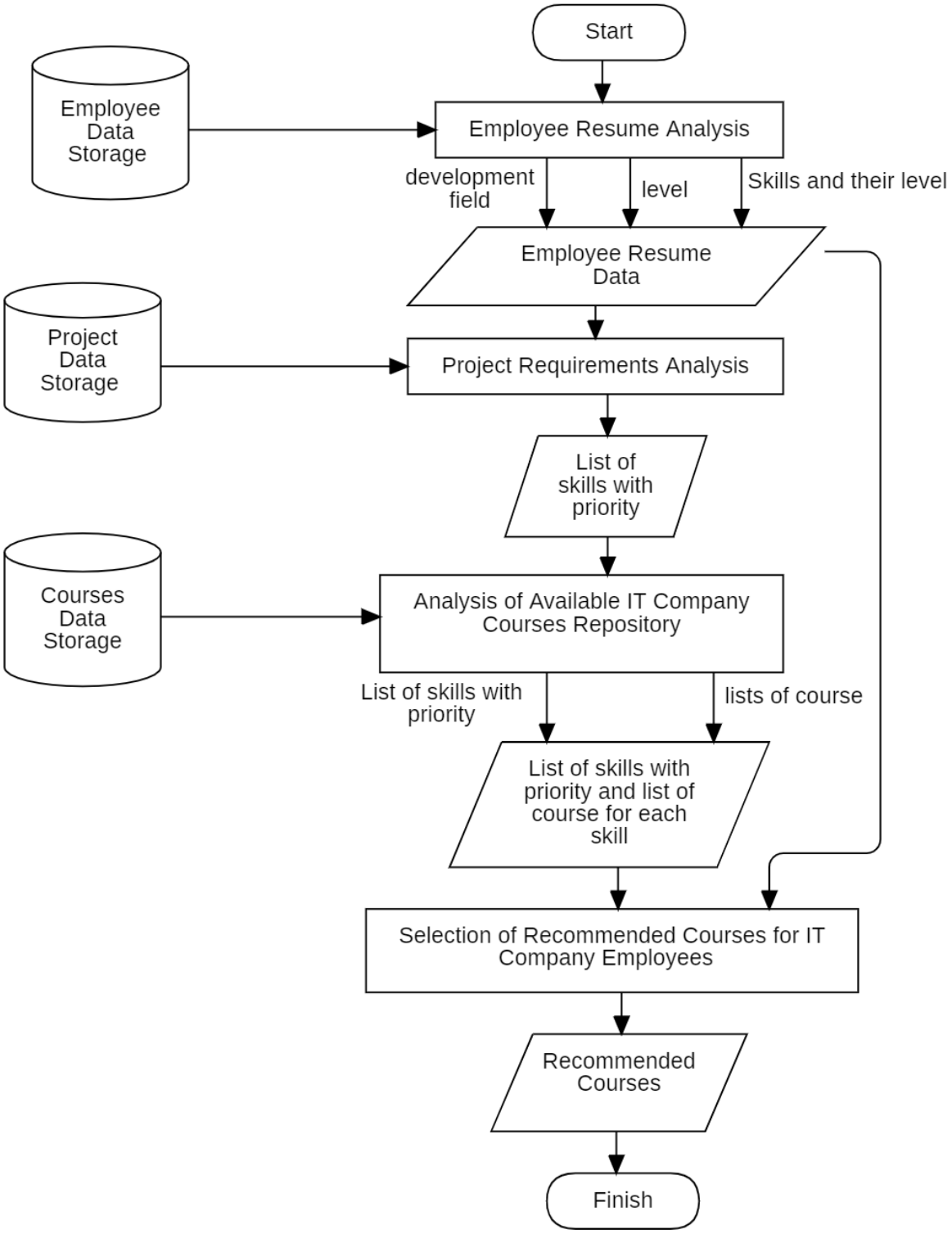

Рисунок 2.2 – Структура алгоритму підбору та розподілу рекомендованих курсів працівникам IT-компанії

2.3 Розробка першого етапу: аналіз резюме працівника

Використовуючи ціль та вимоги до першого етапу з розділу 2.1, розробимо алгоритм аналізу резюме працівника. Після аналізу структури алгоритму (рис. 2.2), спочатку необхідно мати сховище даних «Employee Resume Data». З нього необхідно відокремити дані «Area», «Level» та список «Skill», «Skill level».

Ці дані необхідно передати на наступний етап. Приклад вихідних даних показаний у лістингу 2.1.

Лістинг 2.1 – Приклад вихідних даних етапу аналізу резюме працівника.

```
{
  "area": "Web development",
  "level": "Middle",
  "skills": [
    {
      "name":"Java",
      "level": 4
    },
    {
      "name":"Spring Framework",
      "level": 3
    },
    {
      "name":"SpringBoot",
      "level": 2
    }
  ]
}
```

2.4 Розробка другого етапу: аналіз вимог проєктів

Використовуючи ціль та вимоги до другого етапу з розділу 2.1, розробимо алгоритм аналізу вимог проєктів. На вхід подаються дані у вигляді лістингу 2.1. Також повинно бути сховище даних «Project Data Storage», з яких необхідно буде отримати список сфер розробки проєкту, вимоги до рівня

працівника, вимоги до необхідних навичок, вимоги до бажаних навичок. Приклад даних проєкту поданий в лістингу 2.2.

Лістинг 2.2 – Приклад даних проєкту.

```
{
  "area": [
    "Web development",
    "Mobile development"
  ],
  "requiredLevel": "Middle",
  "requiredSkills": [
    {
      "name": " Java",
      "level": 3
    },
    {
      "name": "Spring Framework",
      "level": 3
    }
  ],
  "desiredSkills": [
    {
      "name": "SpringBoot",
      "level": 3
    },
    {
      "name": "Angular",
      "level": 2
    }
  ]
}
```

Після отримання списку проєктів з поданими даними, необхідно його відфільтрувати, залишивши лише ті проєкти, які мають збіг по одній сфері розробки з працівником. Далі необхідно відфільтрувати за необхідним рівнем працівника для проєкту. Якщо різниця в рівнях більше одного, то ці проєкти необхідно прибрати. Наприклад, необхідний Senior, а наявний Junior, то цей проєкт необхідно відфільтрувати. Також працює і навпаки, якщо наявний Senior, то проєкти з необхідним Junior, необхідно виключити зі списку для отримання більш релевантної інформації.

Для підвищення ефективності алгоритму, найкращим варіантом буде фільтрація по сфері розробки та рівнем під час запиту до сховища даних. Ефективність зумовлена наступними причинами:

– отримуємо лише ті дані, які відповідають вимогам, уникнувши

отримання непотрібної інформації. Це дозволяє мінімізувати обсяг даних, які потрібно обробляти і передавати через мережу;

– зменшуємо обчислювальні витрати на обробку даних, оскільки фільтрація відбувається безпосередньо в сховищі даних, що зменшує потребу в обробці даних на стороні клієнта;

– передача менших кількість полей. А саме буде прибрано поле «area», що може приймати декілька значень.

Отримавши дані необхідно їх опрацювати. Перша вимога полягає в тому, що загальне задоволення навичками працівника вимог проєкту буде хоча б 50%. Ця умова необхідна для того, щоб відфільтрувати проєкти, де більшість технологій не буде знайома співробітнику. Через велику кількість технологій, можуть бути різні ситуації, через які проєкти, що не є потрібними для працівника будуть враховуватися в алгоритмі.

Без фільтрації по навикам, в списку будуть всі проєкти, що були знайдені, навіть якщо проєкт не має взагалі співпадінням по навикам співробітника. Одним із вирішенням цієї проблеми є додавання фільтрації по навикам, тобто вибрати головний навик і по ньому вибирати проєкти. Але з’являється нова проблема пов’язана з цим рішенням. По-перше, необхідно, щоб головний навик був необхідним і основним на проєкті. По-друге, може статися ситуація що в проєкті лише співпадіння по основному навику. Наведемо приклад, є проєкт, де в списку необхідних навиків є мова програмування «Java», але інший стек технологій відрізняється від того, що вивчав працівник. Через це, вимоги проєкту не є релевантними. Тому необхідно, щоб співпадіння по навичкам було хоча б 50%. Якщо співпадіння 100%, то проєкт має сенс прибрати, через те що всі навички задоволені, тобто ні чого розвивати.

Далі з’являється наступна проблема, різносторонність працівника. Співробітник міг вивчати різні теми, наприклад, Frontend, «Angular». Але основний напрямок Backend, мову програмування «Java». Потенційно існує проєкт, який в ключових навиках має «Java», а в бажаних «Angular», при тому бажаних зазвичай більше. Через що працівнику може вибратися такий проєкт для

подальшого аналізу, і в решті рекомендовані навички будуть не правильними. Одним із вирішенням цієї проблеми це аналізувати лише ключові навички. Але і це рішення не є доскональним, через те що існують проєкти, вимоги ключових навичок яких на 100% задовольняє працівник, але деякі бажані навички необхідні для вивчення. Тому бажані навички також необхідно враховувати, що приводить до наступного рішення: врахування всіх навичок але з різними коефіцієнтами, який залежить від типу навички (необхідна чи бажана). Коефіцієнти для типу навички необхідно розрахувати для отримання найбільш релевантних даних. Необхідно розрахувати середнє задоволення необхідними та бажаними навичками, і кожне перемножити на свій коефіцієнт.

Тоді формула для розрахунку загального задоволення вимог проекту:

$$X = \frac{(C_{11} + C_{12} + \cdots + C_{1m})}{m} * K_1 + \frac{(C_{21} + C_{22} + \cdots + C_{2n})}{n} * K_2 , \qquad (2.1)$$

де X – загальне задоволення навичками працівника вимог проекту;

m – кількість необхідних навичок;

n –кількість бажаних навичок;

K_1 – коефіцієнт для необхідних навичок;

K_2 - коефіцієнт для бажаних навичок;

C_{1i} – i-те задоволення необхідного навичка, яке може бути або 50%, якщо рівень навичка працівника є меншим на одиницю, чим рівень вимоги, 100% якщо, відповідає або краще, та 0%, якщо меншим заданого на 2+ (тобто, якщо рівень необхідного навичка 3, а працівник має лише 1, тоді рахується 0% задоволення);

C_{2i} – i-те задоволення бажаного навичка.

Розрахуємо коефіцієнти для типу навички. Для цього необхідно спочатку визначити обмеження. Для правильного розрахунку, візьмемо формулу 2.1 та припустимо ситуацію де всі навички задоволені. Загальне задоволення в такій ситуації дорівнює 100%. Отримаємо:

$$X = \frac{100 * m}{m} * K_1 + \frac{100 * n}{n} * K_2 = 100,$$

де m – кількість необхідних навичок;

n – кількість бажаних навичок.

Скоротимо та отримаємо:

$$K_1 + K_2 = 1.$$

Отже, основна умова, що сума коефіцієнтів повинна бути 1. Враховуючи важливості, найкращим рішення буде виставити коефіцієнти наступним чином. K1=0.7, K2=0.3.

Розглянемо наступний приклад. Існує проєкт, з наступними вимогами до навичок.

Необхідні:

– Java – 3 рівень;

– Spring – 2 рівень;

– SQL – 3 рівень.

Бажано мати:

– Hibernate – 2 рівень;

– Rest Api – 2 рівень.

Резюме навичок працівника виглядає наступним чином:

– Java – 4 рівень;

– Spring – 2 рівень;

– SQL – 1 рівень;

– Hibernate – 1 рівень.

Далі розрахуємо задоволення кожного навичка. Для необхідних, отримуємо $C_{11} = 100$, тому що рівень співробітника дорівнює або вище за

необхідний, $C_{12} = 100$, аналогічно, $C_{13} = 0$, тому що рівень працівника менше аж на 2 рівня (1 рівень при необхідному 3). Для бажаних, $C_{21} = 50$, тому що рівень співробітника нижче на одиницю за необхідний, $C_{22} = 0$, тому що працівник не має даного навичка. Кількість необхідних навичок дорівнює 3, а бажаних 2.

Отримаємо наступне рівняння:

$$X = \frac{(100 + 100 + 0)}{3} * 0.7 + \frac{(50 + 0)}{2} * 0.3 =$$
$$= 66{,}667 * 0.7 + \ 25 * 0.3 = 46{,}67 + 7.5;$$

$$\mathrm{X} = 54{,}17\%.$$

У даному прикладі, загальне задоволення навичками дорівнює 54,17 %.

Друга вимога полягає в тому, що якщо рівні необхідного на проєкті та наявного у працівника однакові, то пріоритет проєкту більше ніж де рівень різний. Наприклад, якщо необхідний рівень «Middle», і такий же у співробітника, то пріоритет проєкту не змінюється. Якщо необхідний рівень «Middle», і у працівника рівень або «Junior» або «Senior», то пріоритет проєкта зменшується, множиться на коефіцієнт, що менше 1.

Після аналізу, коефіцієнт був установлений на рівні 0.75, щоб мати менший вплив, так як рівень проєкту відрізняється від працівника. Але притому, коефіцієнт повинен відображати важливість навичок в порівняні з рівнем проєкту.

Після фільтрації проєктів, залишається лише релевантний список, що відповідає вимогам. Далі необхідно зібрати відібрати список навичок та поставити їм оцінки, що служить пріоритетом для вивчення. Для цього створюємо словник, де ключ – назва навичка, а значення є оцінкою. Термін «Словник» або «Мапа» у програмуванні, особливо в контексті об'єктно-орієнтованих мов програмування, використовується для опису структури даних,

яка забезпечує відображення між ключами і значеннями. Словник є колекцією пар ключ-значення, де кожен ключ унікальний і пов'язаний з одним значенням. Це дозволяє швидко знаходити значення, використовуючи відповідний ключ [11].

Далі проходимось по списку проєктів, та переглядаємо навички. Якщо рівень навичка указаному в проєкті дорівнює або менший ніж у працівника, то пропускаємо. Якщо більший, то перевіряємо чи є в словнику навичок, якщо да, то необхідно додати до оцінки. Значення, що необхідно додати розраховується наступним чином:

$$A = \mathrm{P} * \mathrm{K}\ , \tag{2.2}$$

де P – коефіцієнт проєкту на основі вимог до рівня працівника, що був розглянутий вище. Якщо рівні необхідного на проєкті та наявного у працівника однакові, то P дорівнює 1. Інакше, якщо рівень різний, то P дорівнює 0.75;

K – коефіцієнт типу навичка. Якщо навичок є необхідним, то K дорівнює 1, якщо бажаним то 0.5.

Далі проходимось по словнику, та кожну оцінку ділимо на кількість проєктів і множимо на 100. Також округлюється до цілого числа. Отже оцінка навичка може бути від 0 до 100.

На виході отримуємо словник зі навичками та оцінкою (лістинг 2.3).

Лістинг 2.3 – Приклад вихідних даних словника проєкту.

```
[
  {
    "name": " Java",
    "priority": 56
  },
  {
    "name": "Spring Framework",
    "priority": 73
  },
  {
```

```
    "name": "SpringBoot",
    "priority": 38
  },
  {
    "name": "Angular",
    "priority": 52
  },
  {
    "name": "Rest API",
    "priority": 70
  }
]
```

2.5 Розробка третього етапу: аналіз доступних курсів IT-компанії

Використовуючи ціль та вимоги до третього етапу з розділу 2.1, розробимо алгоритм аналізу доступних курсів IT-компанії. На вхід подаються дані у вигляді лістингу 2.3. Також повинно бути сховище даних «Courses Data Storage», з яких необхідно буде отримати основні дані курсів, а саме назва, час вивчення курсу, максимальна кількість студентів, що може записатися на проходження курсу, кількість студентів, які вже записані на курс, кількість працівників кому був рекомендований курс, дата початку вивчення, сфери розробки та навички. Приклад даних курсу поданий в лістингу 2.4.

Лістинг 2.4 – Приклад вхідних даних курсу.

```
{
  "tittle": "Java Course for Web development",
  "area": [
    "Web development",
    "Mobile development"
  ],
  "skills": [
    {
      "name": " Java",
      "requiredLevel": 2,
      "enhancementLevel": 3
    },
    {
      "name": "REST API",
      "enhancementLevel": 1
    },
    {
```

```
      "name": "Unit Testing",
      "requiredLevel": 1
    }
  ],
  "duration": 80,
  "capacity": 100,
  "enrolled": 30,
  "recommended": 10,
  "start": "10.04.2024"
}
```

Важливими є дані навиків в курсах. Обов’язково повинна бути назва, і хоча б один із наступних атрибутів:

– "requiredLevel" – це необхідний рівень навичка для вивчення курсу, необхідний для фільтрації курсів;

– "enhancementLevel" – це рівень до якого збільшиться навичок після вивчення курсу.

Після отримання списку курсів з поданими даними, необхідно відфільтрувати, залишивши лише ті, які мають збіг по одній сфері розробки з працівником. Також необхідно відфільтрувати за датою не раніше сьогодні, та щоб кількість записавшись на курс студентів було менше, ніж максимальна можлива кількість.

Для підвищення ефективності алгоритму, найкращим варіантом буде фільтрація по сфері розробки, датою та умовою, що кількість записавшись на курс студентів було менше, ніж максимальна можлива кількість під час запиту до сховища даних. Ефективність зумовлена наступними причинами:

– отримуємо лише ті дані, які відповідають вимогам, уникнувши отримання непотрібної інформації. Це дозволяє мінімізувати обсяг даних, які потрібно обробляти і передавати через мережу;

– зменшуємо обчислювальні витрати на обробку даних, оскільки фільтрація відбувається безпосередньо в сховищі даних, що зменшує потребу в обробці даних на стороні клієнта;

– передача менших кількість полей. А саме буде прибрано поле «area», що може приймати декілька значень.

Отримавши дані необхідно їх опрацювати для того, щоб врахувати доступність курсів за обраними навичками. Це є важливим фактором, щоб уникнути ситуацій, коли всіх працівників буде записано на один курс, який має обмеження на кількість учасників, або курс за обраним навиком не доступний.

Спочатку необхідно відсортувати словник за спаданням пріоритету для навичка. Далі для кожного навичка із словника знайти список курсів, що підходить для вивчення. Також потрібно врахувати вимоги.

Перша вимога, навички для яких немає курсів необхідно виключити для видалення непотрібних даних та пришвидшення алгоритму.

Друга вимога, необхідно поставити пріоритет курсам в залежності від пріоритету навиків, та даних курсу. Важливими критеріями є навички, тривалість курсу, максимальна кількість студентів, кількість що уже записались і скільком рекомендований вже курс.

Третя вимога, кількість курсів необхідно обмежити. В каталозі може бути величезна кількість курсів, і в деяких ситуація існує вірогідність знаходження для вивчення навичок непотрібних курсів. Зрозуміло, що якщо буде знайдено 100 000 курсів, то необхідно близько 100 лише, інші лише задають додаткове навантаження системі. Відбір необхідно зробити за допомогою пріоритетів курсів.

Враховуючи вимоги, пройдемось по словнику, що впорядкований по спаданню пріоритету навичка. Для кожного навичку знаходимо курси зі списку, що підходять головній умові:

– назва навичка співпадає;

– якщо навичок курсу має атрибут «requiredLevel», то потрібно перевірити чи відповідає рівень навичка працівника необхідному. Якщо цей атрибут відсутній, то це означає, що немає обмежень в курсі по цьому навичку, і для його вивчення непотрібні попередні знання;

– якщо навичок курсу має атрибут «enhancementLevel», то потрібно перевірити, чи більший він за поточний рівень працівника. Необхідно, щоб хоча б один навичок покращувався за курс, інакше пропускаємо.

Якщо курс відповідає головній умові, отже додаємо його до списку для вивчення вибраної навички. Далі цьому курсу необхідно розрахувати пріоритет. На цьому етапу з’являється складність, через те що, курс може покращувати не лише один навик. Тобто для розрахунку пріоритету курсу, необхідно врахувати декілька критеріїв:

– розрахунок всіх оцінок для навичок, що підвищує курс. Необхідно врахувати пріоритет вивчення навичка, рівень з якого на який підвищується;

– кількість вільних місць на курсі;

– тривалість курсу.

Але якщо курс підвищує декілька навичок, то для оптимізації буде краще опрацювати не словник, а список курсів. Створюємо словник, де ключ це навичок, а значення це список курсів з їх пріоритетами. Опрацюємо список курсів . Для кожного курсу розраховуємо його пріоритет за формулою 2.3.

$$X = \sum_{i=1}^{n} K_i * (e_i - c_i)\,, \tag{2.3}$$

де n – це кількість навичок що підвищуються, тобто мають атрибут «enhancementLevel»;

K_i – оцінка навичка, що береться зі словника отриманого з другого етапу;

e_i – рівень до якого підвищується навичок;

c_i – наявний рівень навичка у співробітника (якщо немає, то 0).

На виході отримаємо список навичок, та курси для їх вивчення з пріоритетом.

2.6 Розробка четвертого етапу: підбір рекомендованих курсів працівникам IT-компанії

Використовуючи ціль та вимоги до четвертого етапу з розділу 2.1, розробимо алгоритм підбору рекомендованих курсів працівникам IT-компанії.

На вхід подається список навичок, та курси для їх вивчення з пріоритетом та дані «Employee Resume Data».

Отримавши дані з попередніх, останній етап використовує їх для підбору рекомендацій, а саме для правильного розподілу курсів. Для цього необхідно виставити пріоритети працівникам на конкретний курс для максимізації навчання всього персоналу. Критеріями для цього будуть навички співробітника, оцінки навичок для навчання, час курсу, кількість рекомендованих працівників на цей курс та максимальна кількість студентів на курсі. Наприклад, можливі такі ситуації: для працівника важливими по оцінці 3 курси, але на один із них рекомендацій більше, ніж максимальна кількість, на інший – великий час навчання та невеликий збіг із даними навичками співробітника, останній краще всього буде підходити через менший час навчання та більше корисності для працівника.

Для підбору рекомендацій використаємо алгоритм рекомендаційної системи заснований на вмісті і аналіз задач багатокритеріальної оптимізації, а саме метод ідеальної точки.

Алгоритм, що включає метод ідеальної точки для аналізу задач багатокритеріальної оптимізації в методі заснованому на вмісті має наступні кроки:

– визначення критеріїв оптимізації, за якими будуть оцінюватися курси. Наприклад, пріоритет курсу, тривалість курсу, складність матеріалу, актуальність, якість навчання тощо;

– нормалізація значень кожного критерію, щоб вони були на однаковій шкалі;

– використовуючи метод ідеальної точки, необхідно знайти точку, яка максимізують кожен критерій окремо. Ця точка буде називатися ідеальною;

– для кожного курсу необхідно обчислити відстань до ідеальної точки за допомогою метрики, такої як евклідова відстань (1.2);

– ранжування курсів в порядку зростання відстані до ідеальної точки і

відбір курсів для рекомендації, які мають найменшу відстань.

Додавання алгоритму ідеальної точки допоможе покращити точність рекомендацій, враховуючи багато критеріїв оптимізації одночасно та забезпечуючи більш персоналізовані рекомендації для працівників IT-компанії.

Визначимо необхідні поля для пошуку критеріїв оптимізації:

– максимальна кількість студентів на курсі;

– кількість студентів, що вже записалася;

– кількість працівників, яким рекомендувався цей курс;

– кількість годин на вивчення курсу;

– пріоритет вивчення курсу на основі навиків.

Спочатку перетворимо поля в критерії для подальших розрахунків. Максимальна кількість студентів на курсі не має значення окремо від кількості студентів, що вже записались. Але важливим є критерій відношення кількості студентів, що вже записалися до максимальної кількості студентів на даному курсі. Він показує відсоток вільних місць, що є важливим для нормального розподілу курсів. Тобто критерієм виберемо «кількість вільних місць».

Далі розглянемо поле «кількість працівників, яким рекомендувався цей курс». Він не має значення окремо від кількості студентів, що вже записались. Але важливим є критерій відношення кількості вільних місць до кількості рекомендацій. Чим більше рекомендацій відносно місць, тим менше повинен бути пріоритет рекомендації. Тобто критерієм виберемо «відношення рекомендацій до вільних місць».

Поля «кількість годин на вивчення курсу» та «пріоритет вивчення курсу на основі навиків» можуть бути критерієм оптимізації.

Далі нормалізуємо ці критерії за допомогою мінімаксного методу нормалізації, що був розглянутий в розділі 1.

Розглянемо нормалізацію кожного критерію:

– «кількість вільних місць». Мінімальне значення 0, максимальне

значення максимальна кількість студентів на цьому курсі. Для нормалізації необхідно розділити кількість студентів, що вже записалися, на максимально можливу кількість студентів на курсі;

– «відношення рекомендацій до вільних місць». Мінімальне значення 0, кількість рекомендацій цього курсу. Необхідно розділити кількість працівників, яким рекомендувався цей курс на кількість вільних місць. Ми отримаємо значення від 0 до кількості рекомендацій. Першою проблемою є те, що при збільшенні кількості рекомендацій або зменшенні кількості вільних місць, наш критерій зростає, але для подальших розрахунків нам необхідна інверсія, щоб чим вище критерій тим більше пріоритет. Ця логіка зумовлена тим, що якщо 10 вільних місць і 1000 рекомендацій, то звісно цей курс необхідно менше рекомендувати працівникам, інакше рекомендації будуть не актуальними. Також необхідно масштабувати значення відрізу від 0 до 1. Для цього була вибрана експоненційна функція зворотної експоненти [12] що має наступну формулу:

$$f(x) = e^{\frac{-x}{6}}.$$

Вибір експонентної функції, зазвичай базованої на числі e, є досить поширеним у математичних та наукових областях з декількох причин:

– математична простота: Функції з базою e часто мають простіші похідні та інтеграли, що полегшує їх аналіз та обчислення;

– широке застосування: Число e є основою для зростання, розпаду, витрати популяції, і багато інших природних та соціальних процесів, тому використання експонентної функції дозволяє зручно моделювати ці явища;

– універсальність: Функція e^x, або експонента, виявляється у різних аспектах математики, фізики, економіки та інших наук, що робить її корисною для широкого спектру задач.

Така формула була вибрана через здатність швидко приближати великі значення до 1, а також зменшити падіння значення функції при x менше 1. Для

підбору коефіцієнтів, використовувався графік та три функції (рис.2.3).

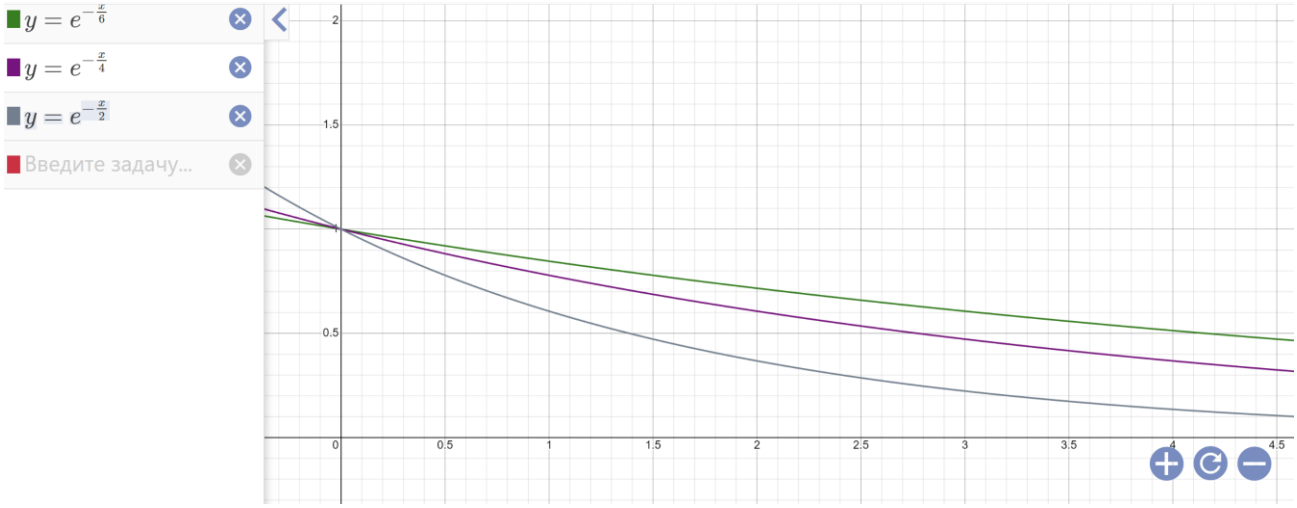

Рисунок 2.3 – Графік експоненційних функцій

Для вирішення вибору коефіцієнта (2, 4, 6), необхідно зрозуміти які значення приймає функція. Візьмемо для прикладу x=4, тобто на одне вільне місце курсу було зроблено 4 рекомендації. Наприклад 10 вільних місць та 40 рекомендацій.

Глянемо на значення функцій при x=4 та проаналізуємо.

$f(4) = e^{-2} = 0.1353$.

$f(4) = e^{-1} = 0.3679$

$f(4) = e^{-0.66} = 0.5151$

Ми отримали значення пріоритету рекомендації курсу на основі кількості рекомендацій відносно вільних місць. Враховуючи, що в середньому людина з вірогідністю 10% запишеться на курс, виходить, вільних місць залишиться 60% від наявних на даний момент. Найбільш підходить значення коефіцієнту 6.

– «кількість годин на вивчення курсу». Мінімальне значення 0, максимальне значення максимальна кількість годин в курсах. Також чим вище кількість годин, тим гірше значення критерію. Отже необхідно відняти від максимально можливої кількість годин на вивчення курсу та розділити на максимально можливу кількість годин на вивчення курсу, тобто $\frac{max-current}{max}$;

– «пріоритет вивчення курсу на основі навиків». Мінімальне значення 0.

Необхідно розділити пріоритет поточного курсу на максимальний пріоритет курсу в списку.

Ідеальною точкою виберемо ту, якій відповідає максимальне значення функції локальної корисності за кожним з критеріїв. Тобто 1 за всіма критеріями. Слід оцінити який з критеріїв є найближчим до ідеального. Спочатку установимо та застосуємо ваги до нормалізованих значень кожного критерію для кожного курсу:

– кількість годин на вивчення курсу – 10% (коефіцієнт 0.1);

– кількість вільних місць – 20% (коефіцієнт 0.2);

– відношення рекомендацій до вільних місць –10% (коефіцієнт 0.1);

– пріоритет на основі навиків – 60% (коефіцієнт 0.6).

Для оцінки будемо використовувати квадратичну функцію витрат. Оптимальне рішення x^0 знаходиться як $\underset{x \in X}{Argmin}\rho(y, \hat{y})$.

$$\underset{x \in X}{Argmin}\left(\sum_{i=1}^{4} \mathrm{w}_i (X'_i - X_i) \right), \tag{2.3}$$

де w_i – вага критерію;

X'_i – максимальне значення, тобто 1;

X_i – значення критерію.

Виконавши розрахунки для кожного курсу, отримаємо значення відстані до ідеальної точки. Далі йде ранжування курсів в порядку зростання відстані до ідеальної точки і відбір курсів для рекомендації, які мають найменшу відстань.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

3.1 Підготовка тестових наборів даних

Для початку експериментального дослідження необхідно підготувати тестові набори даних [13]. Потрібні наступні дані:

– резюме працівника. Основними параметрами є сфера діяльності, наприклад, веб-розробка або мобільна розробка, рівень працівника та навички, що включає назву та їх рівень (від 1 до 4). Існує 4 рівня працівника: Junior, Middle, Senior, Lead. Приклад даних подано у лістингу 2.1;

– список проєктів. Основними параметрами є сфера, рівень працівника, необхідні та бажані вимоги до працівника. Кожний навичок має два атрибута, назву та рівень. Приклад даних одного проєкту подано у лістингу 2.2;

– список курсів. Основними параметрами є назв, сфери розробки, час вивчення у цілих годинах, максимальна кількість студентів, що може записатися, кількість студентів, які вже записані, кількість працівників кому був рекомендований курс, дата початку вивчення, навички. Кожний навичок має обов’язково назву, і хоча б один із наступних атрибутів: необхідний рівень навичка для вивчення курсу або рівень до якого збільшиться навичок після вивчення курсу. Приклад даних одного курсу подано у лістингу 2.4.

Для отримання даних напишемо код, який записує необхідні дані в файли в форматі Json. Код констант поданий у лістингу 3.1. Код для генерації резюме працівника поданий у лістингу 3.2. Результат генерації резюме зображено на рисунку 3.1.

Лістинг 3.1 – Код констант для подальших генерацій даних

```
package com.hnure.interactive.algorithm.random;

import java.util.List;

public class RandomConstants {
    public static final List<String> AREAS = List.of(
            "Web development",
            "Mobile development",
```

```
            "Desktop application development",
            "Cloud computing",
            "Artificial intelligence",
            "Machine learning",
            "Data science",
            "Cybersecurity",
            "Game development",
            "UI/UX design",
            "Embedded systems",
            "DevOps",
            "Blockchain development",
            "AR/VR development",
            "Robotics",
            "Big data analytics",
            "Bioinformatics",
            "Network engineering",
            "Database management",
            "Information security");

    public static List<String> SKILLS = List.of("Java",
            "Python",
            "JavaScript",
            "C++",
            "C#",
            "HTML",
            "CSS",
            "SQL",
            "React",
            "Angular",
            "Vue.js",
            "Node.js",
            "Spring Framework",
            "Hibernate",
            "Express.js",
            "MongoDB",
            "MySQL",
            "PostgreSQL",
            "Git",
            "Docker",
            "Kubernetes",
            "Jenkins",
            "AWS",
            "Azure",
            "Google Cloud Platform",
            "Machine Learning",
            "Deep Learning",
            "Data Mining",
            "Data Analysis",
            "Cybersecurity",
            "Network Security",
            "Penetration Testing",
            "UI/UX Design",
            "Adobe Photoshop",
            "Sketch",
            "Figma",
            "Unity",
            "Unreal Engine",
            "Game Development",
            "Android Development",
            "iOS Development",
            "Windows Development",
```

```
                "Linux Administration",
                "Shell Scripting",
                "Raspberry Pi",
                "Arduino",
                "Embedded Systems",
                "Blockchain",
                "Cryptocurrency",
                "Smart Contracts",
                "AR/VR Development",
                "Internet of Things (IoT)");
    }
```

Лістинг 3.2 – Код для генерації резюме працівника

```
    public void generateResume() {
        User user = new User();
        user.setArea(AREAS.get(random.nextInt(AREAS.size())));
        user.setLevel(Level.values()[random.nextInt(Level.values().length)]);

        int randomCount = random.nextInt(SKILLS.size()) + 1;
        user.setSkills(generateListOfSkills(new              ArrayList<>(SKILLS),
randomCount));
        writeToJson("src/main/resources/resumes.json", user);
     }

    private List<Skill> generateListOfSkills(List<String> copySkillsName, int count)
{
        Collections.shuffle(copySkillsName);

        List<Skill> skills = new ArrayList<>();
        for (int j = 0; j < count; j++) {
            Skill skill = new Skill();
            skill.setName(copySkillsName.get(j));
            skill.setLevel(random.nextInt(4) + 1);
            skills.add(skill);
        }
        return skills;
    }

    private void writeToJson(String file, Object obj) {
        Gson gson = new Gson();
        try (FileWriter writer = new FileWriter(file)) {
            gson.toJson(obj, writer);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Далі розглянемо генерацію списку проєктів. Код поданий у лістингу 3.3. Було згенеровано 1000 проєктів. Результат генерації зображений на рисунку 3.2.

Лістинг 3.3 – Код генерації списку проєктів

```
public void generateProjectsData() {
        List<Project> projects = new ArrayList<>();
```

```
            for (int i = 1; i <= 1000; i++) {
                Project project = new Project();

                project.setArea(generateRandomListOfAreas(5));

project.setRequiredLevel(Level.values()[random.nextInt(Level.values().length)]);

                List<String> skills = new ArrayList<>(SKILLS);
                int randomCount = random.nextInt(Math.min(SKILLS.size(), 10)) + 1;
                List<Skill>     requiredSkills     =     generateListOfSkills(skills,
randomCount);
                project.setRequiredSkills(requiredSkills);

                int    randomCount2    =    random.nextInt(Math.min(SKILLS.size()    -
randomCount, 10)) + 1;
                List<Skill>                        desiredSkills                        =
generateListOfSkills(skills.subList(randomCount, skills.size()), randomCount2);
                project.setDesiredSkills(desiredSkills);

                projects.add(project);
            }
            writeToJson("src/main/resources/projects.json", projects);
        }
    private List<String> generateRandomListOfAreas(int maxCount) {
            int randomCount = random.nextInt(maxCount) + 1;
            List<String> copyAreas = new ArrayList<>(AREAS);
            Collections.shuffle(copyAreas);
            return new ArrayList<>(AREAS.subList(0, randomCount));
        }
```

resumes.json | projects.json | RandomGenerator.java | RandomConstants.java | courses.json

```
{
  "area": "Mobile development",
  "level": "MIDDLE",
  "skills": [
    {
      "name": "AWS",
      "level": 4
    },
    {
      "name": "Hibernate",
      "level": 2
    },
    {
      "name": "Unreal Engine",
      "level": 4
    },
    {"name": "Java"...},
    {"name": "Python"...},
    {"name": "Penetration Testing"...},
    {"name": "Unity"...},
    {"name": "Android Development"...},
    {"name": "Game Development"...},
    {"name": "Spring Framework"...},
    {"name": "Adobe Photoshop"...},
    {"name": "HTML"...},
    {"name": "Shell Scripting"...},
    {"name": "iOS Development"...},
    {"name": "Cybersecurity"...}
  ]
}
```

Рисунок 3.1 – Згенероване резюме працівника

Далі розглянемо генерацію списку курсів. Код поданий у лістингу 3.4. Було згенеровано 1000 курсів. Результат генерації зображений на рисунку 3.3.

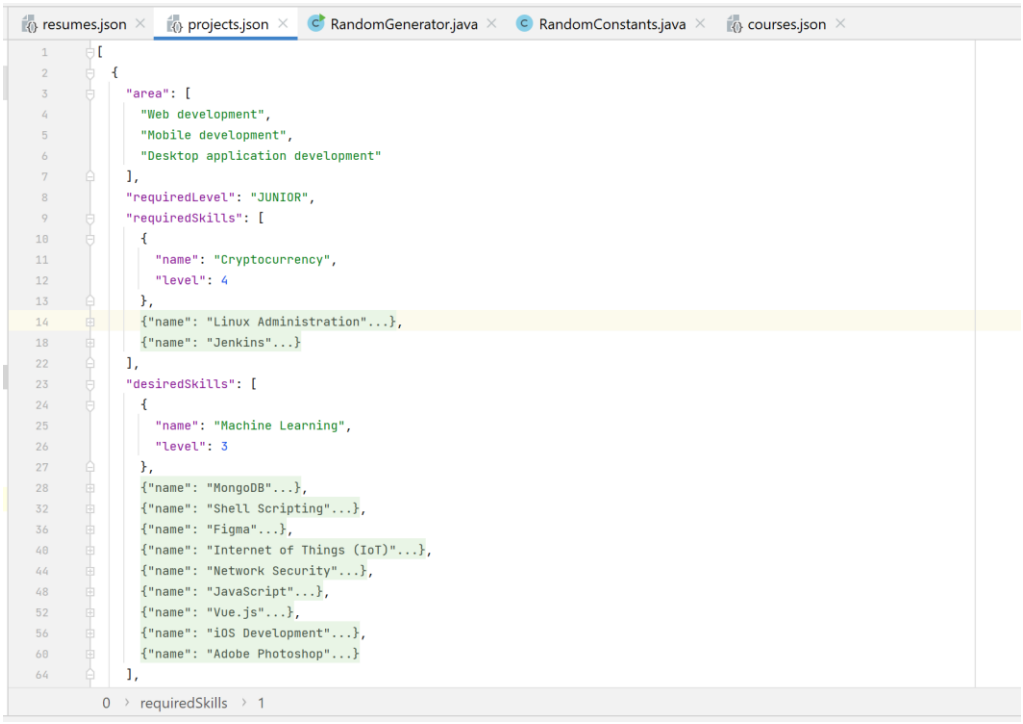

Рисунок 3.2 – Згенеровані дані проєктів

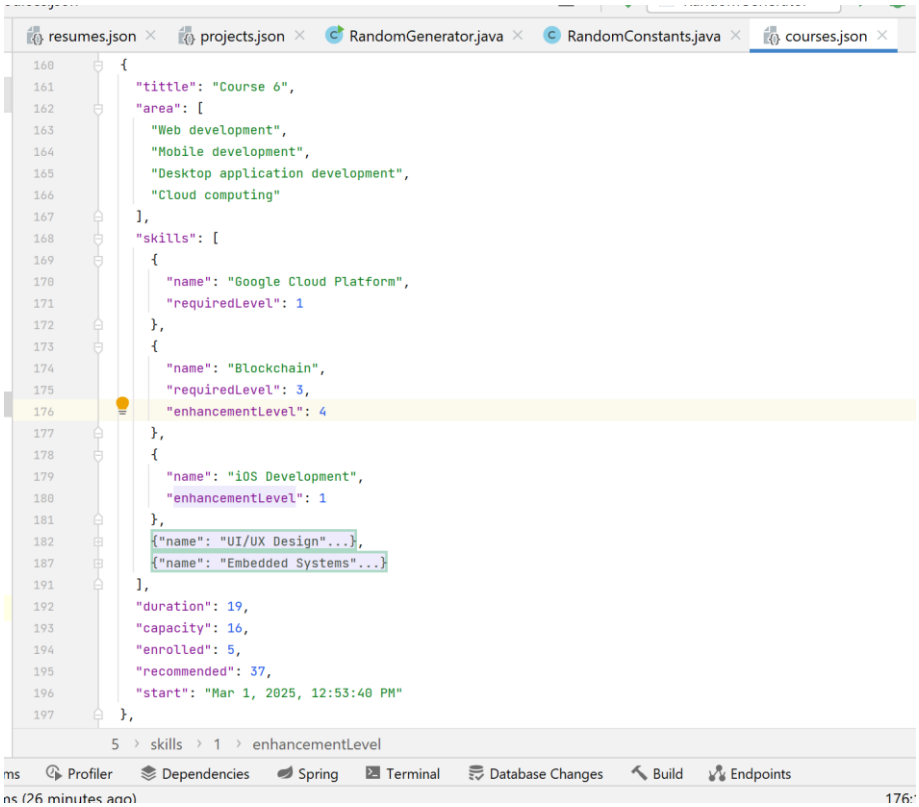

Рисунок 3.3 – Згенеровані дані курсів

Лістинг 3.4 – Код генерації списку курсів

```
public void generateCoursesData() {
        List<Course> courses = new ArrayList<>();

        for (int i = 1; i <= 1000; i++) {
            Course course = new Course();
            course.setTittle("Course " + i);
            course.setArea(generateRandomListOfAreas(4));
            course.setCapacity(random.nextInt(200) + 1);
            course.setEnrolled(random.nextInt(course.getCapacity()));
            course.setDuration(random.nextInt(350) + 1);
            course.setRecommended(random.nextInt(course.getCapacity() * 4));

            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.DAY_OF_YEAR, random.nextInt(500) + 1);
            course.setStart(calendar.getTime());

            List<String> copySkills = new ArrayList<>(SKILLS);
            Collections.shuffle(copySkills);

            List<CourseSkill> skills = new ArrayList<>();
            int randomCount = random.nextInt(Math.min(SKILLS.size(), 5)) + 1;
            for (int j = 0; j < randomCount; j++) {
                CourseSkill skill = new CourseSkill();
                skill.setName(copySkills.get(j));
                boolean hasRequired = random.nextBoolean();
                if (hasRequired) {
                    skill.setRequiredLevel(random.nextInt(4) + 1);
                }
                if (!hasRequired) {
                    skill.setEnhancementLevel(random.nextInt(4) + 1);
                } else if (random.nextBoolean() && skill.getRequiredLevel() < 4)
{
                    int enhancementLevel = skill.getRequiredLevel() +
random.nextInt(3) + 1;
                    skill.setEnhancementLevel(Math.min(enhancementLevel, 4));
                }
                skills.add(skill);
            }
            course.setSkills(skills);
            courses.add(course);
        }
        writeToJson("src/main/resources/courses.json", courses);
    }
```

В результаті ми отримали підготовлені дані для тестування алгоритму.

3.2 Проведення дослідження та отримання результату

Використовуючи тестові вибірки даних, запустимо алгоритм, та отримаємо відповідь: курс (виберемо лише назву) та його відстань до ідеальної точки [14]. Список повинен бути відсортований по зростанню. Чим менша

відстань до ідеальної точки, тим більше курс повинен мати пріоритет рекомендації. Отримаємо список із 162 елементів. Результат зображено на рисунку 3.4.

```
C:\Users\Vlad\.jdks\corretto-11.0.13\bin\java.exe ...
List size:162
Course tittle:Course 577 Priority:0.027926573681315563
Course tittle:Course 939 Priority:0.06947820439744021
Course tittle:Course 235 Priority:0.07752727378368698
Course tittle:Course 74 Priority:0.11134052501698585
Course tittle:Course 430 Priority:0.1407471796016102
Course tittle:Course 980 Priority:0.14173260788468572
Course tittle:Course 944 Priority:0.14281141789312554
Course tittle:Course 853 Priority:0.1873247090114678
Course tittle:Course 402 Priority:0.19402287346104294
Course tittle:Course 32 Priority:0.2030939960913691
Course tittle:Course 921 Priority:0.20356228381470107
Course tittle:Course 314 Priority:0.20424130026517281
Course tittle:Course 220 Priority:0.2079620557236549
Course tittle:Course 952 Priority:0.2098959179823574
Course tittle:Course 201 Priority:0.2301151534200472
Course tittle:Course 903 Priority:0.2499451666711992
Course tittle:Course 990 Priority:0.25624950173864947
Course tittle:Course 145 Priority:0.26613541330977575
Course tittle:Course 461 Priority:0.2679520975651539
Course tittle:Course 377 Priority:0.27676524255353707
Course tittle:Course 628 Priority:0.28141892799487195
Course tittle:Course 352 Priority:0.28891735896528364
Course tittle:Course 985 Priority:0.29697162994162274
Course tittle:Course 516 Priority:0.29883207307471216
Course tittle:Course 796 Priority:0.30350930199618187
Course tittle:Course 280 Priority:0.31106812967952385
Course tittle:Course 339 Priority:0.320521464419349
Course tittle:Course 799 Priority:0.3222089753166807
Course tittle:Course 663 Priority:0.3269008710088077
```

Рисунок 3.4 – Результат виконання алгоритму

3.3 Аналіз отриманих результатів

Розглянемо перший та останній курси в списку. Перший курс з назвою «Course 577» має пріоритет 0.027926573681315563 та поданий у лістингу 3.5. Другий курс «Course 334» має пріоритет 0.8649216674103017 та поданий у лістингу 3.6.

Лістинг 3.5. – Дані курсу «Course 577»


```
{
    "tittle": "Course 577",
    "area": [
      "Web development",
      "Mobile development",
      "Desktop application development",
      "Cloud computing"
    ],
    "skills": [
      {
        "name": "Smart Contracts",
        "enhancementLevel": 4
      },
      {
        "name": "Deep Learning",
        "enhancementLevel": 3
      },
      {
        "name": "Express.js",
        "enhancementLevel": 4
      }
    ],
    "duration": 78,
    "capacity": 28,
    "enrolled": 5,
    "recommended": 72,
    "start": "Sep 17, 2024, 12:53:40 PM"
  }
```


Лістинг 3.6. – Дані курсу «Course 334»


```
{
    "tittle": "Course 334",
    "area": [
      "Web development",
      "Mobile development",
      "Desktop application development"
    ],
    "skills": [
      {
        "name": "Adobe Photoshop",
        "enhancementLevel": 3
      }
    ],
    "duration": 28,
    "capacity": 160,
    "enrolled": 153,
    "recommended": 445,
    "start": "Apr 14, 2024, 12:53:40 PM"
  }
```


Перш за все, перевіримо виконання вимог. Перша вимога полягає у співпадінні сфери розробки курсу та працівника («Mobile development»). Кожен курс має цю сферу в своєму списку, тому ця вимога виконана. Друга вимога

передбачає, що якщо курс має необхідні навички, то резюме працівника повинне відповідати цим вимогам. Однак обидва курси не мають таких навичок.

Далі, ми здійснюємо порівняння курсів і доводимо, що алгоритм правильно розрахував пріоритети, показуючи, що перший курс є більш необхідним для працівника, ніж другий.

Розрахуємо критерій «кількість вільних місць» для обох курсів. Отримаємо 0.82 і 0.04375. Тобто за цим критерієм, перший курс є кращим.

Розрахуємо критерій «відношення рекомендацій до вільних місць». Отримаємо 3, 13 та 63,574. Без використання експоненійної функції зрозуміло, що чим вище значення, тим більше рекомендацій на 1 вільне місце, тим нижче повинен бути пріоритет. Тому за цим критерієм перший курс є кращим.

Розрахуємо критерій «пріоритет вивчення курсу на основі навиків». Для цього подивимось на навички, що покращують курси. Перший курс покращує 3 навичка, кожен з яких згадується в 200 проєктах. Другий курс лише 1 навичок, який також згадується в 200 проєктах. Для порівняння, порівняємо рівень покращення навиків. Для першого курсу, навички підвищуються з 0 до 4, 3 та 4 рівня відповідно. Для другого один навичок з 0 до 3 рівня. Тому за цим критерієм перший курс є кращим.

Розрахуємо критерій «кількість годин на вивчення курсу». Спочатку отримаємо максимальну кількість годин в тестових даних. Отримали значення в 348 годин. Далі розрахуємо значення критеріїв та отримаєм значення 0.776 та 0.92 відповідно. За цим критерієм другий курс є кращим. Але враховуючи попередні три критерія, які в рази краще і мають загальну вагу 0.9 (даний критерій має вагу лише 0.1), то в загальному пріоритеті перевагу має перший курс.

Після аналізу результатів був зроблений висновок, що алгоритм працює правильно і перший курс дійсно більш корисний для вивчення працівником ніж другий.

4 ПРОЕКТУВАННЯ ТА РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ З ІНТЕГРОВАНИМ МЕТОДОМ ПІДБОРУ ТА РОЗПОДІЛУ РЕКОМЕНДОВАНИХ КУРСІВ ПРАЦІВНИКАМ IT-КОМПАНІЇ

4.1 Проектування системи

Для проектування системи проведемо UML-моделювання [15] для визначення основних функцій системи та їх опису. Спочатку розробимо діаграму використання, Use Case, що зображена на рисунку 4.1.

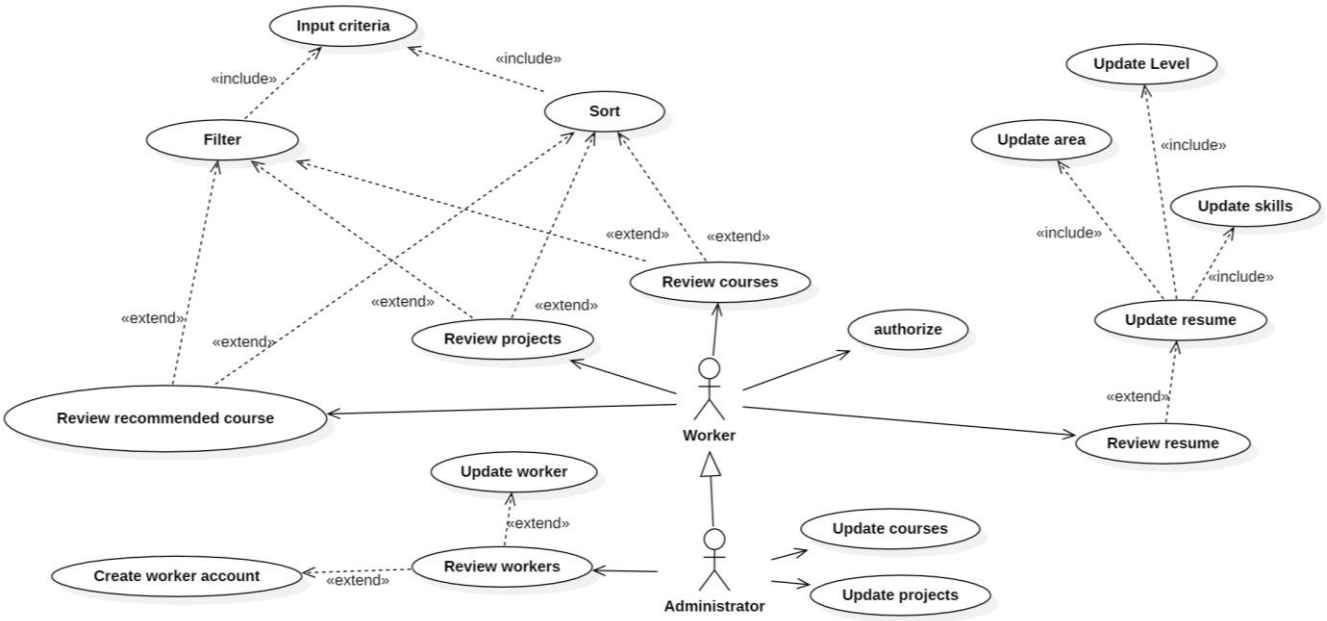

Рисунок 4.1 – Use Case діаграма

Проаналізуємо діаграму використання. Були визначені наступні ролі акторів.

«Працівник» – роль, що видається співробітнику IT-компанії після авторизації в системі. Має доступ до таких прецедентів, що визначають бізнес-функції рекомендаційної системи IT-компанії для підбору рекомендованих курсів для покращення навичок:

– «Авторизація» – функція надання користувачеві, на основі введених логіна та пароля, прав на виконання функцій системи.

– «Перегляд резюме» – функція, що надає користувачеві інформацію про власне резюме;

– «Оновлення резюме» – функція, що надає користувачеві можливість редагувати дані резюме. Є розширенням функції «Перегляд резюме». Включає в себе такі функції як оновлення сфери розробки, оновлення рівня працівника, оновлення навичок;

– «Перегляд проєктів» – функція, що відображає користувачеві список проєктів та їх дані;

– «Перегляд курсів» –функція, що відображає користувачеві список курсів та їх дані;

– «Перегляд рекомендованих курсів» – функція, що відображає користувачеві список рекомендованих курсів по спаданню пріоритету.

– «Сортування» – функція, за допомогою якої користувач може відсортувати список. Включає в себе вибір критерію для сортування. Є розширенням таких функцій «Перегляд проєктів», «Перегляд курсів» та «Перегляд рекомендованих курсів»;

– «Фільтрація» – функція, за допомогою якої користувач може відфільтрувати список за умовою. Включає в себе вибір критерію для фільтрації. Є розширенням таких функцій «Перегляд проєктів», «Перегляд курсів» та «Перегляд рекомендованих курсів».

«Адміністратор» – роль, яка має додатковий функціонал для управління рекомендаційною системою підбору та розподілу курсів між працівниками IT-компанії. Має доступ до наступних прецедентів:

– «Оновлення проєктів» – функція для управління даними проєктів. Може редагувати, видаляти або додавати нові проєкти та вимоги до них;

– «Оновлення курсів» – функція для управління даними курсів. Може редагувати, видаляти або додавати нові курси;

– «Перегляд працівників» – функція за допомогою якої адміністратор може переглянути список працівників та їх дані у вигляді резюме;

– «Створення акаунту працівника» – функція для створення нового акаунту для працівника. Є розширенням функції «Перегляд працівників»;

– «Оновлення даних працівника» – функція, за допомогою якої адміністратор може оновити дані працівника або видалити. Є розширенням функції «Перегляд працівників».

Була розроблена діаграма класів [16] для подальшої реалізації веб-серверу (рис. 4.2).

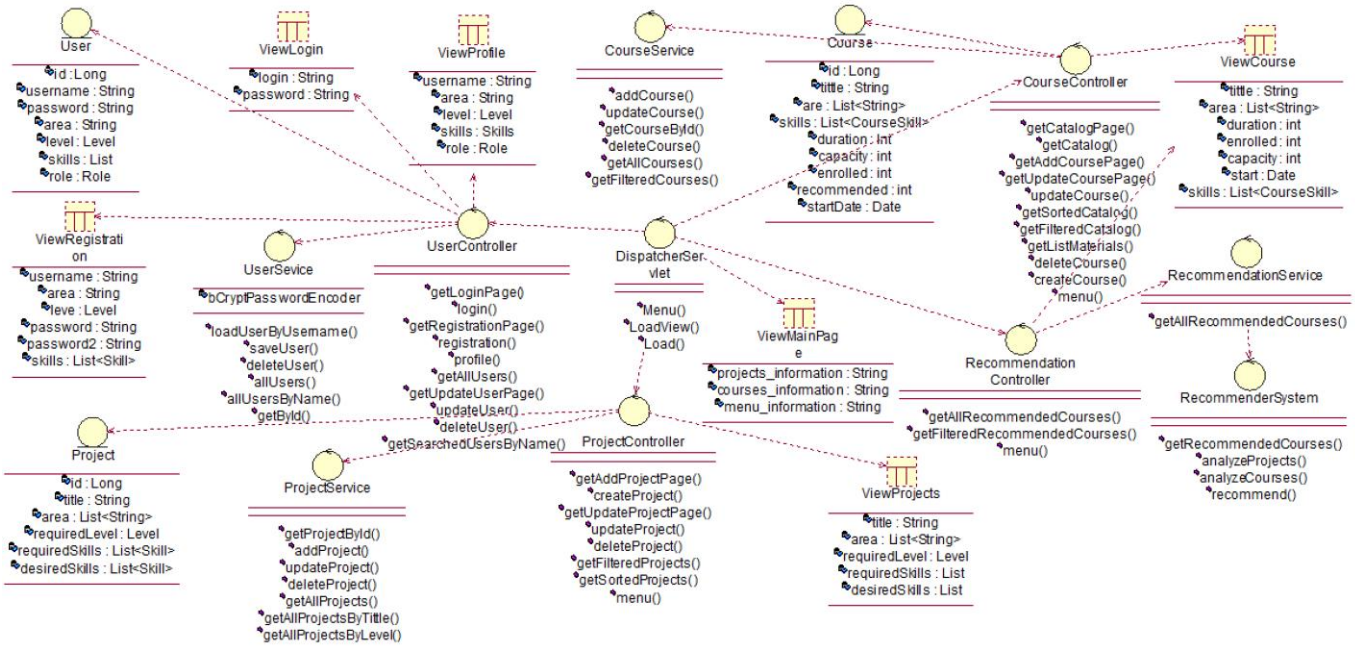

Рисунок 4.2 – Діаграма класів

На діаграмі класів подано наступні класи.

Клас «DispatcherServlet» - який, відповідає за переадресацію запитів на конкретні контролери, для подальшої обробки. Також забезпечує обробку HTTP-запитів із використанням Thymeleaf для відображення динамічного контенту, наприклад, даних з бази даних чи з форм введення користувачем. Таким чином, DispatcherServlet та Thymeleaf співпрацюють для створення динамічних веб-сторінок у додатку на основі Spring Framework.

Клас «UserController» - клас-контролер, що виконує запити на обробку даних користувача та повертає сторінки з моделями даних.

Методи, що містить клас «UserController»:

– getLoginPage() – метод, що забезпечує отримання сторінки авторизації;

– login() – метод, що забезпечує авторизацію та автентифікація користувача в системі. Повертає користувача та його роль;

– getRegistrationPage – метод, що забезпечує отримання сторінки реєстрації користувача;

– registration() – метод, що забезпечує створення користувача в системі та збереження його даних в базі даних;

– profile() – метод, що забезпечує відображення сторінки з даними вибраного користувача;

– getAllUsers() – метод, що забезпечує відображення сторінки з даними користувачами;

– getUpdateUserPage() – метод, що забезпечує отримання сторінки оновлення даних користувача;

– updateUser() – метод, що забезпечує оновлення даних користувача в системі;

– deleteUser() – метод, що забезпечує видалення даних користувача зі системи;

– getSearchedUsersByName() – метод, що забезпечує відображення сторінки з користувачами знайденими пошуком за іменем.

Клас «RecommendationController» - клас-контролер, що виконує запити на обробку даних рекомендаційних курсів для співробітника та повертає сторінки з моделями даних.

Методи, що містить клас «RecommendationController»:

– getAllRecommendedCourses() – метод, що забезпечує отримання сторінки з даними всіх курсів, що рекомендовані авторизованому користувачу;

– getFilteredRecommendedCourses () – метод, що забезпечує фільтрацію рекомендованих курсів за назвою.

Клас «ProjectController» - клас-контролер, що виконує запити на обробку

даних проєктів та повертає сторінки з моделями даних.

Методи, що містить клас «ProjectController»:

– getAddProjectPage() – метод, що забезпечує отримання сторінки додавання нового проєкту;

– createProject() – метод, що забезпечує створення проєкту в системі та додавання в базу даних;

– getUpdateProjectPage – метод, що забезпечує отримання сторінки оновлення даних проєкту;

– updateProject() – метод, що забезпечує оновлення даних проєкту в інформаційній системі;

– deleteProject() – метод, що забезпечує видалення даних проєкту з системи та бази даних;

– getFilteredProjects() – метод, що забезпечує отримання сторінки з відфільтрованим списком проєктів;

– getSortedProjects() – метод, що забезпечує отримання сторінки з відсортованим списком проєктів.

Клас «CourseController» - клас-контролер, що виконує запити на обробку даних курсів та повертає сторінки з моделями даних.

Методи, що містить клас «CourseController»:

– getCatalog() – метод, що забезпечує отримання сторінки з каталогом курсів з їх основними даними;

– getAddCoursePage() – метод, що забезпечує отримання сторінки додавання нового курсу;

– createCourse() – метод, що забезпечує створення курсу в системі та додавання в базу даних;

– getUpdateCoursePage – метод, що забезпечує отримання сторінки оновлення даних курсу;

– updateCourse() – метод, що забезпечує оновлення даних курсу в інформаційній системі;

– deleteCourse() – метод, що забезпечує видалення даних курсу з системи та бази даних;

– getFilteredCatalog() – метод, що забезпечує отримання сторінки з відфільтрованим списком курсів;

– getSortedCatalog() – метод, що забезпечує отримання сторінки з відсортованим списком курсів.

Клас «UserService» - клас-сервіс, що виконує основну бізнес-логіку для даних пов’язаних з користувачем.

Поля, що містить клас:

– BCryptPasswordEncoder – об’єкт класу, що відповідає за шифрування пароля користувача у вигляді хешу.

Методи, що містить клас «UserService»:

– loadUserByUsername() – метод, що забезпечує отримання об’єкта користувача за допомогою логіну;

– saveUser() – метод, що забезпечує збереження даних користувача в базі даних;

– getById () – метод, що забезпечує отримання даних користувача за його ідентифікатором;

– allUsers() – метод, що забезпечує отримання списку всіх користувачів;

– deleteUser() – метод, що забезпечує видалення даних користувача із бази даних;

– allUsersByName() – метод, що забезпечує списку користувачів знайденими пошуком за іменем.

Клас «RecommendationService» - клас-сервіс, що виконує основну бізнес-логіку на отримання рекомендаційних курсів для співробітника.

Методи, що містить клас «RecommendationService»:

– getAllRecommendedCourses() – метод, що забезпечує отримання списку рекомендаційних курсів.

Клас «ProjectService» - клас-сервіс, що виконує основну бізнес-логіку для

обробки даних проєктів.

Методи, що містить клас «ProjectService»:

– addProject() – метод, що забезпечує додавання проєкту в базу даних;

– updateProject() – метод, що забезпечує оновлення даних проєкту в інформаційній системі;

– deleteProject() – метод, що забезпечує видалення даних проєкту з бази даних;

– getAllProjects() – метод, що забезпечує отримання списку всіх проєктів;

– getAllProjectsByTittle() – метод, що забезпечує отримання списку всіх проєктів знайдених за допомогою пошуку по імені;

– getAllProjectsByLevel() – метод, що забезпечує отримання списку всіх проєктів знайдених за допомогою пошуку по рівню;

– getProjectById() – метод, що забезпечує отримання об’єкта проєкту за допомогою ідентифікатора.

Клас «CourseService» – клас-сервіс, що виконує основну бізнес-логіку обробки даних курсів.

Методи, що містить клас «CourseService»:

– getAllCourses() – метод, що забезпечує отримання списку всіх курсів;

– addCourse() – метод, що забезпечує додавання курсу в базу даних;

– updateCourse() – метод, що забезпечує оновлення даних курсу;

– deleteCourse() – метод, що забезпечує видалення даних курсу з бази даних;

– getFilteredCourses() – метод, що забезпечує отримання списку відфільтрованих курсів;

– getCourseById() – метод, що забезпечує отримання курсу за ідентифікатором.

Клас «RecommenderSystem» – клас-сервіс, що виконує основну бізнес-логіку методу підбору та розподілу курсів серед співробітників IT-компанії.

Методи, що містить клас «RecommenderSystem»:

– getRecommendedCourses() – метод, що використовуючи наступні три функції, повертає список курсів відсортованих за пріоритетом;

– analyzeProjects() – метод, що реалізує 2 етап алгоритму метода підбору та розподілу рекомендаційних курсів, а саме аналіз проєктів. Повертає словник навичок та їх пріоритет вивчення;

– analyzeCourses() – метод, що реалізує 3 етап алгоритму метода підбору та розподілу рекомендаційних курсів, а саме аналіз курсів. Повертає список відфільтрований список курсів з їх визначеними пріоритетами;

– recommend() – реалізує метод ідеальної точки, що за допомогою коефіцієнтів розраховує пріоритет вивчення курсу.

Клас-сутність «User» зберігає інформацію про користувача та має наступні поля: «id», «username», «password», «area», «level», «skills», «role».

Клас-сутність «Project» зберігає інформацію про проєкт та має наступні поля: «id», «title», «requiredLevel», «area», «requiredSkills», «desiredSkills».

Клас-сутність «Course» зберігає інформацію про курс та має наступні поля: «id», «title», «skills», «area», «duration», «capacity», «enrolled», «recommended», «startDate».

Клас-відображення «ViewLogin» містить інформацію для авторизації користувача та має наступні поля: «username», «password».

Клас-відображення «ViewRegistration» містить інформацію для реєстрації користувача та має наступні поля: «username», «password», «passwordConfirm» «area», «level», «skills».

Клас-відображення «ViewProfile» містить інформацію про користувача та має наступні поля: «username», «area», «level», «skills», «role».

Клас-відображення «ViewProject» відображає дані проєкту та має наступні поля: «title», «requiredLevel», «area», «requiredSkills», «desiredSkills».

Клас-відображення «ViewCourse» відображає дані про курс та має наступні поля: «title», «skills», «area», «duration», «capacity», «enrolled», «start».

Діаграму послідовності для методу підбору та розподілу рекомендованих курсів IT-компанії між співробітниками зображено на рисунку 4.3. Актор «Worker» переходить на сторінку «Recommendations Courses», яка відправляє запит «/recommendations» на «DispatcherServlet». Далі «DispatcherServlet» обробляє запит і відправляє на клас-контролер «RecommendationController». Далі запит обробляє «RecommendationController» та викликає метод «getAllRecommendedCourses» класу «RecommendationService». Всередині цього методу викликається інший метод «findByCriteria» цього ж класу, який повертає список проєктів за критерієм «area» та список курсів знайдених в базі даних за критеріями «area» та «start». Далі викликається метод «getRecommendedCourses» класу «RecommenderSystem» в який передається такі дані як, «user», «projects», «courses». Поточний метод спочатку викликає метод «analyzeProjects», в який передаються дані «user» та «projects», для аналізу проєктів та отримання словника навичок, необхідних для вивчення та їх пріоритет. Далі викликається метод «analyzeCourses» для аналізу курсів передаючи дані «user», «skillMark», «courses» та отримується список курсів з пріоритетами. Далі викликається метод «recommend» з параметром «coursesWithPriority» для отримання курсів з остаточним пріоритетом вивчення. Потім цей список сортується за пріоритетом та повертається в «RecommendationService» та «RecommendationController» , де викликаються методи «findPaginated» та «addPaginationModel» для додавання пагінації в модель, і повертається модель з контентом в «DispatcherServlet». Останній створює відображення у вигляді «courses.html», що динамічно заповнюється даними та сторінка повертається користувачеві.

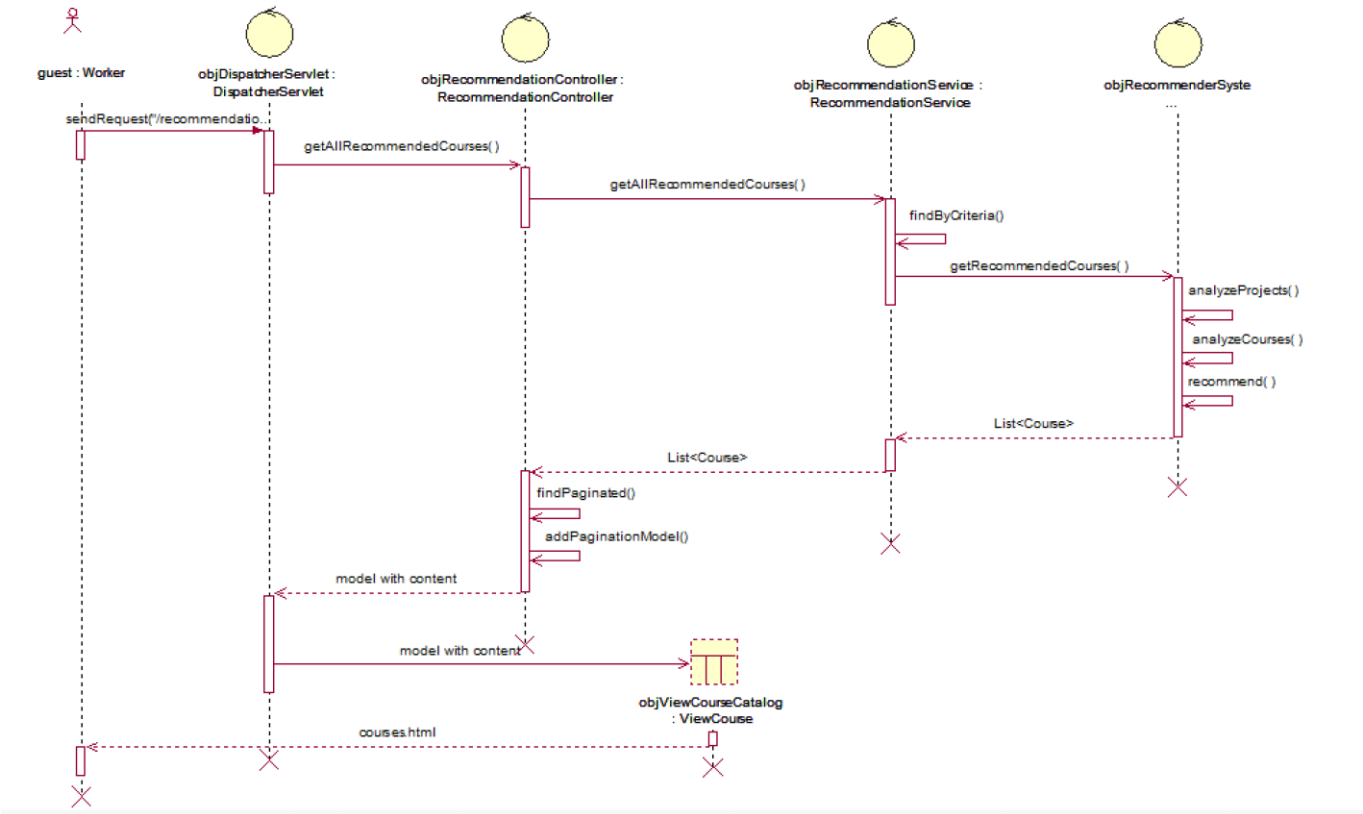

Рисунок 4.3 – Діаграма послідовності для процесу «Перегляд рекомендаційних курсів»

4.2 Розробка архітектури

Для рекомендаційної системи підбору та розподілу курсів між працівниками IT-компанії була вибрана архітектура «Клієнт-сервер» [17], що складається із трьох підсистем: «Вебклієнт», «Вебсервер» та «Сервер бази даних». Розглянемо кожну підсистему окремо.

Сервер бази даних реалізується за допомогою технології «MongoDB» [18] та використовується для зберігання даних та виконання запитів до них.

Вебсервер рекомендаційної системи реалізована за допомогою технології «Spring Boot» [19] та використовується для реалізації бізнес-логіки. Має наступні складові:

– «Spring Rest Controller» – контролер основна функція якого це обробка http-запитів, що відправляються з веб-клієнта. Кожен урл обробляється контролером;

– «Services» – компонент що виконує основну логіку функції;

– «Thymeleaf View» – відображає дані на клієнтській стороні у вигляді html файлів з даними, що передаються за допомогою «Java»;

– «Model» – забезпечує зберігання даних, які отримані з бази даних;

– «Spring Data JPA» [20] – забезпечує отримання даних з БД за допомогою виконання CRUD запитів. Основними операціями є вибірка даних, додавання, оновлення, видалення.

Веб-клієнт представлений у вигляді браузері, що відображає сторінки з даними користувачам.

Для розробки були вибрані технології «Java» [21] а саме «Spring Boot Framework» для розробки бізнес-логіки та «MVC» [22] для каркасу додатку, а також «Thymeleaf» для забезпечення створення сторінок з даними для користувачів. Також використовувалися технології «Spring Security» [23] для забезпечення захисту додатку та «Junit» [24] для забезпечення тестування.

Архітектура системи подана на рисунку 4.4.

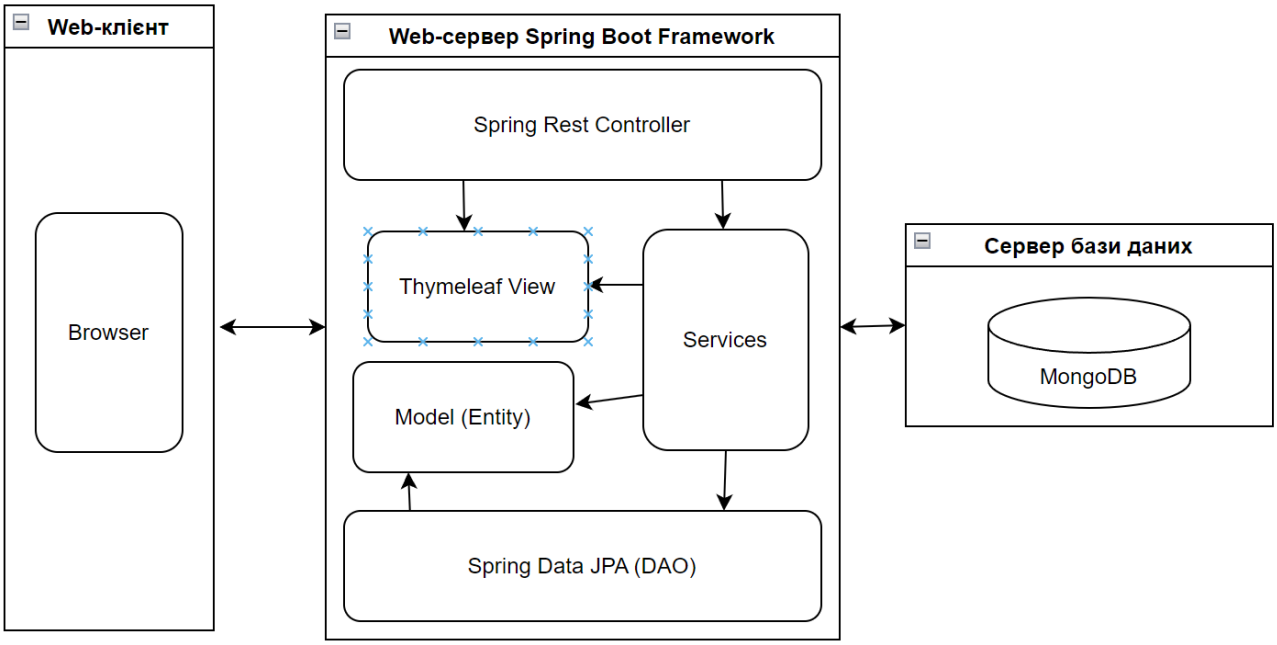

Рисунок 4.4. – Архітектура системи

4.3 Розробка серверу бази даних

Фізична модель даних була реалізована на платформі MongoDB за допомогою таких програмних забезпечень як MongoDBCompass та Hackolade та подана на рисунку 4.5. Використовувалась NoSQL база даних замість SQL обґрунтована наступними причинами.

Гнучкість у схемі даних. NoSQL бази даних дозволяють гнучко організовувати схему даних без строгих вимог до структури. Це особливо корисно, коли потрібно зберігати дані різної природи або змінної структури. Дані курсів, проєктів, користувачів не перетинаються і мають різні цілі використання. Також дані навичок мають змінну структуру, якщо в проєкті мають лише два поля, такі як назва та рівень, то в курсах, навички мають три поля, назва, необхідний рівень та рівень підвищення. Також структура може змінюватися, тому необхідна гнучка схема даних.

Швидкий доступ до даних. NoSQL бази даних, такі як MongoDB, можуть бути дуже ефективними у виконанні запитів на великі обсяги даних. Це особливо корисно у великих проектах, де потрібен швидкий доступ до великої кількості даних. Кількість проєктів може сягати більше 100 тисяч, а курсів за весь час десятки тисяч, враховуючи, що важливо також зберігати курси які пройшли вже.

Масштабованість. NoSQL бази даних добре масштабуються горизонтально та вертикально, що дозволяє легко розширювати обсяг даних і кількість запитів без значного збільшення витрат на обслуговування бази даних.

Підтримка гнучких типів даних. MongoDB підтримує різноманітні типи даних, такі як масиви, вбудовані документи та геолокаційні дані, що дозволяє зберігати та обробляти різнорідну інформацію.

Дані системи мають різноманітну структуру та включають в себе інформацію про користувачів, проєкти та курси з різними навичками та рівнями навичок. Це є типовим сценарієм для використання NoSQL бази даних, так як вона дозволяє зручно зберігати та обробляти такі дані без обмежень стандартної SQL схеми даних.

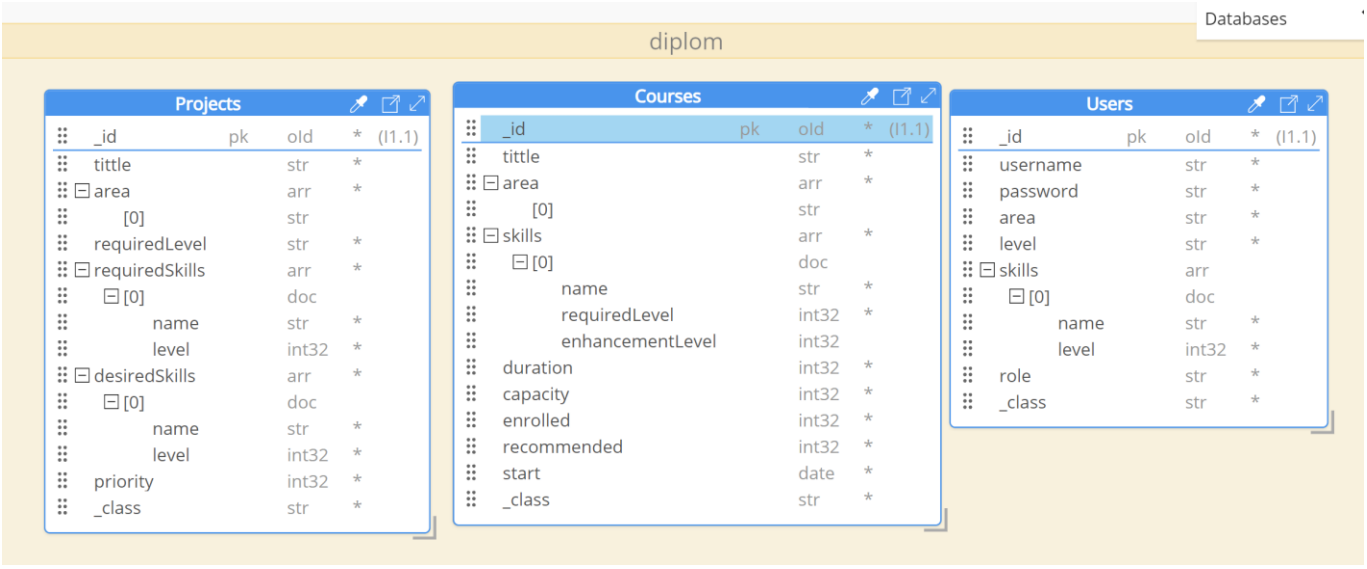

Рисунок 4.5 – Схема бази даних

Колекція «Проєкти», містить дані проєктів інформаційної системи підбору та розподілу рекомендованих курсів. Має наступні поля:

– _id – унікальний ідентифікатор проєкту в базі даних MongoDB.

– title – назва проєкту;

– area – область, до якої відноситься проєкт, наприклад, "Розробка кібербезпеки" або "Розробка блокчейну»;

– requiredLevel – необхідний рівень для участі в проекті, наприклад, "JUNIOR" або "SENIOR";

– requiredSkills – список необхідних навичок для проекту, кожна з яких має назву (name) і рівень володіння (level);

– _class – клас об'єкта, в якому зберігаються дані проекту.

Колекція «Користувачі», містить дані користувачів інформаційної системи підбору та розподілу рекомендованих курсів. Має наступні поля:

– _id – унікальний ідентифікатор користувача в базі даних MongoDB.

– username – логін користувача для входу в систему;

– password – хешований пароль користувача для забезпечення безпеки;

– area – область діяльності або інтереси користувача, наприклад,

"Тестування" або "Розробка";

– level – рівень навичок користувача, наприклад, "MIDDLE" або "SENIOR";

– skills – список навичок користувача, кожна з яких має назву (name) і рівень володіння (level);

– role – роль користувача в системі, наприклад, "ROLE_USER" або "ROLE_ADMIN";

– _class – клас об'єкта, в якому зберігаються дані користувача.

Колекція «Курси», містить дані курсів інформаційної системи підбору та розподілу рекомендованих курсів. Має наступні поля:

– _id – унікальний ідентифікатор курсу в базі даних MongoDB.

– title – назва курсу;

– area – область, до якої відноситься курс, наприклад, "Розробка блокчейну" або "Вбудовані системи";

– skills – список навичок, які можна отримати на курсі, кожна з яких має назву (name) та вимогу рівня (requiredLevel) і можливість покращення рівня (enhancementLevel);

– duration – тривалість курсу в днях або годинах;

– capacity – максимальна кількість студентів, яку може вмістити курс;

– enrolled – кількість зареєстрованих студентів на курсі;

– recommended – кількість рекомендованих студентів для курсу;

– start – дата початку курсу;

– _class – клас об'єкта, в якому зберігаються дані курсу.

4.4 Розробка карти сайту

Для розробки карти сайту було використано деревоподібний тип структури. Сама карта сайту рекомендаційної системи підбору та розподілу рекомендованих курсів подана на рисунку 4.6.

Список сторінок сайту рекомендаційної системи.

– «Login» – сторінка авторизації працівника IT-компанії в системі. Без авторизації неможливо отримати доступ до системи. Містить поля для введення ім'я користувача та пароля, а також кнопку для входу. Після успішного входу користувач отримує доступ до свого профілю та інших функцій системи.

– «StartPage» – головна сторінка сайту, на якій зазвичай розміщуються загальна інформація про систему, новини, важливі повідомлення або посилання на інші ключові сторінки. Головна сторінка вітає користувача і надає шляхи для подальшої навігації по сайту;

– «ErrorPage» – ця сторінка відображається, коли виникає помилка або сторінка не може бути знайдена. Вона містить інформацію про помилку та можливі дії, які може виконати користувач для виправлення ситуації;

– «Profile» – сторінка користувача, де містяться всі його дані;

– «Logout» – функція для виходу користувача з системи. При натисканні на цю кнопку користувач виходить зі свого облікового запису і перенаправляється на сторінку входу;

– «Course catalog» – сторінка, де можна переглянути каталог курсів. На цій сторінці доступна фільтрація за рівнем проєкту та пошук за назвою;

– «Create course» – сторінка створення курсу в системі заповнюючи форму. Доступна лише адміністратору;

– «Update course» – сторінка оновлення даних курсу в системі заповнюючи форму. Доступна лише адміністратору;

– «Delete course» – функція, яка призначена для видалення курсів з системи. Вона зазвичай вимагає підтвердження видалення та виконує видалення курсу з бази даних після підтвердження. Доступна лише адміністратору;

– «Recommendation courses» – сторінка, яка призначена для відображення списку рекомендаційних курсів за допомогою методу підбору та розподілу курсів серед працівників IT-компанії;

– «Project catalog» – сторінка, де можна переглянути каталог проєктів та вимоги до них. На цій сторінці доступний пошук за назвою;

– «Create project» – сторінка створення проєкту в системі заповнюючи форму. Доступна лише адміністратору;

– «Update project» – сторінка оновлення даних проєкту в системі заповнюючи форму. Доступна лише адміністратору;

– «Delete project» – функція, яка призначена для видалення проєкту з системи. Вона зазвичай вимагає підтвердження видалення та виконує видалення курсу з бази даних після підтвердження. Доступна лише адміністратору;

– «Users» – сторінка, де можна переглянути список користувачів системи з їх основними даними. На цій сторінці доступний пошук за назвою. Сторінка доступна лише адміністатору;

– «Create user» – сторінка створення профілю працівника в системі заповнюючи форму. Доступна лише адміністратору;

– «Update user» – сторінка оновлення даних користувача в системі заповнюючи форму. Доступна лише адміністратору;

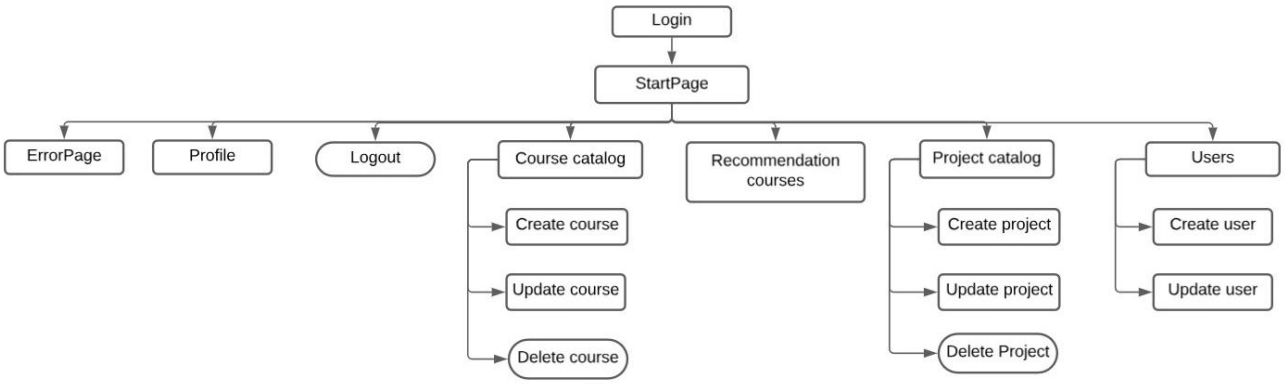

Рисунок 4.6 – Карта сайту

ВИСНОВКИ

Метою кваліфікаційної роботи було проведення аналізу предметної області, аналіз аналогів, постановка задачі, розробка методу підбору та розподілу рекомендаційних курсів серед працівників IT-компаній та проектування і розробка інформаційної системи з інтеграцією розробленого методу.

Спочатку був проведений аналіз предметної області, що включає в себе декілька етапів. Першим етапом є дослідження сфери працевлаштування в IT-компаніях та їх вимоги, рекомендаційних систем та їх основних методів. Також було проаналізовано методи для прийняття рішень. На останньому етапі була визначена та описана постановка задачі.

Далі була проведена розробка алгоритму підбору та розподілу рекомендованих курсів працівникам IT-компаній в декілька етапів. Перший етап – розробка схеми алгоритму та його вимог, на основі яких була розроблена структура. Далі були розроблені всі частини алгоритму: «аналіз резюме працівника», «аналіз вимог проєктів», «аналіз доступних курсів IT-компанії», «підбір рекомендованих курсів працівникам IT-компанії».

У третьому розділі були проведені експериментальні дослідження та проаналізовано результат для перевірки правильності алгоритму.

У останньому розділі проектувалася та розроблялася рекомендаційна система з інтеграцією методу підбору та розподілу рекомендованих курсів працівникам IT-компанії. Спочатку були створені UML-діаграми для визначення бізнес-функцій та їх детального опису. Далі розроблена архітектура додатку та визначено основні технології для реалізації системи. За допомогою визначеної бази даних «MongoDB» була створена схема та колекції на її основі. В решті був розроблений додаток та інтегровано метод підбору та розподілу рекомендованих курсів в нього.

Отже, розроблена рекомендаційна система з інтегрованим методом підбору та розподілу рекомендованих курсів між співробітниками IT-компанії.

Результати кваліфікаційної роботи пройшли апробацію на двох міжнародних конференціях [1, 25].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Батраченко В.О., Колесник Л.В. Розробка алгоритму підбору та розподілу рекомендаційних курсів працівникам IT-компанії // «Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві», 22 листопада 2023. Харків, Україна. 2023. С. 280-284.

2. Tarnowska K., Ras Z. W., Daniel L. Recommender System for Improving Customer Loyalty. Cham : Springer International Publishing, 2020. URL: https://doi.org/10.1007/978-3-030-13438-9 (date of access: 03.03.2024).

3. Recommender System Applications. Recommender Systems. 2020. P. 39–62. URL: https://doi.org/10.1142/9789811224638_0003 (date of access: 03.03.2024).

4. Saleh I., Chartron G., Kembellec G. Recommender Systems. Wiley & Sons, Incorporated, John, 2021. 252 p.

5. Agarwal D. K., Chen B.-C. Statistical Methods for Recommender Systems. Cambridge University Press, 2015.

6. Наконечний О. Г., Гребеннік І. В., Романова Т. Є., Тевяшев А. Д. Методи прийняття рішень : навч. посіб. Харків : ХНУРЕ, 2016. 132 с.

7. Dimanova D. Mathematical Methods for Decision-Making. Journal Scientific and Applied Research. 2015. Vol. 7, no. 1. P. 88–97. URL: https://doi.org/10.46687/jsar.v7i1.169 (date of access: 09.03.2024).

8. Веб-сайт дистанційних курсів IT-компанії «EPAM». URL: https://training.epam.ua (дата звернення: 14.03.2024).

9. Hansen S., Narayanan N. H., Hegarty M. Designing Educationally Effective Algorithm Visualizations. Journal of Visual Languages & Computing. 2002. Vol. 13, no. 3. P. 291–317. URL: https://doi.org/10.1006/jvlc.2002.0236 (date of access: 15.03.2024).

10. Helson H. E. Structure Diagram Generation. Reviews in Computational Chemistry. Hoboken, NJ, USA, 2007. P. 313–398. URL: https://doi.org/10.1002/9780470125908.ch6 (date of access: 15.03.2024).

11. Bergin J. Data Structures and Algorithms. Data Structure Programming. New York, NY, 1998. P. 1–26. URL: https://doi.org/10.1007/978-1-4612-1630-8_1 (date of access: 18.03.2024).

12. Beebe N. H. F. Exponential and logarithm. The Mathematical-Function Computation Handbook. Cham, 2017. P. 267–298. URL: https://doi.org/10.1007/978-3-319-64110-2_10 (date of access: 20.03.2024).

13. Andrew C. O., Hildebrand P. E. Experimental Data Collection. Planning and Conducting Applied Agricultural Research. 2019. P. 34–46. URL: https://doi.org/10.1201/9780429301711-4 (date of access: 22.03.2024).

14. Peress M. Large-Scale Ideal Point Estimation. Political Analysis. 2021. P. 1–18. URL: https://doi.org/10.1017/pan.2021.5 (date of access: 22.03.2024).

15. Sundaramoorthy S. Uml Diagramming. Taylor & Francis Group, 2022.

16. Gomaa H. Designing software product lines with UML: From use cases to pattern-based software architectures. Boston : Addison-Wesley, 2005. 701 p.

17. Melnichuk M., Kornienko Y., Boytsova O. WEB-SERVICE. RESTFUL ARCHITECTURE. Automation of technological and business processes. 2018. Vol. 10, no. 1. URL: https://doi.org/10.15673/atbp.v10i1.876 (date of access: 26.03.2024).

18. Sharma M. MongoDB Complete Guide: Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands, MongoDB Compass, MongoDB Server, ... and MongoDB Sharding. BPB Publications, 2021. 470 p.

19. Craig Walls. Spring. Boot in Action: 6rd Edition. USA: Manning, 2022. 520 p.

20. Gutierrez F. Spring Data Within Your Spring Application. Introducing Spring Framework. Berkeley, CA, 2014. P. 203–216. URL: https://doi.org/10.1007/978-1-4302-6533-7_15 (date of access: 26.03.2024).

21. Grunitzky G. Java Programming: Java. Independently Published, 2017.

22. Sarcar V. MVC Pattern. Java Design Patterns. Berkeley, CA, 2018. P. 437–457. URL: https://doi.org/10.1007/978-1-4842-4078-6_26 (date of access: 26.03.2024).

23. di Pisa F. Spring Security. Beginning Java™ and Flex. Berkeley, CA, 2009. P. 183–194. URL: https://doi.org/10.1007/978-1-4302-2386-3_6 (date of access: 26.03.2024).

24. Java testing patterns / ed. by T. Jon. Indianapolis, Ind : Wiley, 2004. 400 p.

25. Батраченко В.О. Розробка алгоритму метода підбору та розподілу рекомендованих курсів працівникам IT-компанії. // 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6. Конференція «Інформаційні інтелектуальні системи». Харків: ХНУРЕ. 2024. С. 699-701.