

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему СИСТЕМА ІНТЕЛЕКТУАЛЬНОГО РОЗПОДІЛУ

РОБОЧИХ МІСЦЬ ДЛЯ БІРЖИ ТРУДА

THE SYSTEM OF INTELLIGENT DISTRIBUTION OF JOBS

FOR THE LABOR EXCHANGE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-130
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАВЛЕЦЬ О.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник

СЕРДЮК С.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент

ПОДКОВАЛІХІНА О.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАВЛЕЦЯ Олексія Дмитровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Система інтелектуального розподілу робочих місць для біржі труда

The System of Intelligent Distribution of Jobs for the Labor Exchange

керівник проєкту (роботи) к.т.н., доцент, СЕРДЮК Сергій Микитович,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СЕРДЮК С.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “15” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Керівництво програміста	5 тиждень	Розділ 4
7	Керівництво оператора	6 тиждень	Розділ 5
8	Оформлення пояснювальної записки та документів до неї.	7 тиждень	Додатки
9	Нормоконтроль та рецензування.	8 тиждень	
10	Захист роботи.	9 тиждень	

Студент(ка)

(підпис) Олексій ПАВЛЕЦЬ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СЕРДЮК
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
104 с., 46 рис., 9 табл., 2 дод., 18 джерел.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, БІРЖА ТРУДА, РОБОТОДАВЕЦЬ,
ПРАЦІВНИК, РИНОК ПРАЦІ.

Об'єкт дослідження – процес інтелектуального розподілу робочих місць.

Предмет дослідження – комп'ютерні програми для інтелектуального розподілу робочих місць.

Мета роботи – розробка програмного забезпечення для інтелектуального розподілу робочих місць на біржі труда з метою автоматизації процесу пошуку роботи для працівників біржи труда.

Матеріали, методи та технічні засоби: мова програмування – C# як мова програмування, середовище розробки – Microsoft Visual Studio, графічна платформа – Windows Forms.

Результати. Створено програмний Windows-застосунок який має інтелектуальну систему для покращення розподілу робочих місць.

Висновки. Розроблено систему інтелектуального розподілу робочих місць, яка автоматизує та прискорює процес пошуку роботи.

Галузь використання – біржа труда.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 104 pages, 46 figures, 9 tables, 2 appendixes, 18 sources.

INTELLIGENT SYSTEM, LABOR EXCHANGE, EMPLOYER, EMPLOYEE, LABOR MARKET.

Object of study is the process of intelligent distribution of jobs.

Subject of study are computer programs for the intelligent distribution of jobs.

The purpose of the work is to develop software for the intelligent distribution of jobs on the labor exchange in order to automate the job search process for labor exchange employees.

Materials, methods, and tools: programming language – C#, development environment – Microsoft Visual Studio, graphic platform – Windows Forms.

Results. A Windows software application has been created that has an intelligent system for improving the distribution of jobs.

Conclusions. A system of intelligent distribution of jobs has been developed, which automates and accelerates job allocation process.

The field of use is a labor exchange.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Основні поняття і визначення.....	10
1.2 Формальна постановка задачі.....	10
1.3 Огляд аналогів та порівняльний аналіз.....	13
1.4 Методика проектування.....	18
1.5 Технічне завдання.....	18
1.6 Висновки за розділом.....	22
2 Опис програми.....	23
2.1 Визначення структури системи.....	23
2.2 Розробка системи правил.....	27
2.3 Вибір засобів проектування і розробки.....	28
2.4 Висновки за розділом.....	36
3 Основні рішення щодо реалізації компонентів системи.....	37
3.1 Розробка бази даних.....	37
3.2 Опис реалізованого інтерфейсу.....	40
3.3 Опис реалізації компонентів програми.....	45
3.4 Опис використаних бібліотек.....	49
3.5 Тестування програми.....	50
3.6 Висновки за розділом.....	54
4 Керівництво програміста.....	55
4.1 Призначення та умови застосування програми.....	55
4.2 Характеристики програми.....	55
4.3 Структура програми.....	55
4.4 Вхідні та вихідні дані.....	56
4.5 Повідомлення програмісту.....	57

5 Керівництво оператора	58
5.1 Призначення програми	58
5.2 Умови виконання програми	58
5.3 Виконання програми	58
5.4 Повідомлення оператору	62
Висновки	63
Перелік джерел посилання	64
Додаток А Текст програми	66
Додаток Б Слайди презентації	97

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

VS – Microsoft Visual Studio;

WF – Windows Forms;

БЗД – база знань і даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СІРРМ – Система інтелектуального розподілу робочих місць;

ТЗ – технічне завдання.

ВСТУП

У сучасному світі розвитку технологій та постійних змін на ринку праці актуальністю набуває питання раціонального розподілу робочих місць. Одним із ключових аспектів цього процесу є розвиток систем інтелектуального розподілу робочих місць. Такі системи не лише сприятимуть ефективнішому використанню людських ресурсів, але й забезпечують оптимальний розподіл робочих місць відповідно вимог ринку та потреб суспільства.

Головна мета заходів, що здійснює біржа труда, для регулювання зайнятості населення, є стимулюванням попиту на працю та зближенні якості робітників з якістю попиту на них [1].

Мета даної дипломної роботи полягає у створенні функціонуючої системи інтелектуального розподілу робочих місць для біржі труда. Для досягнення цієї мети розглянуті такі аспекти, як технології, використовувані у таких системах, аналіз їх впливу на процеси залучення та розподілу робочих місць, а також оцінка переваг та недоліків використання інтелектуальних систем.

Дослідження даної теми має важливе значення для розвитку сучасного ринку праці та підвищення його ефективності, а також для розуміння та вдосконалення процесів розподілу робочих місць у сучасному суспільстві.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття і визначення

У сучасному суспільстві, де стрімко розвиваються інформаційні технології, концепція «Системи інтелектуального розподілу робочих місць» дуже актуальна. Ця концепція поєднує технічні та соціально-економічні аспекти з метою підвищення ефективності процесу поділу праці, дозволяє прискорити та спростити процес прийому на роботу для роботодавців та робітників.

Експертна система – це програмний засіб який використовує експертні знання певної предметної області, ставлячи мету ефективного вирішення задач які цікавлять користувача, на рівні експерта.

Біржа труда – місце, де зустрічаються попит і пропозиція на ринку праці. Мета біржи труда це забезпечення ефективної комунікації між роботодавцями та робітниками, що допомагає швидко знайти роботу та робітників, а також відслідковувати та аналізувати тенденції ринку [2].

Система інтелектуального розподілу робочих місць для біржі труда — це інноваційний інформаційно-аналітичний застосунок, який використовує передові технології у вигляді експертної системи для автоматизації відбору та розподілу робітників на біржах труда. Система буде спрямована на оптимізацію використання людських ресурсів, та забезпечення відповідності кваліфікації робітників вимогам роботодавців.

З огляду на наведене вище визначення, інтелектуальна система розподілу робочих місць для біржі труда спрямована на оптимізацію процесу розподілу праці, підвищення ефективності біржі праці та зниження рівня безробіття за рахунок впровадження сучасних інформаційних технологій.

1.2 Формальна постановка задачі

Концептуальну модель, яка описує розподіл робочих місць на біржі труда, показано на рис. 1.1



Рисунок 1.1 – Концептуальна модель розподілу робочих місць на біржі труда

На даний момент на більшості бірж труда процес розподілу робочих місць провадить кваліфікований спеціаліст, інспектор з працевлаштування. Нижче наведено вхідні та вихідні дані інспектора з працевлаштування під час розподілу робочих місць.

Вхідні дані які отримує інспектор включають:

- вакансії та особливі зауваження щодо вимог від підприємництва;
- дані про робітника, що шукає роботу, включаючи інформацію про їх навички, досвід роботи;
- оцінка результату роботи та вказівка, що саме в результаті не сподобалося або сподобалося, від роботодавців і робітників.

Вихідні дані інспектора включають:

- систематизовані дані, які були отримані, від робітників та роботодавців;

- інформація про вакансії, які відповідають робітнику;
- інформація про робітників, які відповідають вимогам вакансії.

Через інспектора з працевлаштування проходить величезна кількість даних про робітників та роботодавців, через це його швидкість роботи може бути дуже повільною. Ефективність такої роботи дуже низька, тому було вирішено розробити інтелектуальну систему, щоб збільшити якість та швидкість розподілу робочих місць. Також, завдяки інтелектуальності системи, користуватися нею зможе більшість людей, навіть без освіти у сфері управління людськими ресурсами. Нижче наведено вхідні та вихідні дані системи що розробляється.

Вхідні дані системи включають:

- знання експерта про необхідні характеристики робітників та їх значення для вимог роботодавців;
- вимоги щодо робітників від підприємництва введені в поля інтерфейсу програми користувачем;
- дані про робітника що шукає роботу, включаючи інформацію про їх навички, досвід роботи тощо введені в поля інтерфейсу програми користувачем;
- оцінка результату роботи системи та вказівка чи сподобалося або не сподобалося, від роботодавців, які її використовували її результат введені в поля інтерфейсу програми користувачем.

Вихідні дані системи включають:

- дані щодо вимог роботодавців та параметрів робітників, які були отримані та систематизовані системою;
- інформація про робітників, які відповідають вимогам вакансії та відсоток того, наскільки ці робітники задовольняють вимогам вакансії.

Бажані результати включають:

- максимізацію відповідності між вимогами роботодавців та параметрами робітників;
- мінімізацію часу, необхідного для знаходження відповідних робочих місць для робітників та роботодавців;
- оптимізацію використання людських ресурсів на біржі труда.

Критерії оцінювання якості результатів можуть включати:

- відповідність пропозиції роботи до профілю робітника;
- рівень задоволення роботодавця від взаємодії.

Обмеження, які потрібно врахувати, можуть включати: обмеження на кількість робітників, які можуть бути запропоновані певному роботодавцю.

Додавання інтелектуальної системи для процесу розподілу робочих місць має багато переваг. Автоматизація процесу, яка передбачає використання експертних знань та зворотного зв'язку, дозволить значно зменшити навантаження на інспекторів з працевлаштування, підвищити швидкість обробки даних та прийняття рішень. Завдяки систематизації та аналізу отриманих даних, система зможе забезпечити більш точне відповідність між вимогами роботодавців та параметрами робітників, що збільшить рівень задоволення. Також завдяки простоті системи, біржа труда зможе використовувати більше робітників для процесу розподілу робочих місць. Таким чином, розробка інтелектуальної системи є доцільним і перспективним кроком, який забезпечить підвищення якості та ефективності процесу працевлаштування.

1.3 Огляд аналогів та порівняльний аналіз

В даний момент система для розподілу робочих місць, як ніколи, затребувана, оскільки популяція людей на планеті Земля, і відповідно кількість людей, що шукає роботу, безперервно зростає. Корпорації, що розвиваються, потребують нових кадрів, випускники шкіл та інститутів, що шукають роботу, всі вони потребують програмне забезпечення що допоможе втілити ці потреби швидко та задовільно. Щоб створити програмне забезпечення, що буде користуватися попитом, необхідно розглянути вже існуючі аналоги.

На сучасному ринку існує безліч систем і платформ для підбору та працевлаштування, кожна з яких має свої особливості та переваги.

LinkedIn Talent Solutions є однією з провідних платформ пошуку роботи та працевлаштування. Роботодавці можуть публікувати вакансії, а кандидати можуть створювати профілі та надсилати резюме.

Переваги: зручний інтерфейс, велика база даних кандидатів, можливість налаштування різноманітних параметрів пошуку, інтеграція з іншими платформами LinkedIn.

Недоліки: вартість використання платних планів висока, а функціональність для безкоштовних користувачів обмежена [3].

Інтерфейс показано на рис. 1.2.

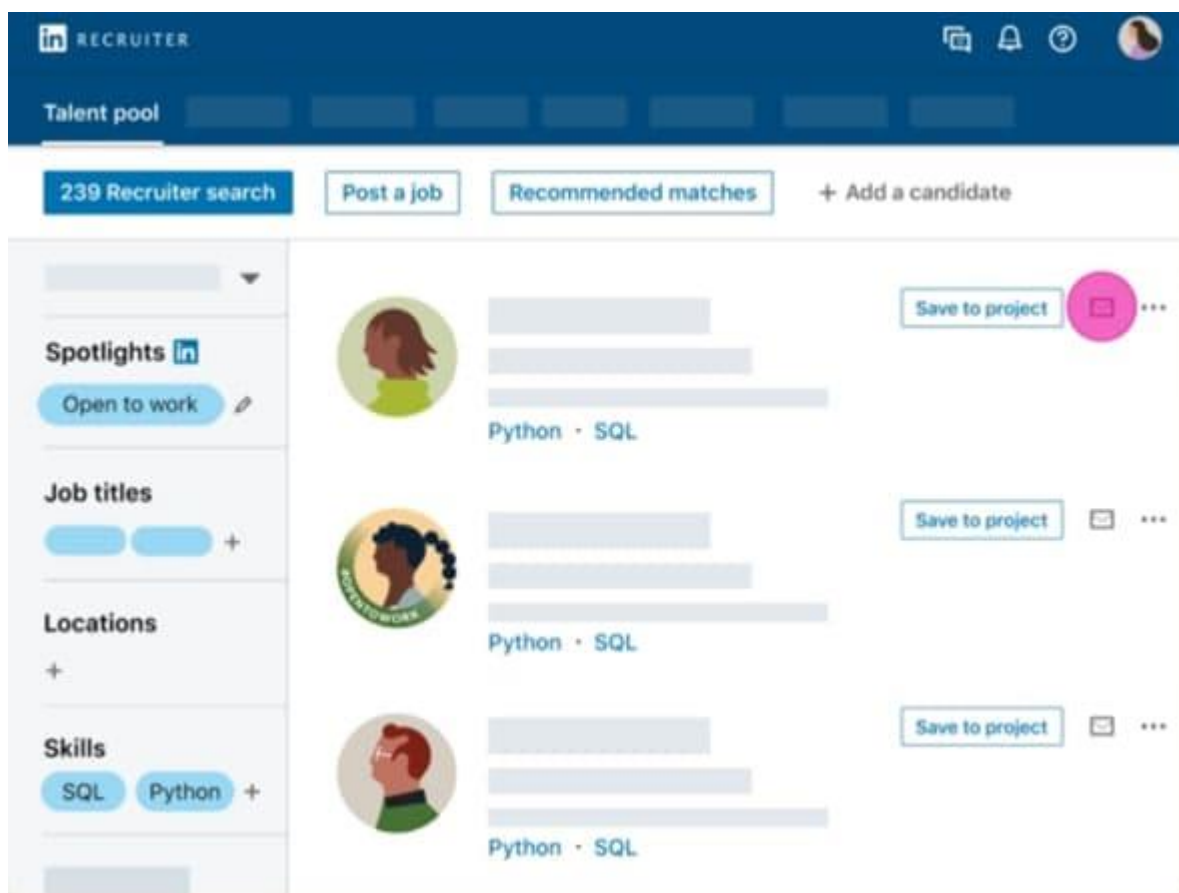


Рисунок 1.2 – Інтерфейс LinkedIn Talent Solutions

Greenhouse – це платформа для управління підбором персоналу, яка допомагає роботодавцям керувати всім процесом найму, включаючи створення робочих місць, відбір кандидатів і співбесіди.

Переваги: інтеграція з різними інструментами зв'язку та аналізу.

Недоліки: складний у використанні інтерфейс, висока вартість і обмежені можливості для малого бізнесу [4].

Інтерфейс показано на рис. 1.3.

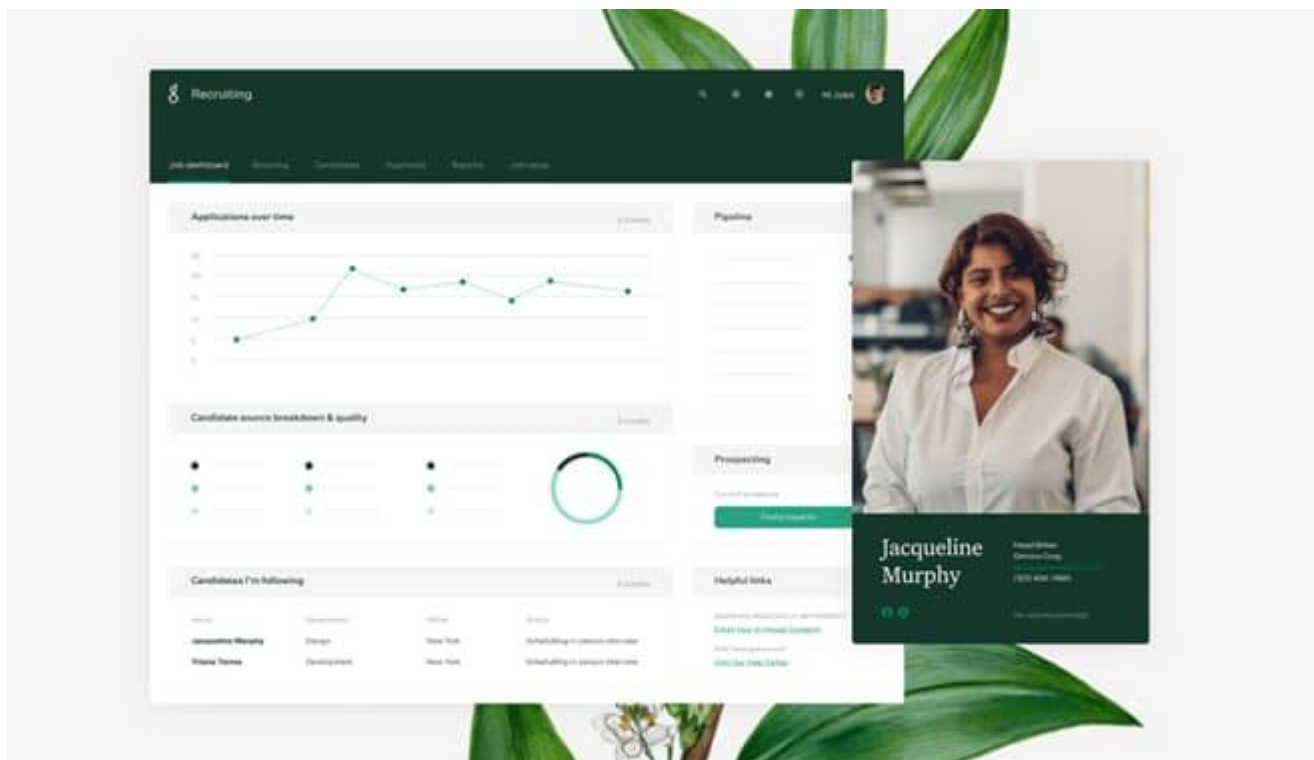


Рисунок 1.3 – Інтерфейс Greenhouse

SAP SuccessFactors – це комплексне рішення для управління людськими ресурсами, яке включає модулі для найму та відбору персоналу.

Переваги: широкий набір функцій, можливість інтеграції з іншими рішеннями SAP, розширена аналітична звітність.

Недоліки: високі витрати на впровадження та підтримку, складний процес налаштування, незручний інтерфейс, довгий відгук системи [5].

Інтерфейс показано на рис. 1.4.

Job Title	Req ID	Hiring Manager	Department	Location	Candidates	Progress	Requisition Status	Job Start Date
Sales Manager	1312	Vic Stokes ES	Sales (SALES)	San Mateo (US, SFO)	4 (1 New)	████████████████████	Offer Pending	06/15/2014
Sales Director	1403	Carla Grant ES	Sales (SALES)	San Mateo (US, SFO)	-	████████████████████	Offer Pending	11/30/2014
Employee Relations Manager	1380	Linda Lewis ES	Talent Management	Boston, MA	4	████████████████████	Offer Pending	04/30/2014
Solutions Consultant, Portland	1373	Wilma Soren ES	Sales	Portland, OR	-	████████████████████	Sourcing	04/30/2014
Compensation Analyst	1372	Linda Lewis ES	Talent Management	San Mateo, CA	-	████████████████████	Sourcing	04/30/2014
Financial Analyst	1371	Janet James ES	Finance	San Mateo, CA	-	████████████████████	Sourcing	04/30/2014
Cost Accountant	1370	Janet James ES	Finance	San Mateo, CA	2	████████████████████	Phone Screening	04/14/2014
Payroll Supervisor	1369	Janet James ES	Finance	San Mateo, CA	3	████████████████████	Phone Screening	04/30/2014
HR Generalist	1368	Linda Lewis ES	Talent Management	Denver, CO	6	████████████████████	Interviewing	04/14/2014
Help Desk Manager	1367	Jennifer Harley ES	Technology	Boston, MA	1	████████████████████	Sourcing	04/11/2014
Database Programmer	1366	Jennifer Harley ES	Technology	Boston, MA	4 (1 New)	████████████████████	Interviewing	04/25/2014
IT Project Manager	1365	Jennifer Harley ES	Technology	Denver, CO	5	████████████████████	Interviewing	05/11/2014
Solutions Consultant	1364	Wilma Soren ES	Sales	Portland, OR	-	████████████████████	Sourcing	11/04/2014
Account Manager	1363	Stet Morton ES	Sales (SALES)	Arlington, Virginia (US, DCM)	-	████████████████████	Sourcing	04/29/2014

Рисунок 1.4 – Інтерфейс SAP SuccessFactors

У табл. 1.1 порівняємо попередні аналоги із застосунком, що розробляється. Серед критеріїв виділимо наступні:

- інтелектуальність системи: забезпечення можливості автоматичного підбору вакансій для робітників за характеристиками що дозволяє зменшити час, необхідний для пошуку роботи, і підвищити ефективність системи завдяки використанню експертних даних;

- висока швидкість роботи: швидкий відгук системи є важливим, оскільки це підвищує задоволеність користувачів;

- низька ціна: прийнятно низька ціна введення програми в роботу є ключовою для забезпечення доступності системи для різних організацій;

- зручність та сучасність: інтуїтивно зрозумілий та зручний інтерфейс сприяє покращенню користувацького досвіду і підвищує сприйняття системи користувачами;

- можливість інтеграції з іншими платформами: можливість інтеграції з іншими системами дозволить забезпечити більше функціональності системи;

– надання можливості спілкування між роботодавцем та робітником: забезпечення можливості спілкування в межах системи дозволить забезпечити ефективний обмін інформацією між учасниками процесу працевлаштування.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій порівняння	LinkedIn Talent Solutions	Greenhouse	SAP SuccessFactors	Мій застосунок
Інтелектуальність системи	-	-	-	+
Висока швидкість роботи	+	+	-	+
Низька ціна	-	-	-	+
Зручність та сучасність	+	+	-	+
Можливість інтеграції з іншими платформами	+	+	+	-
Надання спілкування між роботодавцем та робітником	+	+	-	-

За результатами порівняння можна побачити, що аналоги пропонують різні переваги, але програми-конкуренти занадто дорогі і мають недоліки. Саме тому актуальною залишається задача розробки нового застосунку, який буде мати високу швидкість роботи, сучасний дизайн, а головне – мати інтелектуальну систему та бути недорогим. Його простота, ефективність і доступність мають зацікавити біржу труда.

1.4 Методика проектування

Для розробки системи інтелектуального розподілу робочих місць була використана методика проектування, яка включає наступні етапи:

- аналіз вимог та потреб ринку праці: вивчення потреб роботодавців та робітників, визначення основних вимог та очікувань щодо системи розподілу робочих місць;
- огляд існуючих рішень: проведення аналізу існуючих систем розподілу робочих місць для виявлення недоліків та можливостей для покращення;
- визначення функціональних та технічних вимог;
- проектування та розробка архітектури системи;
- розробка та тестування: реалізація системи на основі розробленої архітектури та проведення тестування для перевірки її працездатності та відповідності вимогам.

Ця методика дозволе забезпечити якісний та ефективний процес розробки системи інтелектуального розподілу робочих місць, відповідно до вимог та потреб ринку праці.

1.5 Технічне завдання

1.5.1 Призначення та область застосування програми

Метою є створення програмного забезпечення, яке представляє собою систему інтелектуального розподілу робочих місць для біржі труда. Основні завдання цієї програми полягають у спрощенні процесу розподілення робочих місць для працівників біржи труда, покращення його якості та ефективності.

Основна область застосування програми це біржа труда. Завдяки простоті та інтелектуальності системи користуватися нею зможе будь-який робітник біржи труда. Таким чином, програма має широкий спектр застосування і може бути корисною для усіх учасників ринку праці, сприяючи ефективнішому розподілу

робочих місць та забезпеченню відповідності між вимогами роботодавців та параметрами робітників.

1.5.2 Підстави для розробки

У сучасних умовах на багатьох біржах труда виникає нагальна потреба в удосконаленні процесів розподілу робочих місць. Існуючі методи розподілу, що базуються на ручному аналізі даних інспекторами з працевлаштування, виявляються неефективними через такі фактори: великий обсяг даних, суб'єктивний підхід, обмежена швидкість обробки.

Ці проблеми вимагають розробки інтелектуальної системи розподілу робочих місць, яка б автоматизувала процеси аналізу та прийняття рішень, підвищила їх ефективність та забезпечила об'єктивність і точність у відповідності вимог роботодавців та параметрів робітників.

Розробка виконується у відповідності з завданням, яке було визначено на кваліфікаційну роботу бакалавра за темою «Система інтелектуального розподілу робочих місць для біржі труда», що затверджена наказом № 168 від 24 квітня 2024 р. по Національному університету «Запорізька політехніка» [6].

1.5.3 Вимоги до функціональних характеристик

Система має реалізовувати наступні функціональні характеристики:

- система має надавати можливість виконання операцій CRUD (Create, Read, Update, Delete) під час роботи з базою даних вимог роботодавця та даних про робітника;
- система, при запиті, повинна показувати, які робітники найбільше відповідають вимогам певної вакансії, а також відсоткову оцінку ступеня відповідності, яка показує шанс того, що роботодавець буде задоволений результатом найму, запропонованим системою;

- система повинна надавати можливість створення відгуків на попередні результати її оцінки ступеня відповідності, який показував шанс того, що роботодавець буде задоволений результатом найму, запропонованим системою, із зазначенням, сподобався чи не сподобався результат вибору системи роботодавцю. Система має зберігати ці відгуки;
- система повинна змінювати оцінку ступеня відповідності, яка показує, який шанс того, що роботодавець буде задоволений результатом найму, запропонованим системою, залежно від залишених відгуків для адаптації та покращення результату своєї роботи, з метою максимального задоволення робітників та роботодавців результатами її оцінки;
- система має вміти працювати навіть із недостатньою інформацією у вимогах роботодавця. Система повинна змінювати правила заповнення недостатньої інформації в залежності від залишених відгуків [7].

1.5.4 Вимоги до надійності

Вимоги до надійності включають наступні компоненти:

- алгоритми, що використовуються для підбору вакансій та кандидатів, повинні бути перевірені на точність і стабільність вихідних результатів;
- регулярне тестування та оновлення знань та алгоритмів для забезпечення їх актуальності та відповідності сучасним вимогам ринку праці.

1.5.5 Умови експлуатації

Система повинна бути доступна для користування з будь-якого пристрою в межах локальної мережі. Ефективна робота системи передбачає її стабільну роботу у будь-який час доби без перебоїв та надійну обробку даних користувачів.

1.5.6 Вимоги до складу та параметрів технічних засобів

ПК користувача для роботи з системою має відповідати таким мінімальним вимогам:

- ОС: Windows 7;
- процесор: 4 ядра з тактовою частотою 2,5 ГГц;
- оперативна пам'ять: 4 ГБ;
- відеокарта з підтримкою DirectX 10;
- вільний дисковий простір: 1 ГБ.

1.5.7 Необхідні стадії і терміни розробки

Розробка програми для системи інтелектуального розподілу робочих місць включає наступні стадії та терміни:

- аналіз вимог: визначення вимог до програми. Термін: 1 тиждень;
- проектування системи: розробка архітектури та вибір інструментів розробки. Термін: 1 тиждень;
- реалізація та програмування: написання коду для програми та реалізація визначених вимог. Термін: 2 тижні;
- тестування: проведення випробувань для виявлення та усунення помилок. Термін: 1 тиждень.

1.5.8 Види випробувань

Для забезпечення якості та надійності необхідно провести наступні види випробувань:

- модульні тести: перевірка окремих модулів програми на коректність роботи;
- функціональні тести: перевірка відповідності функціональних вимог програми.

1.6 Висновки за розділом

У даному розділі був проведений теоретичний аналіз предметної області та опис основних вимог для програмного забезпечення «Системи інтелектуального розподілу робочих місць», що розробляється. Було наведено основні поняття, визначення та зробив формальне визначення завдання розв'язуваної проблеми. У цьому розділі було розглянуто переваги та недоліки існуючих аналогів та зроблено їх порівняння. Також було описано технічне завдання, де позначено призначення та сферу застосування програми, позначено вимоги до програми та терміни розробки.

З огляду на проведену роботу, інтелектуальна система розподілу робочих місць для біржі труда є важливим і перспективним завданням спрямованим на оптимізацію процесу розподілу праці, підвищення ефективності біржі труда та зниження рівня безробіття за рахунок впровадження сучасних інформаційних технологій.

2 ОПИС ПРОГРАМИ

2.1 Визначення структури системи

Типовий сценарій роботи користувача з програмою може бути описаний за допомогою прецеденту «Розподіл робочих місць»:

- основний виконавець – відповідальний за розподіл робочих місць;
- передумови: програма відкрита.

Післяумова: розподілено робочі місця.

Основний успішний сценарій наступний.

Крок 1. Виконавець переглядає дані про вакансії роботодавців.

Крок 1.1. Виконавець отримує нові дані про вакансії від роботодавців.

Крок 1.2. Виконавець додає, змінює, видаляє дані про вакансії.

Крок 2. Виконавець переглядає дані про робітників.

Крок 2.1. Виконавець отримує нові дані від робітників.

Крок 2.2. Виконавець додає, змінює, видаляє дані про робітників.

Крок 3. Виконавець робить запит на розподіл місць для обраної вакансії.

Крок 4. Виконавець отримує список із топ-10 робітників відповідних вакансії та обирає одного з них, після чого повідомляє робітника про місце його розподілу та роботодавця про розподіленого робітника.

Крок 5. Виконавець отримує відгук від роботодавця та вводить його до програми.

Use-case діаграму, що ілюструє даний прецедент наведено на рис. 2.1.

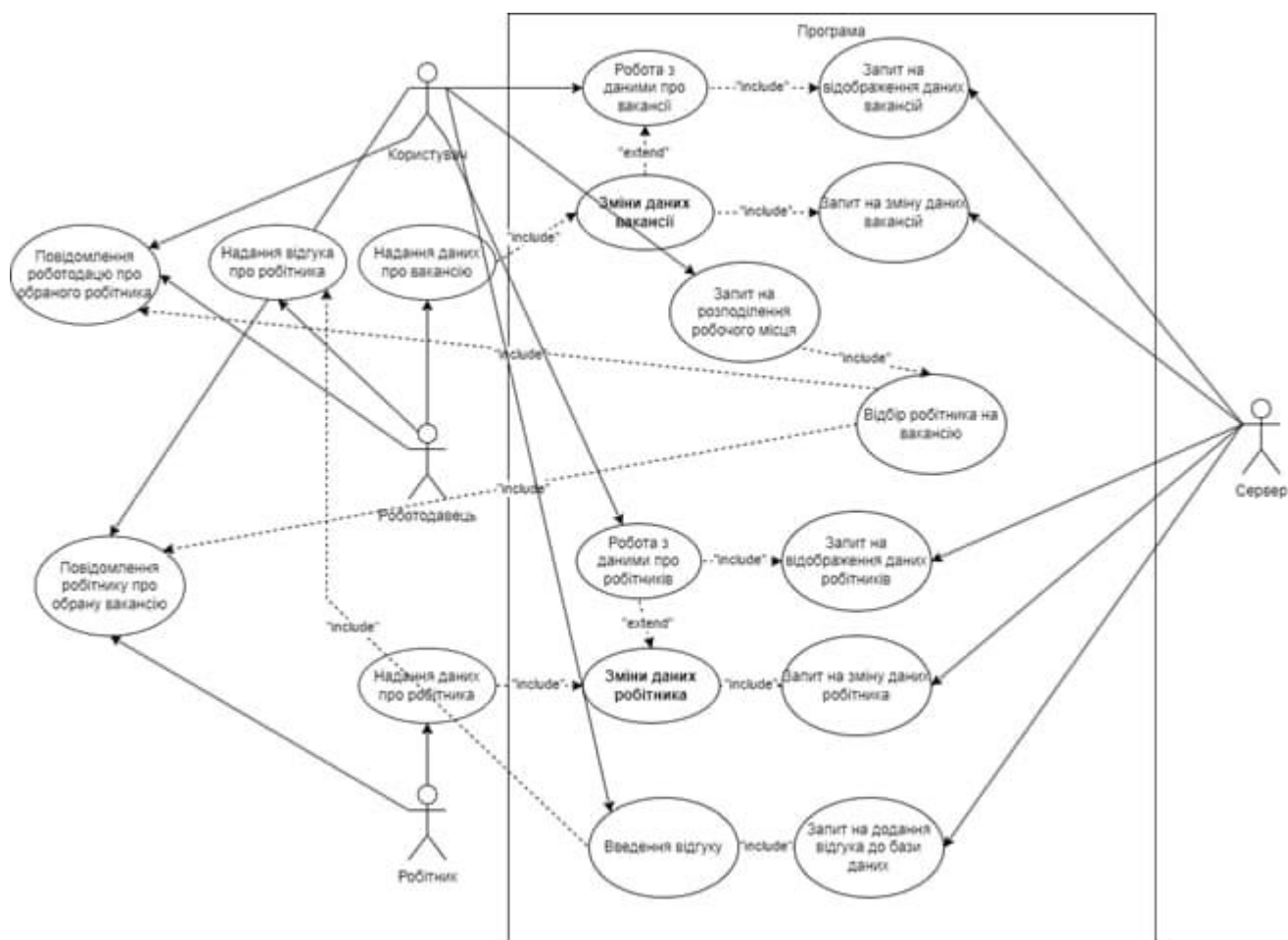


Рисунок 2.1 – Use-case діаграма «Розподіл робочих місць»

Програма складатиметься з наступних основних компонентів: вікна інтерфейсу користувача (форма роботи з даними вакансій, форма роботи з даними робітників, форма роботи із запитом на розподіл робочих місць, форма роботи з логами розподілених робітників, форма відображення відповідних робітників, форма складання відгуку), модуль інтерпретації знань, модуль роботи з базою даних та база даних. Нижче наведено опис цих компонентів:

- форма роботи з даними вакансій – це інтерфейс для взаємодії з усіма даними про вакансії. Основними функціями для користувача на цій формі є додавання, зміна або видалення наявних записів про вакансії або просто перегляд цих даних. З цією формою взаємодіє база даних, при додаванні або зміні записів форма взаємодіє з модулем інтерпретації знань для категоризації даних задля потреб системи;

- форма роботи з даними робітників – це інтерфейс для взаємодії з усіма даними про робітників. Основними функціями для користувача на цій формі є додавання, зміна або видалення наявних записів про робітників або просто перегляд цих даних. З цією формою взаємодіє база даних, при додаванні або зміні записів форма взаємодіє з модулем інтерпретації знань для категоризації даних задля потреб системи;

- форма роботи із запитом на розподіл робочих місць – це інтерфейс для взаємодії зі списком вакансій, які потребують працівників. Основними функціями є пошук потрібної вакансії за назвою компанії та вибір вакансії для якої відбуватиметься розподіл робочих місць. З цією формою взаємодіє база даних та відображення відповідних робітників;

- форма роботи з логами розподілених робітників – це інтерфейс для взаємодії зі списком логів розподілених робочих місць. Основними функціями форми є перегляд логів та вибір запису про розподіл для відкриття форми для складання відгуку про розподіл із цього запису. З цією формою взаємодіє база даних та форма складання відгуку, при додаванні або зміні записів форма взаємодіє з модулем інтерпретації знань для категоризації даних задля потреб системи;

- форма відображення відповідних робітників – це інтерфейс для взаємодії зі списком оптимальних робітників для обраної на формі роботи із запитом на розподіл робочих місць вакансії. Основними функціями форми є перегляд оптимальних робітників та вибір одного для відправки у компанію. З цією формою взаємодіє база даних, задля створення списку оптимальних робітників форма взаємодіє з модулем інтерпретації знань;

- форма складання відгуку – для складання відкликання розподіленого працівника від роботодавця. Основними функціями форми є вибір подобається чи не подобається роботодавцю результат роботи системи. З цією формою взаємодіє база даних, задля зміни знань системи форма взаємодіє з модулем інтерпретації знань;

- модуль інтерпретації знань відповідає за завантаження знань у систему та їх використання під час роботи з даними про вакансії та робітників. Інтерпретатор дозволяє категоризувати вакансії та робітників, шукати оптимальних для вакансій робітників завдяки правилам у ньому, а також оновлювати знання системи;
- модуль роботи з базою даних відповідає за додавання, оновлення або видалення інформації з бази даних;
- база даних зберігає в собі дані про вакансії, працівників, логи розподілу та знання для їх ефективного використання.

Для отримання кращого розуміння взаємодії між компонентами програми, на рис. 2.2 наведено структурну схему програми. Вона відображає основні елементи системи, їх функціональні зв'язки та взаємодію між собою: форма роботи з даними вакансій та форма роботи з даними робітників підключаються до модулю роботи з базою даних для отримання, додавання, зміни або видалення записів, а також взаємодіють з модулем інтерпретації знань для категоризації даних; форма роботи із запитом на розподіл робочих місць отримує дані про вакансії та відображає оптимальних робітників за допомогою модуля інтерпретації знань; форма роботи з логами розподілених робітників взаємодіє з базою даних для отримання логів та відкриття форми складання відгуку; форма відображення відповідних робітників використовує дані з бази даних і модуль інтерпретації знань для створення списку оптимальних робітників; форма складання відгуку з'єднується з базою даних для зберігання відгуків роботодавців та взаємодіє з модулем інтерпретації знань для оновлення знань системи.

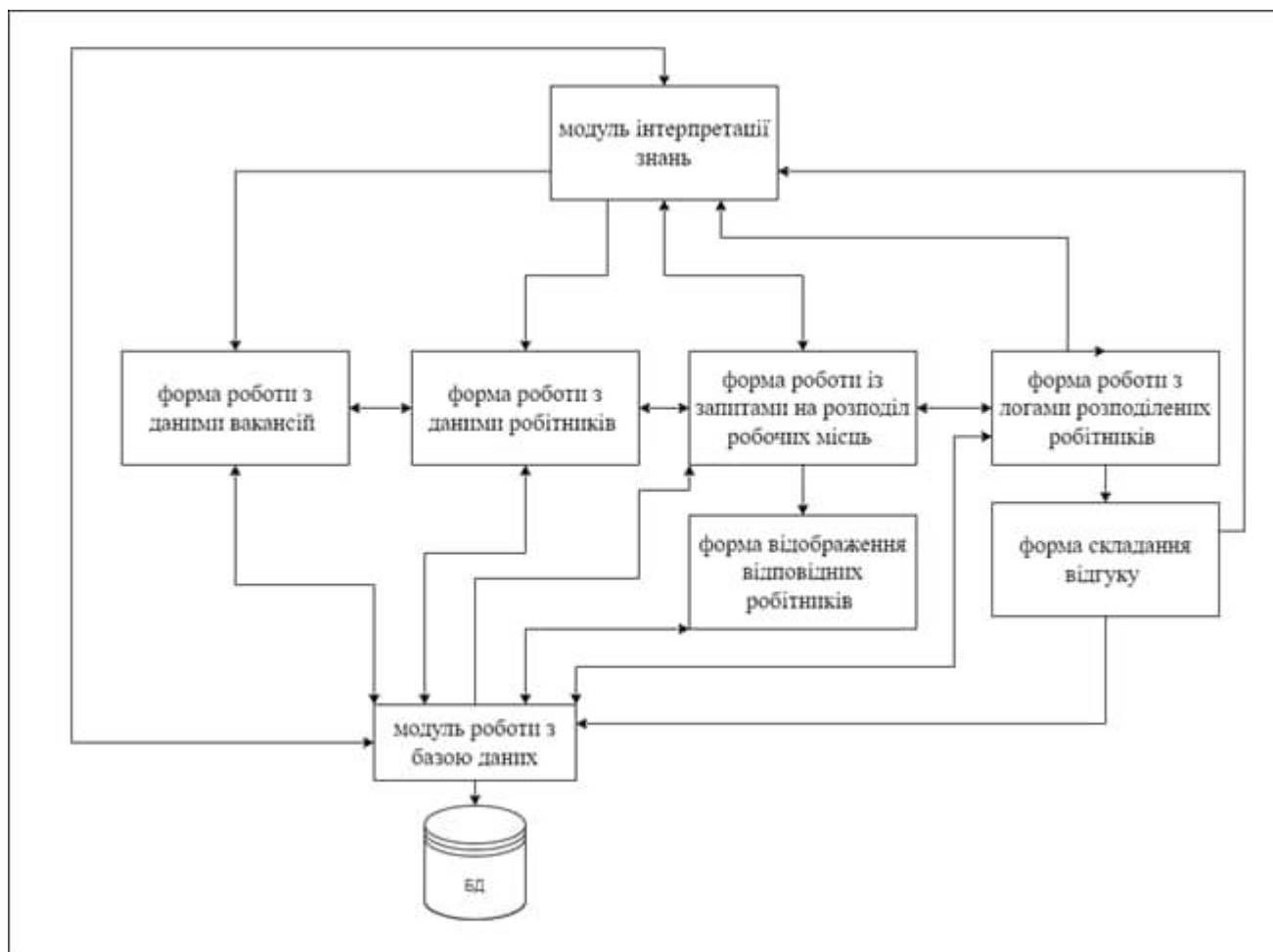


Рисунок 2.2 – Структурна схема програми

2.2 Розробка системи правил

Продукційне правило складається з частини **ЯКЩО** (умова або передумова) і частини **ТО** (дія або висновок). Продукційні правила, відіграють ключову роль у процесі автоматизації підбору працівників на вакансії. У БЗД зберігається наступна інформація:

- терміни, що використовуються для опису професійних якостей працівників;
- навички, що відповідають кожному з цих термінів;
- вимоги до досвіду роботи для деяких термінів, особливо для посад, що потребують певного досвіду;
- категорії діяльності, яким відповідають терміни;

- лічильники позитивних та негативних відгуків щодо визначеного досвіду роботи для цих термінів;
- дата оновлення цих термінів.

Продукційні правила в системі формалізують логіку ухвалення. Нижче наведено декілька прикладів правил системи.

П1. ЯКЩО «вакансія = містить Java junior» І «вакансія = НЕ містить вимог до досвіду роботи» ТО «вимоги до навичок = додати Java Basics, OOP Principles, Data Structures» І «вимоги до досвіду роботи = 2 роки».

П2. ЯКЩО «вакансія = містить Frontend developer» І «вакансія = НЕ містить вимог до досвіду роботи» ТО «вимоги до навичок = додати HTML, CSS, JavaScript» І «вимоги до досвіду роботи = 3 роки».

П3. ЯКЩО «вакансія = містить Python developer» І «вакансія = містить вимоги до досвіду роботи» ТО «вимоги до навичок = додати Python, Flask, SQL».

П10. ЯКЩО «негативних відгуків на термін General Practitioner на 5 більше ніж позитивних відгуків» І «оновлення терміну General Practitioner < 2 роки тому» ТО «вимоги до досвіду роботи General Practitioner = додати 1 рік» І «оновлення терміну General Practitioner = сьогодні».

П11. ЯКЩО «негативних відгуків на термін Nurse на 5 більше ніж позитивних відгуків» І «оновлення терміну General Practitioner >= 2 роки тому» ТО «відгуків на термін Nurse = 0» І «оновлення терміну Nurse = сьогодні».

Ці правила дозволять системі ефективно підбирати працівників на основі аналізу наявних даних про вакансії та кандидатів, а також підтримувати актуальність знань у базі даних. Продукційні правила є основним інструментом прийняття рішень в системі інтелектуального розподілу робочих місць.

2.3 Вибір засобів проєктування і розробки

2.3.1 Вибір середовища розробки

Для вибору найбільш підходящого середовища розробки для реалізації проєкту потрібно розглянути доступні популярні середовища: Microsoft Visual

Studio, JetBrains IntelliJ IDEA, Eclipse та NetBeans. Ці середовища мають свої переваги та недоліки, які будуть проаналізовані нижче.

Переваги Microsoft Visual Studio:

- інтеграція .NET: Visual Studio забезпечує інтеграцію з платформою .NET, що значно полегшує розробку на C#;
- функціональність: містить багато інструментів для профілювання, тестування, налагодження та аналізу коду;
- розширюваність: підтримує багато плагінів, що дозволяє адаптуватись до конкретних потреб розробника;
- зручний інтерфейс: інтуїтивно зрозумілий інтерфейс;
- вартість: наявність безкоштовної версії з нелімітованим строком користування.

Недоліки: вимоги до ресурсів: VS споживає багато системних ресурсів, що уповільнює роботу на слабших ПК [8].

Переваги JetBrains IntelliJ IDEA:

- спеціалізація на Java: краща підтримка Java;
- інтелектуальне середовище: розширені функції автозаповнення, рефакторингу та аналізу коду;
- інтеграція з VCS: підтримка систем контролю версій;
- розширюваність: підтримує багато плагінів, що дозволяє адаптуватись до конкретних потреб розробника.

Недоліки: вартість: хоча існує безкоштовна версія з обмеженою функціональністю, вона діє лише 30 днів [9].

Переваги Eclipse:

- вартість: безкоштовна;
- багатомовна підтримка: підтримує багато мов програмування за допомогою різних доступних плагінів;
- інтеграція з VCS: підтримка систем контролю версій.

Недоліки:

- складність використання: вимагає значного часу для встановлення та налаштування;
- вимоги до ресурсів: Eclipse споживає багато системних ресурсів, що уповільнює роботу на слабших ПК [10].

Переваги NetBeans:

- вартість: безкоштовна;
- підтримка багатьох мов: підтримує багато мов програмування за допомогою різних доступних плагінів;
- інтеграція з VCS: підтримує систем контролю версій.

Недоліки:

- підтримка .NET: дуже обмежена підтримка розробки .NET.;
- інтерфейс: не сучасний і не інтуїтивно зрозумілий інтерфейс [11].

Проаналізувавши переваги та недоліки доступних середовищ розробки я зупинився на двох варіантах: Microsoft Visual Studio та JetBrains IntelliJ IDEA. Щоб зробити остаточний вибір були візуалізовані недоліки та переваги у вигляді табл. 2.1.

Таблиця 2.1 – Порівняння існуючих середовищ розробки

Критерій	Microsoft Visual Studio	JetBrains IntelliJ IDEA
Підтримка .NET	Відмінна	Обмежена
Підтримка Java	Обмежена	Відмінна
Інструменти налагодження	Дуже хороші	Дуже хороші
Інтерфейс	Інтуїтивно зрозумілий	Інтуїтивно зрозумілий
Розширюваність	Дуже висока	Дуже висока
Вартість	Безкоштовно	Платно

Враховуючи специфіку проекту, використання Visual Studio буде найкращим вибором. VS пропонує потужні інструменти для розробки, налагодження та тестування додатків на платформі .NET, інтуїтивно зрозумілий інтерфейс, можливість налаштування під конкретні потреби проекту, а головне

можливість безкоштовного використання роблять його ідеальним для реалізації складних систем. Незважаючи на високі вимоги до ресурсів ПК, переваги, які надає VS, значно переважають недоліки.

2.3.2 Вибір мови програмування

При виборі мови програмування будуть розглянуті наступні мови програмування: C#, Java та Python. Кожна з цих мов має свої переваги та недоліки, які будуть наведені нижче.

Переваги C#:

- широке використання: C# підходить для розробки широкого спектру додатків;
- інтеграція з платформою .NET: C# має глибоку інтеграцію з платформою .NET, що означає доступ до потужних функцій та бібліотек;
- безпека: C# має внутрішню систему типізації, яка допомагає уникнути багатьох типових помилок на етапі компіляції;
- легкість навчання: синтаксис C# досить зрозумілий та схожий на інші мови з сімейства C, що полегшує використання новачками.

Недоліки:

- обмежена крос-платформенність: Хоча .NET Core робить C# більш крос-платформеним, деякі функції можуть бути обмеженими на платформах, що не підтримують .NET;
- вимоги до ресурсів: деякі версії .NET можуть вимагати значних ресурсів ПК для виконання додатків [12].

Переваги Java:

- крос-платформенність: Java є однією з найбільш крос-платформених мов програмування, що дозволяє розробляти додатки, які працюють на різних ОС;
- велика спільнота: Java має велику та активну спільноту розробників, яка надає безліч інструментів та бібліотек для розробки додатків;

- безпека: Java має внутрішню систему безпеки та управління пам'яттю, що робить додатки стійкими до атак.

Недоліки:

- швидкодія: Java є менш ефективною порівняно з мовами, що компілюються в нативний код;
- високі вимоги до пам'яті: додатки Java вимагають більше пам'яті для ефективної роботи [13].

Переваги Python:

- простота використання: Python дозволяє розробникам швидко створювати програми;
- багатофункціональність: Python має велику кількість бібліотек та фреймворків для різних цілей, включаючи веб-розробку, обробку даних або машинне навчання.

Недоліки:

- швидкодія: Python є менш ефективним порівняно з мовами, що компілюються в нативний код;
- безпека: Python має не дуже хорошу безпеку через свою динамічну природу [14].

Проаналізувавши мови програмування, були візуалізовані недоліки та переваги у вигляді табл. 2.2, щоб зробити остаточний вибір.

Таблиця 2.2 – Порівняння мов програмування

Критерій	C#	Java	Python
Широке використання	Так	Так	Так
Інтеграція з .NET	Висока	Немає	Немає
Крос-платформенність	Обмежена	Так	Так
Простота вивчення	Середня	Висока	Висока
Безпека	Висока	Висока	Низька
Швидкодія	Висока	Середня	Низька

Мова програмування C# виявилася найкращим вибором. C# має багато переваг, включаючи широку підтримку, ефективність виконання та безпеку, що робить його ідеальним вибором.

2.3.3 Вибір інтерфейсу програмування

При виборі інтерфейсу програмування для C# будуть розглянуті такі варіанти: Windows Forms, Windows Presentation Foundation та Universal Windows Platform. Кожен з них має свої переваги та недоліки, які будуть розглянуті нижче.

Переваги Windows Forms:

- простота використання: Windows Forms має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати без глибоких знань;
- широка підтримка: Windows Forms має широку підтримку від Microsoft, включаючи оновлення та допоміжні ресурси.

Недоліки:

- менша гнучкість: порівняно з іншими інтерфейсами, Windows Forms може бути менш гнучким у створенні додатків;
- обмежена можливість стилізації: Windows Forms має обмежені можливості щодо стилізації інтерфейсу [15].

Переваги Windows Presentation Foundation:

- модульність та роздільність: дозволяє розділити інтерфейс та логіку додатку, що полегшує підтримку та розвиток;
- гнучкість в оформленні: надає великі можливості стилізації інтерфейсу, це дозволяє створювати привабливі додатки;
- підтримка мультимедіа: має внутрішню підтримку мультимедіа, що робить його ідеальним вибором для створення додатків з мультимедійними елементами.

Недоліки:

- складність використання: може вимагати більшого часу та зусиль для вивчення;

- вимоги до ресурсів: створення складних вимагає більше ресурсів системи [16].

Переваги Universal Windows Platform:

- універсальність: дозволяє створювати додатки, які працюють на різних пристроях та платформах, включаючи ПК, планшети або мобільні пристрої;
- інтеграція зі службами Windows: має внутрішню підтримку служб та функцій Windows, що дозволяє створювати додатки з глибокою інтеграцією з цією ОС.

Недоліки:

- обмежена підтримка платформ: може є обмеженою, бо деякі функції можуть бути доступні тільки для платформ Windows 10 або певних пристроїв, що обмежує універсальність додатка;
- вимоги до підтримуваних пристроїв: деякі функції не підтримуються на старіших версіях Windows або на певних пристроях, що може обмежити аудиторію користувачів [17].

Проаналізувавши графічні інтерфейси остаточною вибором було зроблено Windows Forms. WF надає швидкий та простий спосіб створення інтерфейсів користувача для додатків, що дозволяє швидко розробляти та впроваджувати функціональність.

2.3.4 Вибір системи управління базою даних

При виборі системи управління базами даних будуть розглянуті такі варіанти: SQL Server, MySQL та PostgreSQL. Кожна з цих систем управління базами даних має свої переваги та недоліки, які будуть розглянуті нижче.

Переваги SQL Server:

- широкі можливості: має широкий спектр функцій та інструментів для управління даними та розробки додатків;

- інтеграція з іншими продуктами Microsoft: інтеграція з продуктами Microsoft дозволяє спростити розробку у середовищах Windows;
- підтримка бізнес-рішень: призначений для підприємств будь-якого масштабу, надаючи інструменти для обробки та аналізу даних.

Недоліки:

- високі витрати на ліцензії: SQL Server є комерційним продуктом, що може призвести до значних витрат;
- обмежена підтримка для відкритих ОС: основна підтримка спрямована а на Windows, хоча існують обмежені версії для Linux.

Переваги MySQL:

- відкритий вихідний код: є вільно розповсюджуваною, що робить її доступною для широкого кола користувачів;
- широка підтримка мов програмування: підтримує багато мов програмування та платформ розробки;
- висока продуктивність: відомий своєю високою продуктивністю та масштабованістю, особливо при роботі з великими обсягами даних.

Недоліки: є обмеженою в деяких функціях та можливостях.

Переваги PostgreSQL:

- гнучка архітектура: пропонує гнучку архітектуру та обширний набір функцій для розробників;
- безкоштовний вихідний код: розповсюджується з відкритою ліцензією, що робить його доступним для багатьох користувачів;
- широкі можливості аналізу даних: підтримує складні запити та аналітичні функції.

Недоліки:

- складність використання: є складним для новачків через свою потужність та гнучкість;
- менше інтеграцій з продуктами Microsoft: інтеграції є обмеженими.

Проаналізувавши системи управління базами даних, були візуалізовані недоліки та переваги у вигляді табл. 2.3, щоб зробити остаточний вибір.

Таблиця 2.3 – Порівняння систем управління базами даних

Характеристика	SQL Server	MySQL	PostgreSQL
Ліцензування	Платне	Безкоштовне	Безкоштовне
Підтримка ОС	Windows, Linux	Windows, Linux, macOS	Windows, Linux, macOS
Інтеграція	З продуктами Microsoft	Обширна	Обмежена
Продуктивність	Висока	Висока	Висока
Функціональність	Обширна	Середня	Обширна
Гнучкість	Середня	Висока	Висока
Спільнота користувачів	Велика	Велика	Велика
Підтримка JSON	Так	Так	Так

Система управління базою даних SQLServer являється найкращим вибором. Найбільш оптимальним варіантом є використання SQL Server, оскільки він надає широкі можливості для зберігання, обробки та аналізу даних, а також легко інтегрується з іншими продуктами Microsoft, що спростить розробку та розгортання системи у середовищі Visual Studio та Windows Forms.

2.4 Висновки за розділом

У другому розділі була визначена структура системи, що розробляється. Було проведено аналіз середовищ розробки та обрано Visual Studio для розробки системи. Також було проведено аналізи та обрано інструменти для розробки: мову програмування було обрано C#, графічний інтерфейс було обрано Windows Forms, проведено аналіз систем управління баз даних та обрано SQLServer.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка бази даних

База даних DiplomaDB яку було створено для зберігання даних необхідних системі має комплексну структуру [18]. Основну структуру розробленої бази даних можна побачити на рис. 3.1.

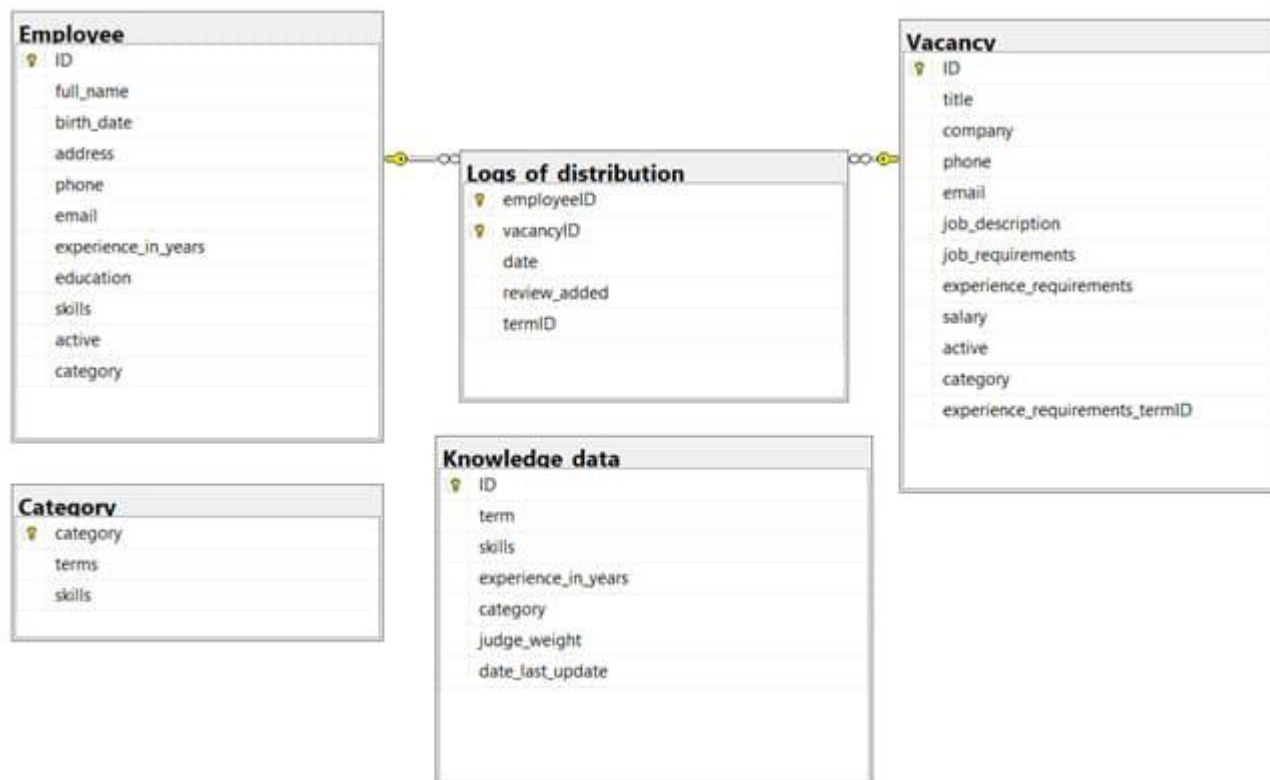


Рисунок 3.1 – Основна структура бази даних DiplomaDB

Нижче буде наведено детальний аналіз кожної таблиці даної бази даних.

Таблиця «Employee» зберігає в собі інформацію про робітників. Детальний опис наведено у табл. 3.1.

Таблиця 3.1 – Таблиця «Employee»

Назва поля	Тип даних поля	Опис поля
ID	int	містить id запису про робітника
full_name	varchar(50)	містить ПІБ робітника

Кінець таблиці 3.1

Назва поля	Тип даних поля	Опис поля
birth_date	date	містить дату народження робітника
address	varchar(50)	містить адресу робітника
phone	varchar(50)	містить контактний телефон робітника
email	varchar(50)	містить контактну пошту робітника
experience_in_years	tinyint	містить досвід роботи робітника у вигляді числа
education	varchar(50)	містить освіту робітника
skills	text	містить перелік того, що робітника вміє, які навички має
active	bit	чи шукає робітник роботу на даний момент, так\ні
category	varchar(50)	категорія праці відповідна робітнику

Таблиця «Vacancy» зберігає в собі інформацію про вакансії. Детальний опис наведено у табл. 3.2.

Таблиця 3.2 – Таблиця «Vacancy»

Назва поля	Тип даних поля	Опис поля
ID	int	містить id запису про вакансію
title	varchar(50)	назва вакансії
company	varchar(50)	назва компанії
phone	varchar(50)	контактний телефон
email	varchar(50)	контактна пошта
job_description	text	опис вакансії
job_requirements	text	вимоги вакансії
experience_requirements	tinyint	вимоги до досвіду роботи
salary	money	пропонована зарплатня

Кінець таблиці 3.2

Назва поля	Тип даних поля	Опис поля
active	bit	чи шукає вакансія робітників на даний момент, так\ні
category	varchar(50)	категорія праці відповідна вакансії
experience_requirements_termID	int	id терміну через який система робила висновок про вимоги до досвіду роботи

Таблиця «Logs_of_distribution» зберігає в собі інформацію про логи розподілених робітників. Детальний опис наведено у табл. 3.3.

Таблиця 3.3 – Таблиця «Logs_of_distribution»

Назва поля	Тип даних поля	Опис поля
employeeID	int	містить id запису про розподілення
vacancyID	int	містить id вакансії на яку розподілили
date	date	містить id робітника якого розподілили
review_added	bit	чи був доданий відгук до цього розподілу, так\ні
termID	int	id терміну через який система робила висновок про вимоги до досвіду роботи

Таблиця «Knowledge_data» зберігає в собі знання про терміни необхідні системі. Детальний опис наведено у табл. 3.4.

Таблиця 3.4 – Таблиця «Knowledge_data»

Назва поля	Тип даних поля	Опис поля
ID	int	містить id запису термін
term	varchar(50)	термін
skills	text	навички відповідні до терміну
experience_in_years	tinyint	досвід роботи відповідний терміну
category	varchar(50)	категорія праці відповідна терміну
judge_weight	smallint	поле для збору інформації про відгуки
date_last_update	date	остання дата оновлення інформації про термін

Таблиця «Category» зберігає в собі дані про терміни та навички відповідних категорій. Детальний опис наведено у табл. 3.5.

Таблиця 3.5 – Таблиця «Category»

Назва поля	Тип даних поля	Опис поля
category	varchar(50)	категорія
terms	text	терміни
skills	text	навички

3.2 Опис реалізованого інтерфейсу

Для зручності взаємодії із системою для користувача був розроблений інтерфейс взаємодії з системою. Нижче буде наведено розбір декількох основних елементів інтерфейсу.

Основним робочим вікном програми є Форма 1 , в ній користувач здійснює такі дії як: робота з даними про вакансії, робота з даними про робітників, перегляд та пошук по вакансіях для розподілу робочих місць та вибір однієї з них, перегляд логів розподілених робочих місць та вибору для складання відгуку. Також вгорі за допомогою елементів `toolStrip1` та `toolStripButton1` була створена кнопка для оновлення даних з бази даних для всієї програми.

Форма 1 була розділена на 4 частини за допомогою елемента `tabControl1`. На першій вкладці `tabControl1 – tabPage1` було створено місце роботи з даними про вакансії за допомогою елемента `dataGridView1` що дозволяє підвантажувати дані з бази даних та відображати їх для взаємодії у вигляді таблиці, на рис. 3.2 зображено вигляд даного елемента.

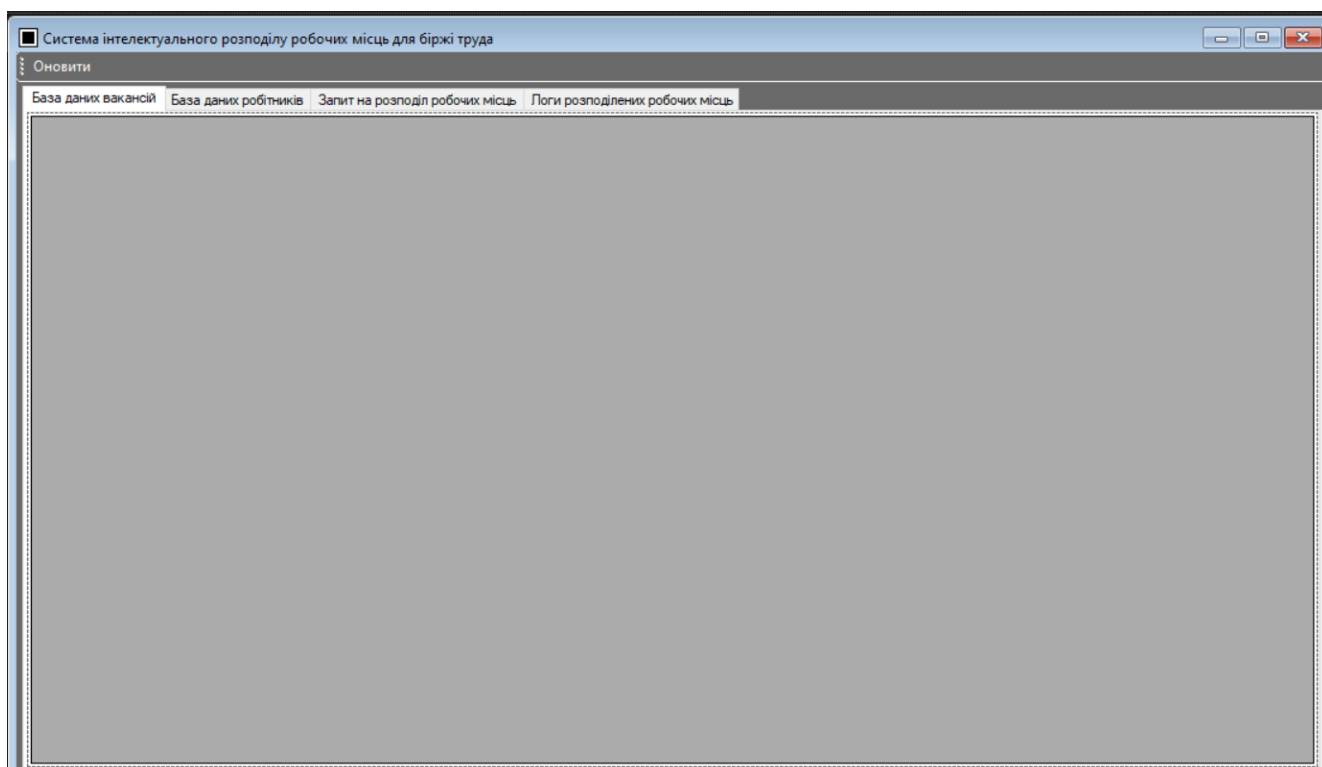


Рисунок 3.2 – Вигляд `tabPage1`

На другій вкладці `tabControl1 – tabPage2` було створено місце роботи з даними про робітників за допомогою елемента `dataGridView2` що дозволяє підвантажувати дані з бази даних та відображати їх для взаємодії у вигляді таблиці, на рис. 3.3 зображено вигляд даного елемента.

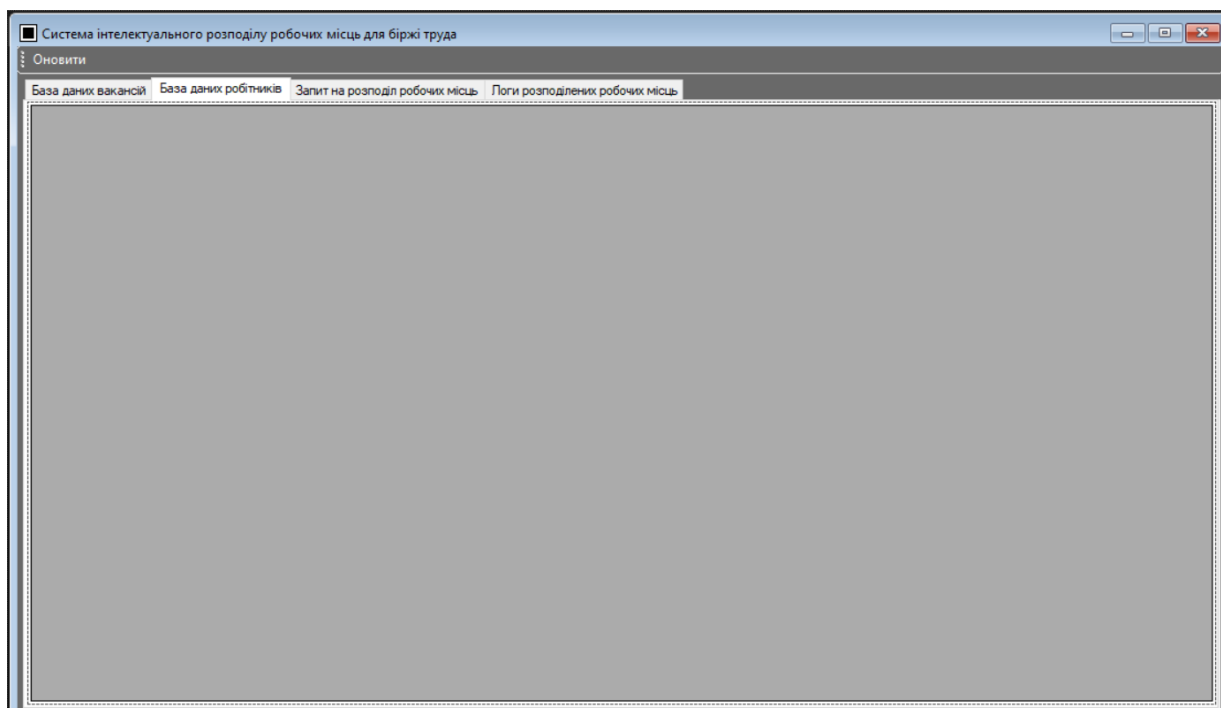


Рисунок 3.3 – Вигляд tabPage2

На третій вкладці `tabControl1 – tabPage3` було створено місце для запитів на розподіл робочих місць, за допомогою елементу `Label1` що текстом вказує на `textBox1` що дозволяє ввести назву компанії та за допомогою кнопки `button1` зробити пошук за цією назвою у елементі `dataGridView3` що дозволяє підвантажувати дані з бази даних та відображати їх для взаємодії у вигляді таблиці, на рис. 3.4 зображено вигляд даного елемента.

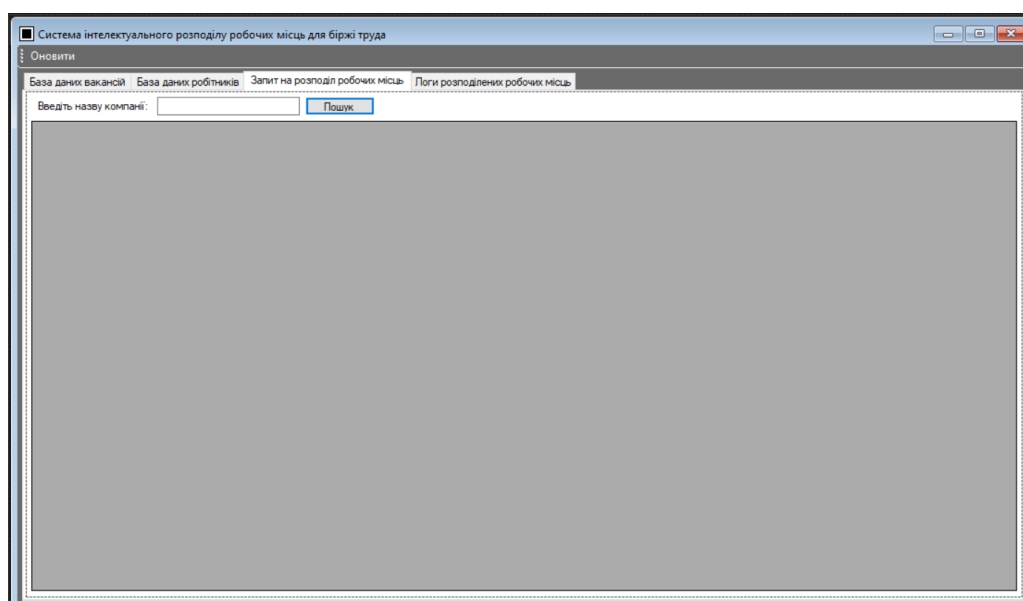


Рисунок 3.4 – Вигляд tabPage3

На четвертій вкладці `tabControll – tabPage4` було створено місце для перегляду та вибору логів розподілених робітників, за допомогою елементу `dataGridView5` що дозволяє підвантажувати дані з бази даних та відображати їх для взаємодії у вигляді таблиці, на рис. 3.5 зображено вигляд даного елемента.

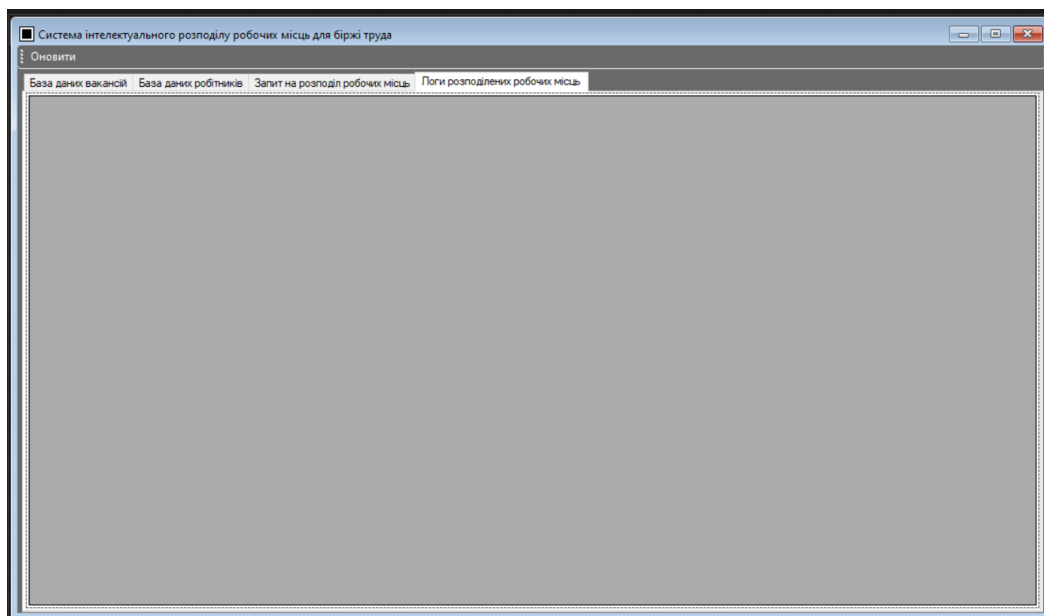


Рисунок 3.5 – Вигляд `tabPage4`

Допоміжним вікном програми є Форма 2, в ній користувач здійснює вибір з топ-10 оптимальних працівників яких система підбрала для вакансії після вибору на `tabPage3`. Зроблено це за допомогою елементу `dataGridView1` що дозволяє підвантажувати дані з бази даних та відображати їх для взаємодії у вигляді таблиці, на рис. 3.6 зображено вигляд даного елемента.

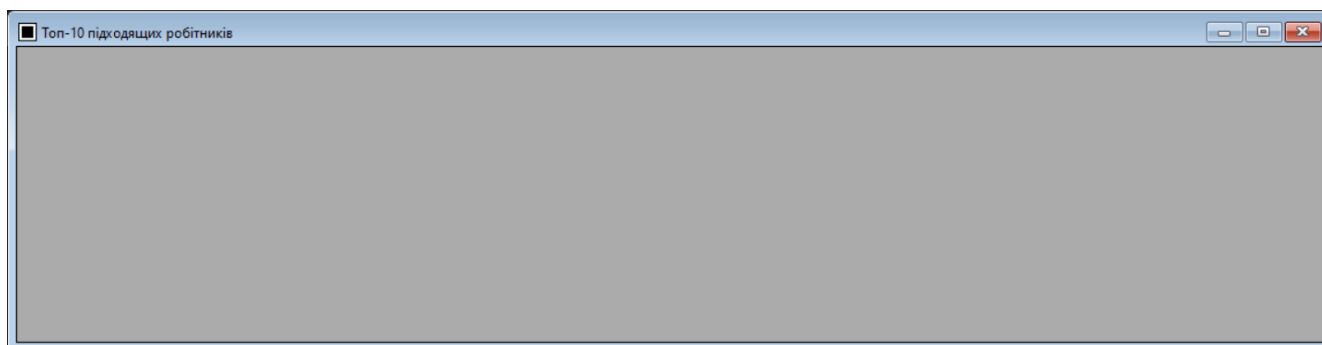


Рисунок 3.6 – Вигляд Форми 2

Останнім допоміжним вікном програми є Форма 3, в ній користувач здійснює складення відгука після вибору на tabPage4. Зроблено це за допомогою елементів label1, label2 та label3 які мають текст для звернення уваги користувача та кнопки button1 та button2 які дозволяють обрати позитивним чи негативним буде відгук, на рис. 3.7 зображено вигляд даного елемента. Так як у цій формі присутні кнопки, був створений обробник події натискання на кнопку, код зображено на рис 3.8, він перевіряє та створює підключення до БД та відправляє в базу даних дані залежно від того яка кнопка натиснута, після чого закриває форму.

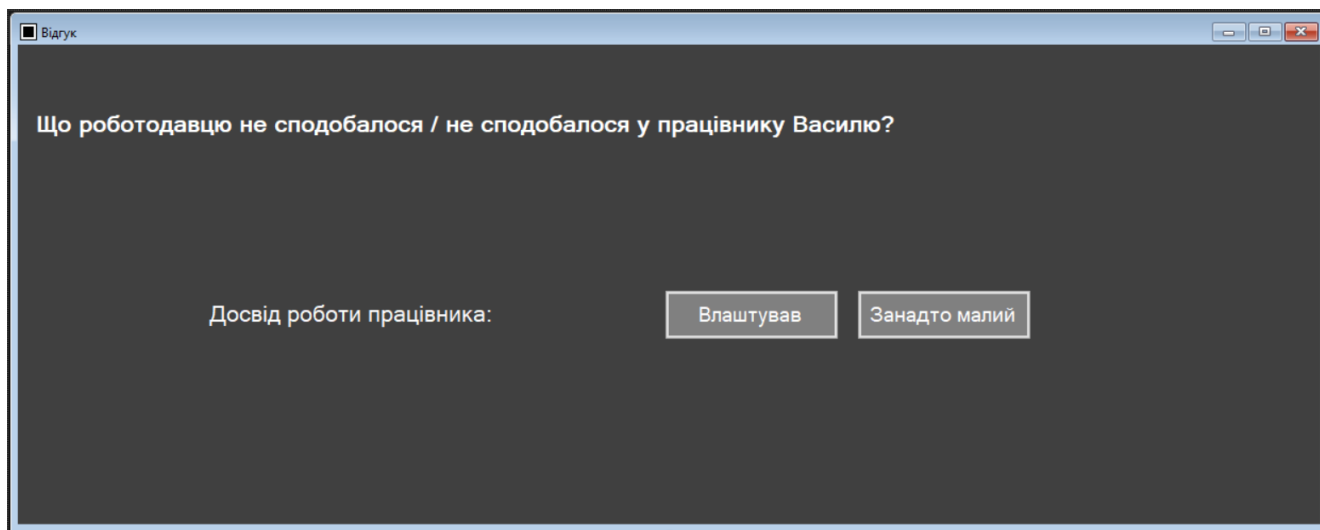


Рисунок 3.7 – Вигляд Форми 3

```
private void button1_Click(object sender, EventArgs e)
{
    int judge_weight;
    using (SqlConnection)
    {
        if (sqlConnection.State == ConnectionState.Closed)
        {
            sqlConnection.Open();
        }

        string query = $"SELECT judge_weight FROM Knowledge_data WHERE ID = {Experience_requirements_termID}";
        using (SqlCommand command = new SqlCommand(query, sqlConnection))
        {
            object result = command.ExecuteScalar();

            if (result != DBNull.Value)
            {
                judge_weight = Convert.ToInt32(result);
            }
            else
            {
                judge_weight = 0;
            }
        }

        judge_weight++;

        string updateQuery = $"UPDATE Knowledge_data SET judge_weight = {judge_weight} WHERE ID = {Experience_requirements_termID}" +
            $"UPDATE Logs_of_distribution SET review_added = 1 WHERE employeeID = {EmployeeID} AND vacancyID = {VacancyID}";

        using (SqlCommand updateCommand = new SqlCommand(updateQuery, sqlConnection))
        {
            updateCommand.ExecuteNonQuery();
        }
    }
    this.Close();
}
```

Рисунок 3.8 – Обробник події натискання на кнопку для Форми 3

3.3 Опис реалізації компонентів програми

Для правильної роботи програмного забезпечення було розроблено декілька класів. Клас «Form1.css» та класи «Form2.css», «Form3.css», які «Form1.css» інстанціює, містять методи обробки подій з елементами інтерфейсу: натискання кнопок, вибір полів таблиць, відкриття нових форм, введення даних та інше. «Form1.css» та «Form3.css» інстанціюють для своєї роботи клас «Interpreter» для обробки даних.

Клас «Interpreter» виконує роль основного модуля аналізу та обробки даних та системи правил у програмі. Нижче наведено детальний опис функцій та призначення основних методів класу «Interpreter». Діаграму класів програми зображено на рис. 3.9.



Рисунок 3.9 – Діаграма класів програми

Метод «ChangeKnowledge_data» оновлює дані про знання в базі даних для обраного терміну. Метод змінює значення поля «judge_weight» на 0 (для обнуління лічильника), якщо число негативних відгуків стало перевищувати позитивні на 5, та підвищує необхідну кількість років у вимогах до досвіду роботи на 1. Також при кожному виклику, тобто після кожного відгука, метод перевіряє термін на сучасність, якщо термін не оновлювалася більше 2 років – метод робить обнуління лічильника для збору нових даних щодо терміну програмою та ставить дату оновлення на сьогоднішню. Код представлено на рис. 3.10.

```

public void ChangeKnowledge_data(int ID)
{
    int judge_weight;
    int experience_in_years;
    SqlConnection sqlConnection = null;
    sqlConnection = new SqlConnection(@"Data Source=DESKTOP-RE15D35;Initial Catalog=DiplomaDB;Integrated Security=True");
    using (sqlConnection)
    {
        if (sqlConnection.State == ConnectionState.Closed)
        {
            sqlConnection.Open();
        }

        string query = $"SELECT judge_weight FROM Knowledge_data WHERE ID = {ID}";
        using (SqlCommand command = new SqlCommand(query, sqlConnection))
        {
            judge_weight = Convert.ToInt32(command.ExecuteScalar());
        }

        query = $"SELECT experience_in_years FROM Knowledge_data WHERE ID = {ID}";
        using (SqlCommand command = new SqlCommand(query, sqlConnection))
        {
            experience_in_years = Convert.ToInt32(command.ExecuteScalar());
        }
        experience_in_years++;

        string updateQuery = $"UPDATE Knowledge_data " +
            $"SET judge_weight = " +
            $"CASE " +
            $"    WHEN DATEDIFF(YEAR, date_last_update, GETDATE()) >= 2 THEN 0 " +
            $"    ELSE judge_weight " +
            $"END, " +
            $"date_last_update = " +
            $"CASE " +
            $"    WHEN date_last_update IS NULL OR DATEDIFF(YEAR, date_last_update, GETDATE()) >= 2 THEN GETDATE() " +
            $"    ELSE date_last_update " +
            $"END " +
            $"WHERE ID = {ID}";

        using (SqlCommand updateCommand = new SqlCommand(updateQuery, sqlConnection))
        {
            updateCommand.ExecuteNonQuery();
        }

        if (judge_weight == -5)
        {
            query = $"SELECT experience_in_years FROM Knowledge_data WHERE ID = {ID}";
            using (SqlCommand command = new SqlCommand(query, sqlConnection))
            {
                experience_in_years = Convert.ToInt32(command.ExecuteScalar());
            }
            experience_in_years++;

            updateQuery = $"UPDATE Knowledge_data SET judge_weight = {0} WHERE ID = {ID}" +
                $"UPDATE Knowledge_data SET experience_in_years = {experience_in_years} WHERE ID = {ID}";

            using (SqlCommand updateCommand = new SqlCommand(updateQuery, sqlConnection))
            {
                updateCommand.ExecuteNonQuery();
            }
        }
    }
    LoadKnowledge_data();
}

```

Рисунок 3.10 – Метод «ChangeKnowledge_data»

Метод «LoadKnowledge_data» завантажує дані про знання з таблиці та записує їх у словник. Код представлено на рис. 3.11.

```
private Dictionary<string, (string skills, int experience, string category, int ID)> LoadKnowledge_data()
{
    var knowledge_data = new Dictionary<string, (string skills, int experience, string category, int ID)>();

    string query = "SELECT ID, term, skills, experience_in_years, category FROM Knowledge_data";

    using (SqlConnection connection = new SqlConnection(@"Data Source=DESKTOP-RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True"))
    {
        SqlCommand command = new SqlCommand(query, connection);
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            string term = reader["term"].ToString();
            string skills = reader["skills"].ToString();
            int experience = Convert.IsDBNull(reader["experience_in_years"]) ? -1 : Convert.ToInt32(reader["experience_in_years"]);
            string category = reader["category"].ToString();
            int ID = int.Parse(reader["ID"].ToString());

            knowledge_data.Add(term, (skills, experience, category, ID));
        }
    }

    return knowledge_data;
}
```

Рисунок 3.11 – Метод «ChangeKnowledge_data»

Метод «AnalyzeJob_skills» аналізує опис вакансії та вимоги до роботи, щоб визначити необхідні навички. Він порівнює знайдені терміни з даними зі словників, щоб зібрати всі необхідні навички для даної вакансії. Код представлено на рис. 3.12.

```
private List<string> AnalyzeJob_skills(string jobDescription, string jobRequirement, string category)
{
    var requiredSkills = new HashSet<string>();

    string fullText = jobDescription + " " + jobRequirement;

    string[] allSkillsArray = categoryData[category].skills.Split(new[] { ' ', '\n' }, StringSplitOptions.RemoveEmptyEntries);

    foreach (string skill in allSkillsArray)
    {
        if (fullText.Contains(skill))
        {
            requiredSkills.Add(skill.TrimStart());
        }
    }

    string[] words = fullText.Split(new[] { ' ', '\n', '\t', '\r', '\f' }, StringSplitOptions.RemoveEmptyEntries);

    for (int i = 0; i < words.Length; i++)
    {
        string term = words[i].Trim();

        if (knowledgeData.ContainsKey(term))
        {
            string skills = knowledgeData[term].skills;

            string[] termSkills = skills.Split(new[] { ' ', '\n', '\t', '\r', '\f' }, StringSplitOptions.RemoveEmptyEntries);

            foreach (var skill in termSkills)
            {
                requiredSkills.Add(skill.TrimStart());
            }
        }

        if (i < words.Length - 1)
        {
            term = string.Join(" ", words[i].Trim(), words[i + 1].Trim());

            if (knowledgeData.ContainsKey(term))
            {
                string skills = knowledgeData[term].skills;

                string[] termSkills = skills.Split(new[] { ' ', '\n', '\t', '\r', '\f' }, StringSplitOptions.RemoveEmptyEntries);

                foreach (var skill in termSkills)
                {
                    requiredSkills.Add(skill.TrimStart());
                }
            }
        }
    }

    return requiredSkills.ToList();
}
```

Рисунок 3.12 – Метод «AnalyzeJob skills»

Метод «CalculateSkillMatchPercentage» обчислює відсоток відповідності навичок та досвіду працівників вимогам вакансії. Він аналізує навички працівників, порівнює їх з необхідними навичками та досвідом для певної вакансії, і повертає список працівників, відсортований за відсотком відповідності. Код представлено на рис. 3.13.

```
public List<(int ID, double Percentage)> CalculateSkillMatchPercentage(int id, string jobDescription, string jobRequirements, int experience_in_years, string category)
{
    DataTable dataTable = new DataTable();
    string query = "SELECT ID, skills, active, experience_in_years, category FROM Employee";
    SqlConnection connection = new SqlConnection(@"Data Source=DESKTOP-RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True");
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        connection.Open();
        using (SqlDataReader reader = command.ExecuteReader())
        {
            dataTable.Load(reader);
        }
    }
    DataTable employeeTable = dataTable;

    var skillMatchPercentages = new List<(int ID, double Percentage)>();
    List<string> requiredSkills = AnalyzeJob_skills(jobDescription, jobRequirements, category);

    if (experience_in_years == -1)
    {
        experience_in_years = AnalyzeJob_experience(id, jobDescription, jobRequirements, category);
    }

    foreach (DataRow employeeRow in employeeTable.Rows)
    {
        if (employeeRow["category"].ToString() == category && (bool)employeeRow["active"] == true)
        {
            if (int.Parse(employeeRow["experience_in_years"].ToString()) >= experience_in_years)
            {
                string employeeSkillsString = employeeRow["skills"].ToString();
                List<string> employeeSkills = AnalyzeSkills(employeeSkillsString, category);
                int matchingSkillsCount = requiredSkills.Intersect(employeeSkills).Count();
                double matchPercentage = (double)matchingSkillsCount / requiredSkills.Count * 100;
                if (matchPercentage > 60)
                {
                    skillMatchPercentages.Add((Convert.ToInt32(employeeRow["ID"]), matchPercentage));
                }
            }
        }
    }

    return skillMatchPercentages.OrderByDescending(x => x.Percentage).Take(10).ToList();
}
```

Рисунок 3.13 – Метод «CalculateSkillMatchPercentage»

Метод «AnalyzeCategory_vacansy» аналізує опис вакансії та вимоги до роботи, щоб визначити відповідну категорію вакансії. Він порівнює кількість входжень термінів та навичок з категоріями зі словника та обирає категорію з найбільшим загальним балом. Код представлено на рис. 3.14.

```

public string AnalyzeCategory_vacancy(string jobDescription, string JobRequirements)
{
    Dictionary<string, (int termCount, int skillCount)> result = new Dictionary<string, (int termCount, int skillCount)>();

    foreach (var category in categoryData)
    {
        string categoryName = category.Key;
        string[] terms = category.Value.terms.Split(new[] { ',' }, StringSplitOptions.RemoveEmptyEntries);
        string[] skills = category.Value.skills.Split(new[] { ',' }, StringSplitOptions.RemoveEmptyEntries);

        int termCount = 0;
        int skillCount = 0;

        foreach (var term in terms)
        {
            termCount += CountOccurrences(jobDescription, term.Trim()) + CountOccurrences(JobRequirements, term.Trim());
        }

        foreach (var skill in skills)
        {
            skillCount += CountOccurrences(jobDescription, skill.Trim()) + CountOccurrences(JobRequirements, skill.Trim());
        }

        result[categoryName] = (termCount, skillCount);
    }

    string maxCategory = null;
    int maxScore = 0;

    foreach (var category in result)
    {
        int totalScore = category.Value.termCount * 3 + category.Value.skillCount;
        if (totalScore > maxScore)
        {
            maxScore = totalScore;
            maxCategory = category.Key;
        }
    }

    return maxCategory;
}

```

Рисунок 3.14 – Метод «AnalyzeCategory_vacancy»

3.4 Опис використаних бібліотек

Для розробки програми були використані декілька бібліотек:

- System.Collections.Generic: бібліотека надає використані в програмі універсальні колекції – списки, словники, черги та стеки. Ці колекції дозволяють забезпечити ефективне управління даними, обробляючи великі обсяги інформації з мінімальними витратами ресурсів. У програмі ця бібліотека використовується для зберігання та маніпулювання колекціями вакансій, робітників та знань;
- System.Data: бібліотека надає класи для роботи з базами даних. Вона забезпечує виконання запитів для роботи з базами даних. У програмі використовується для зв'язку з базою даних;
- System.Data.SqlClient: бібліотека надає класи для взаємодії з базами даних SQL Server. Має згідні з System.Data властивості, та через використання

SQLServer в якості обраної системи управління базами даних має більше використання у програмі;

– System.Windows.Forms: бібліотека надає засоби для створення графічного інтерфейсу для Windows. Вона має різноманітні елементи управління: форми, кнопки, текстові поля, що дозволяють робити інтерактивні та зручні для користувача інтерфейси. У програмі System.Windows.Forms використовується для створення форм, за допомогою яких користувач може взаємодіяти з системою, переглядати та редагувати дані, а також виконувати інші операції.

3.5 Тестування програми

Для забезпечення якості та надійності було проведено декілька видів випробувань.

Модульні тести: перевірка окремих модулів програми на коректність роботи.

Першим модульним тестом було вирішено зробити перевірку на можливість додавання даних до таблиці вакансій.

Мета тесту: перевірити, чи коректно додається запис у таблицю.

Процедура тестування:

- запуск програми;
- введення в останній ряд нових даних про вакансію «Test Java Junior»;
- натискання на кнопку додати;
- підтвердження додавання даних до таблиці програми;
- натискання на кнопку оновити;
- підтвердження додавання даних до бази даних.

Результат тесту: тест пройшов успішно, запис було додано до обох джерел даних.

Тестування представлено на рис. 3.15

Система інтелектуального розподілу робочих місць для біржі труда

Оновити

База даних вакансій База даних робітників Запит на розподіл робочих місць Логи розподіленних робочих місць

	ID	title	company	phone	email	job_description	job_requirements	experience_requirei	salary	active	Co
	30	Software Developer	Tech Solutions Inc.	+1 (555) 11111	hr@techsolutions.co	We are looking for a skilled Software Developer to join our team.	Proficient in HTML, CSS, JavaScript.	5	5000,0000	☒	De
	31	Data Analyst	Data Insights Corp	+1 (555) 22222	careers@datainsight	Seeking a Data Analyst to analyze and interpret complex data sets.	Experienced in Data Visualization and Statistical Analysis.	4	5000,0000	☒	De
	32	Network Security Specialist	SecureNet Ltd	+1 (555) 33333	jobs@securenet.com	We need a Network Security Specialist to protect our systems from cyber threats.	Skilled in Network Security Protocols and Incident Response.	5	5000,0000	☒	De
	33	Registered Nurse	HealthCare Solutions	+1 (555) 44444	apply@healthcare.co	Hiring a Registered Nurse for our busy clinic.	Proficient in Patient Care and Medical Procedures.	2	5000,0000	☒	De
	34	Financial Analyst	FinCorp	+1 (555) 55555	hr@fincorp.com	We are seeking a Financial Analyst to evaluate investment opportunities.	Skilled in Financial Modeling and Market Analysis.	3	5000,0000	☒	De
	35	Physical Therapist	RehabWorks	+1 (555) 66666	careers@rehabwork	Looking for a Physical Therapist to help patients recover from injuries.	Expertise in Physical Therapy and Rehabilitation Techniques.	4	5000,0000	☒	De
	36	High School Teacher	EducationFirst	+1 (555) 77777	jobs@educationfirst	We need a High School Teacher for our English department.	Proficient in Lesson Planning and Classroom Management.	2	5000,0000	☒	De
	37	Investment Analyst	WealthBuilders LLC	+1 (555) 88888	careers@wealthbuil	Seeking an Investment Analyst to research and recommend investment strategies.	Experienced in Financial Markets and Investment Analysis.	5	5000,0000	☒	De
	39	Cybersecurity Specialist	CyberGuard Inc.	+1 (555) 99999	hr@cyberguard.com	We are looking for a Cybersecurity Specialist to protect our digital assets.	Proficient in Network Security and Threat Detection.	4	5000,0000	☒	De
	40	Marketing Manager	MarketLeaders Co.	+1 (555) 00000	jobs@marketleaders	We are hiring a Marketing Manager to develop and execute marketing campaigns.	Skilled in Strategic Marketing and Brand Management.	6	5000,0000	☒	De
	42	Test Java junior	testtestcoo	test	test@test.test	мы ищем junior Java разработчика	знание Java		5000,0000	☒	De
*										☐	

Рисунок 3.15 – Результат тесту додавання запису у таблицю

Другим модульним тестом було вирішено зробити перевірку на можливість додавання відгуку якщо відгук уже був доданий раніше.

Мета тесту: перевірити, чи можливо додавання відгуку якщо відгук уже був доданий раніше.

Процедура тестування:

- перехід на вкладку логи розподілених місць;
- вибір лога з галкою, яка показує, що відгук вже був поставлений;
- підтвердження неможливості створення відгуку.

Результат тесту: тест пройшов успішно, з'являється вікно про помилку, що відгук вже було поставлено, можна продовжувати роботу в програмі після закриття вікна про помилку.

Тестування представлено на рис. 3.16

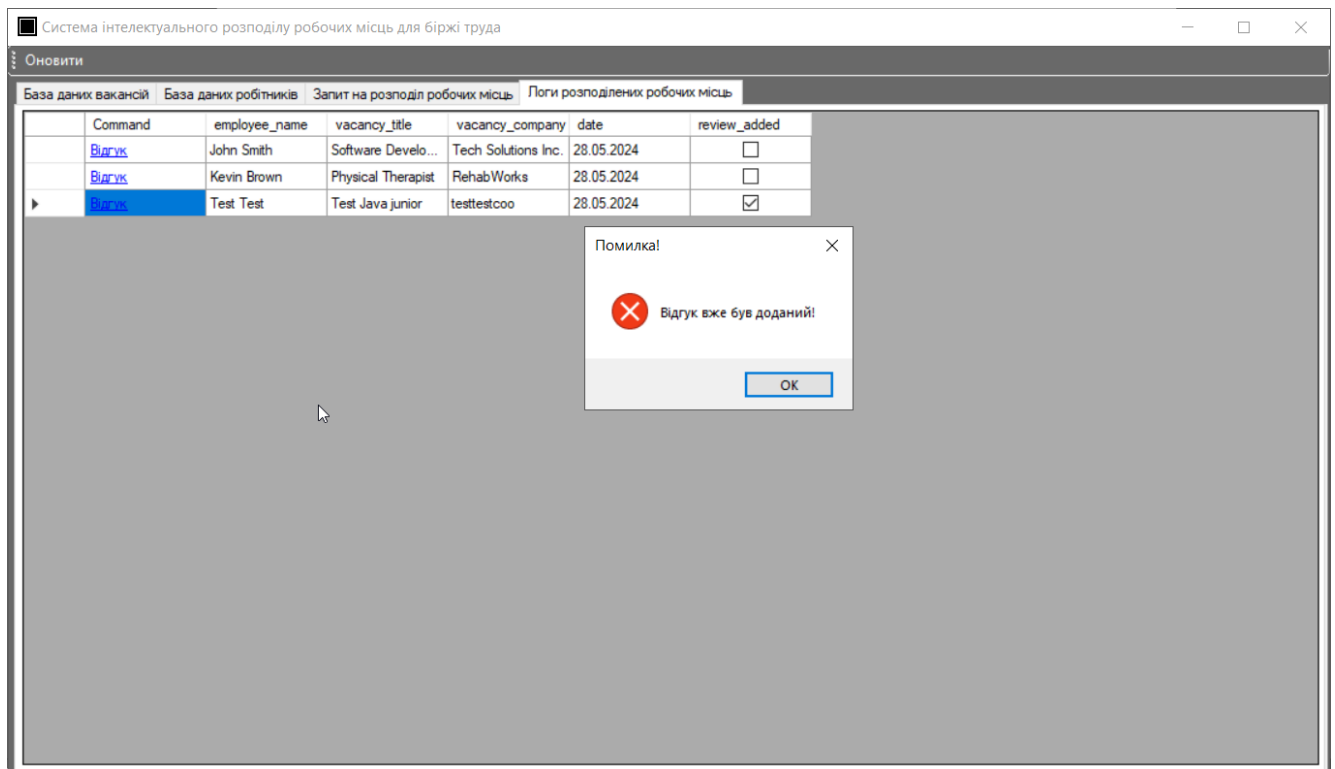


Рисунок 3.16 – Результат тесту додавання відгуку якщо відгук уже був доданий раніше

Функціональні тести: перевірка відповідності функціональних вимог програми.

Першим функціональним тестом було вирішено зробити перевірку правильного розподілу робочих місць.

Мета тесту: перевірити, чи коректно робиться розподіл робочих місць.

Процедура тестування:

- запуск програми;
- введення в останній ряд нових даних про вакансію «Test Java Junior»;
- введення в останній ряд нових даних про робітника з 80% навичок та потрібним досвідом роботи для «Test Java Junior»;
- перехід до вкладки запиту на розподіл робочих місць;
- вибір вакансії яку ми додали;
- підтвердження правильності відсотка відповідності робітника вакансії.

Результат тесту: тест пройшов успішно, отриманий відсоток відповідності робітника вакансії є правильним.

Тестування представлено на рис. 3.17

Обрати	% of suitability	ID	Full Name	Birth Date	Address	Phone	Email	Experience (years)	Education	Skills
Обрати	80	60	Test Test	12.12.2012 00:0...	Teststreet	test	test@test.test	5	High	0

Рисунок 3.17 – Результат тесту на перевірку правильного розподілу робочих місць

Другим функціональним тестом було вирішено зробити перевірку на адаптацію системи.

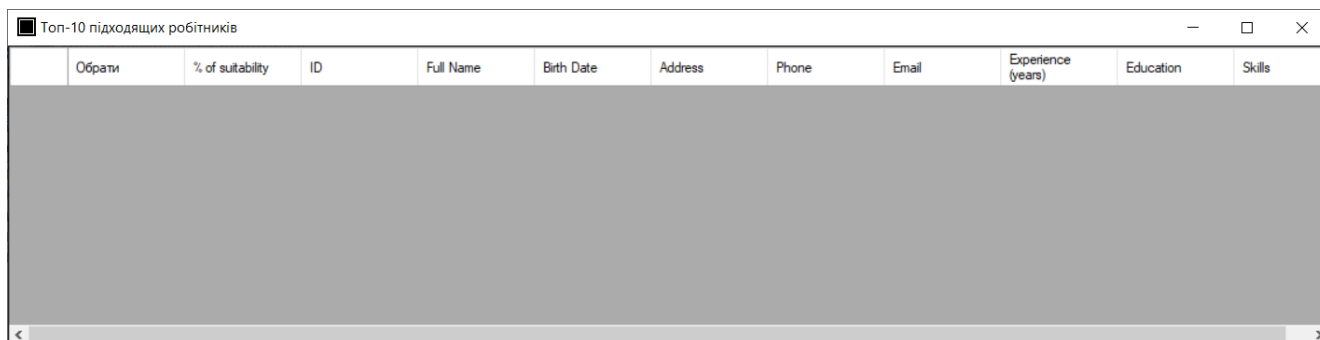
Мета тесту: перевірити, чи адаптується система через відгуки.

Процедура тестування:

- відкриття програми;
- створення 6 працівників з навичками та досвідом роботи рівними певному терміну;
- створення вакансії без вказівки досвіду роботи та зазначенням терміну, який ми вибрали на другому кроці, у вимогах до вакансії;
- відправлення 5 працівників на цю вакансію за допомогою форми запитів на розподіл робочих місць;
- залишення 5 негативних відгуків за допомогою форми логів розподілу;
- спроба відправлення 6-го робітника;
- підтвердження відсутності 6-го працівника у списку робітників для відправки на створену вакансію.

Результат тесту: тест пройшов успішно, система адаптувалася через відгуки.

Тестування представлено на рис. 3.18



Обрати	% of suitability	ID	Full Name	Birth Date	Address	Phone	Email	Experience (years)	Education	Skills

Рисунок 3.18 – Результат тесту на перевірку на адаптації систем

3.6 Висновки за розділом

У цьому розділі були розглянуті ключові аспекти функціонування системи. Починаючи з алгоритму роботи програми, було розкрито, як система взаємодіє з базою даних та інтерфейсом користувача для ефективного управління даними та взаємодії з користувачем. Опис реалізованих таблиць бази даних відображає структуру даних, яка використовується для зберігання інформації про вакансії, робітників та інші аспекти системи. Далі було детально розглянуто інтерфейс користувача, що включає форми взаємодії та їх функціональні можливості. У цьому контексті були використані певні бібліотеки, що допомагають у розробці та оптимізації програмного забезпечення. Також у розділі наведено опис процесу тестування програми, що включає в себе перевірку функціональності, надійності та продуктивності системи.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Основні завдання цієї програми полягають у спрощенні процесу розподілення робочих місць для працівників біржи труда, покращення його якості та ефективності.

4.2 Характеристики програми

Функції системи описані у ТЗ п. 1.5.3. Вимоги до технічних засобів та ОС наведені у ТЗ п. 1.5.6.

Для запуску програмного забезпечення необхідно виконати файл DiplomaAPP.exe. Програма була створена мовою C# у середовищі Visual Studio 2020 на платформі .NET Framework 4.8. Програма має інтерфейс який дозволяє користувачу здійснювати такі дії як: робота з даними про вакансії, робота з даними про робітників, перегляд та пошук по вакансіях для розподілу робочих місць та вибір однієї з них, перегляд логів розподілених робочих місць та вибору для складання відгуку.

4.3 Структура програми

Загальну структуру програми наведено на рис. 4.1, вона складається з наступних компонентів: файлів бібліотек, програмного коду, форм.

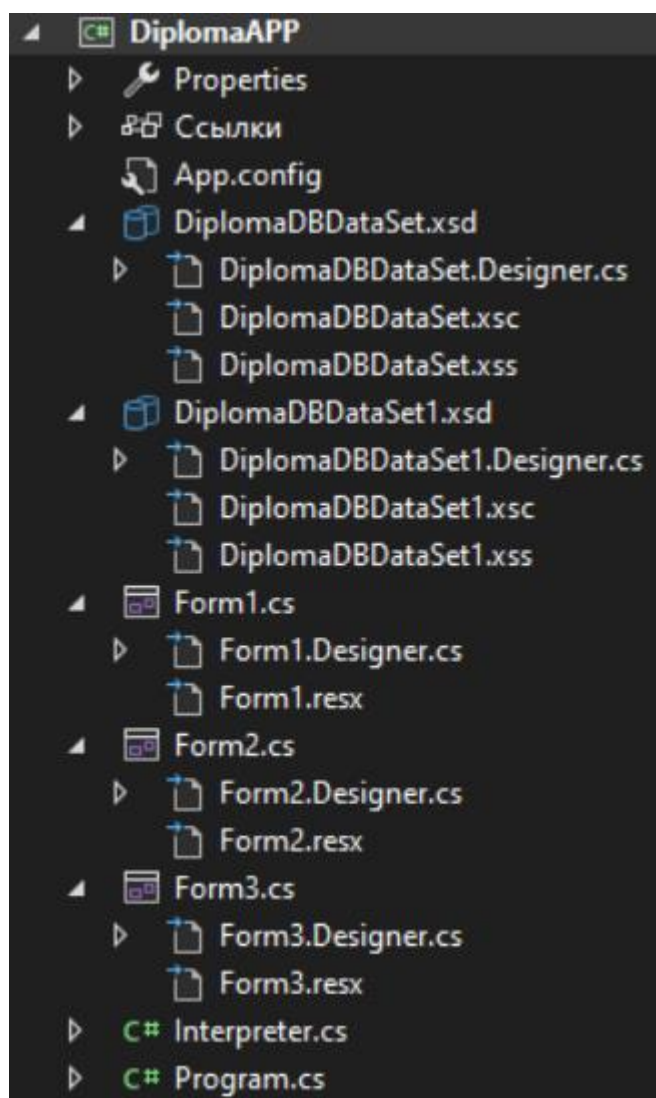


Рисунок 4.1 – Структура програми

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Вхідні дані системи включають:

- знання експерта про необхідні характеристики робітників та їх значення для вимог роботодавців;
- вимоги щодо робітників від підприємництва введені в поля інтерфейсу програми користувачем;
- дані про робітника що шукає роботу, включаючи інформацію про їх навички, досвід роботи тощо введені в поля інтерфейсу програми користувачем;

- оцінка результату роботи системи та вказівка чи сподобалося або не сподобалося, від роботодавців, які її використовували її результат введені в поля інтерфейсу програми користувачем.

4.4.2 Вихідні дані

Вихідні дані системи включають:

- дані щодо вимог роботодавців та параметрів робітників, які були отримані та систематизовані системою за допомогою системи правил;
- інформація про робітників, які відповідають вимогам вакансії та відсоток того, наскільки ці робітники задовольняють вимогам вакансії який був розрахований за допомогою системи правил.

4.5 Повідомлення програмісту

Програма містить такі повідомлення «Не вдалося підключитися до бази даних!». Воно означає, що програма не змогла підключитися до обраної бази даних.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Основні завдання цієї програми полягають у спрощенні процесу розподілення робочих місць для працівників біржи труда, покращення його якості та ефективності.

5.2 Умови виконання програми

Для забезпечення роботи програми необхідний склад апаратно-технічних засобів наведений у п. 1.5.6.

Для роботи з програмою від користувача потрібні:

- знання та володіння українською та англійською мовою;
- навички користування комп'ютерною мишею та клавіатурою.

5.3 Виконання програми

Програма запускається за допомогою виконуваного файлу DiplomaAPP.exe шляхом подвійного натискання комп'ютерної миші.

При запуску програми користувачу відображається головне вікно програми – Форма 1, яка зображена на рис. 5.1.

База даних вакансій										
База даних робітників		Запит на розподіл робочих місць		Логи розподіленних робочих місць						
	title	company	phone	email	job_description	job_requirements	experience_require	salary	active	Command
►	Software Developer	Tech Solutions Inc.	+1 (555) 11111	hr@techsolutions.co	We are looking for a skilled Software	Proficient in HTML, CSS, Java Script	5	5000.0000	☒	Delete
	Data Analyst	Data Insights Corp	+1 (555) 22222	careers@datainsight	Seeking a Data	Experienced in Data Visualization	4	5000.0000	☒	Delete
	Network Security	SecureNet Ltd	+1 (555) 33333	jobs@securenet.co	We need a	Skilled in Network Security	5	5000.0000	☒	Delete
	Registered Nurse	HealthCare Solutions	+1 (555) 44444	apply@healthcare.co	Hiring a Registered Nurse to provide	Proficient in Medication	2	5000.0000	☒	Delete
	Financial Analyst	FinCorp	+1 (555) 55555	hr@fincorp.com	We are seeking a	Skilled in Financial Analysis	3	5000.0000	☒	Delete
	Physical Therapist	RehabWorks	+1 (555) 66666	careers@rehabwork	Looking for a	Expertise in Physical Therapy	4	5000.0000	☒	Delete
	High School Teacher	EducationFirst	+1 (555) 77777	jobs@educationfirst	We need a High School Teacher	Proficient in Lesson Planning	2	5000.0000	☒	Delete
	Investment Analyst	WealthBuilders LLC	+1 (555) 88888	careers@wealthbuild	Seeking an	Experienced in Financial Markets	5	5000.0000	☒	Delete
	Cybersecurity	CyberGuard Inc.	+1 (555) 99999	hr@cyberguard.com	We are looking for	Proficient in Malware Analysis	4	5000.0000	☒	Delete
	Marketing Manager	MarketLeaders Co.	+1 (555) 00000	jobs@marketleaders	We are hiring a	Skilled in Strategic Marketing	6	5000.0000	☒	Delete
	Test Java junior	testtestcoo	test	test@test.test	сынап ab, Java	фыафыаа Java		5000.0000	☒	Delete
*									☐	

Рисунок 5.1 – Форма 1

На Формі 1 у лівому верхньому куті є кнопка «Оновити», при натисканні на яку дані, що відображаються на формі, оновлюються згідно з базою даних. Для профілактики рекомендується натискати цю кнопку перед кожною справою з даними таблиць програми. Кнопка зображена на рис. 5.2.

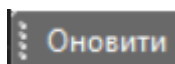


Рисунок 5.2 – Кнопка «Оновити»

Перша вкладка «База даних вакансій» це місце роботи з даними про вакансії. На цій вкладці можна оновлювати та додавати дані про вакансії, для цього необхідно змінити дані у клітинках таблиці та натиснути на команду у одинадцятому рядку. У випадку потреби видалення треба натиснути на команду «Delete» та у вікні, яке зображено на рис. 5.3, підтвердити видалення запису. Команди додання та оновлення зображені на рис. 5.4.

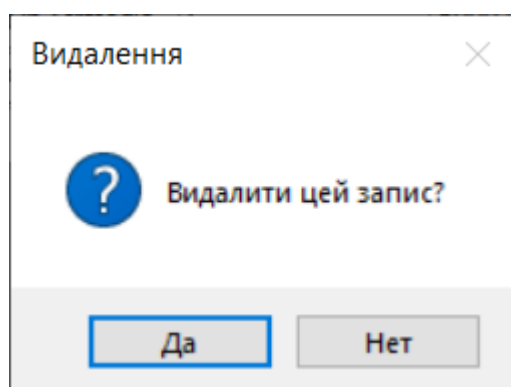


Рисунок 5.3 – Вікно видалення

[Update](#)

[Insert](#)

Рисунок 5.4 – Кнопки «Оновити» та «Додати»

Друга вкладка «База даних робітників» має той же процес роботи на ній як і попередня вкладка, тут можна взаємодіяти з даними про працівників.

Третя вкладка "Запит на розподілення робочих місць", зображена на рис. 5.5, це вкладка для створення запитів на розподілення робочих місць. Щоб

працювати з цією вкладкою, потрібно ввести в поле поруч із написом "Введіть назву" потрібну компанію, натиснути кнопку "Пошук" і з виведених у таблиці вакансій вибрати потрібну нам для розподілу працівників для неї. Після цих дій буде відкрито вікно, зображене на рис. 5.6, у цьому вікні будуть показані топ-10 працівників, які підходять до цієї вакансії. Після вибору одного шляхом натискання на кнопку "Вибрати" він буде вказаний в системі як людина, яка була розподілена на дану вакансію. Після вибору вікно закриється і ви зможете продовжувати роботу з головним вікном програми.

Введіть назву компанії:

	Command	ID	title	company	phone	email	job_description	job_requirements	experience_require	salary	
►	Обрати	30	Software Develo...	Tech Solutions Inc.	+1 (555) 11111	hr@techsolutions...	We are looking f...	Proficient in HTM...	5	5000,0000	
	Обрати	31	Data Analyst	Data Insights Corp	+1 (555) 22222	careers@datainsi...	Seeking a Data ...	Experienced in D...	4	5000,0000	
	Обрати	32	Network Security...	SecureNet Ltd	+1 (555) 33333	jobs@securenet...	We need a Netw...	Skilled in Networ...	5	5000,0000	
	Обрати	33	Registered Nurse	HealthCare Servi...	+1 (555) 44444	apply@healthcar...	Hiring a Register...	Proficient in Medi...	2	5000,0000	
	Обрати	34	Financial Analyst	FinCorp	+1 (555) 55555	hr@fincorp.com	We are seeking ...	Skilled in Budgeti...	3	5000,0000	
	Обрати	35	Physical Therapist	RehabWorks	+1 (555) 66666	careers@rehabw...	Looking for a Phy...	Expertise in Ther...	4	5000,0000	
	Обрати	36	High School Tea...	EducationFirst	+1 (555) 77777	jobs@educationfi...	We need a High ...	Proficient in Less...	2	5000,0000	
	Обрати	37	Investment Analyst	WealthBuilders L...	+1 (555) 88888	careers@wealthb...	Seeking an Inves...	Experienced in Fi...	5	5000,0000	
	Обрати	39	Cybersecurity En...	CyberGuard Inc.	+1 (555) 99999	hr@cyberguard.c...	We are looking f...	Proficient in Netw...	4	5000,0000	
	Обрати	40	Marketing Manager	MarketLeaders Co.	+1 (555) 00000	jobs@marketlead...	We are hiring a ...	Skilled in Strategi...	6	5000,0000	
	Обрати	42	Test Java junior	testtestcoo	test	test@test.test	сынааn ab, Java...	фыафыава Java...		5000,0000	

Рисунок 5.5 – Вкладка «Запит на розподіл робочих місць»

■ Топ-10 підходящих робітників

	Обрати	% of suitability	ID	Full Name	Birth Date	Address	Phone	Email	Experience (years)	Education	Skills
►	Обрати	100	26	John Smith	10.08.1990 00:0...	123 Main St, Any...	+1 (555) 12345	john.smith@exam...	5	Bachelor's in Co...	0

Рисунок 5.6 – Вікно «Топ-10 підходящих робітників»

Четверта вкладка "Логи розподілених робочих місць", зображена на рис. 5.7, це вкладка для створення відгуків на розподіленні робочі місця. Якщо ви

отримали відгук на результат розподілу робочого місця, ви повинні обрати лог цього результату, шляхом натиснення кнопки «Відгук». Після цього відкрито вікно, зображене на рис. 5.8, у цьому вікні будуть дві кнопки «Влаштував» та «Занадто малий», якщо роботодавець скаржиться на занадто малий досвід робітника розподіленого до нього – натисніть на кнопку «Занадто малий», у будь-якому іншому випадку натисніть на «Влаштував». Після вибору вікно закриється і ви зможете продовжувати роботу з головним вікном програми.

Для закінчення роботи з програмою закрийте її шляхом натискання на хрестик у верхньому правому кутку.

	Command	employee_name	vacancy_title	vacancy_company	date	review_added
	Відгук	John Smith	Software Develo...	Tech Solutions Inc.	29.05.2024	☐
►	Відгук	Test Test	Test Java junior	testtestcoo	29.05.2024	☐

Рисунок 5.7 – Вкладка «Логи розподілених робочих місць»

Відгук

Що роботодавцю не сподобалося / не сподобалося у працівнику Test Test?

Досвід роботи працівника:

Влаштував

Занадто малий

Рисунок 5.8 – Вікно «Відгук»

5.4 Повідомлення оператору

Програма містить такі повідомлення як:

- «Відгук вже був доданий!» – означає, що було обрано лог розподілення з вже доданим відгуком;
- «Роботодавець сам вказував потрібний досвід роботи або не вказував посади чи термінів, що мають на увазі певний досвід роботи!» – означає, що обраний лог розподілення має вакансію в якій було вказано необхідний досвід роботи, тож відгук неможливий.

ВИСНОВКИ

У цій дипломній роботі була розглянута проблема неефективності та складнощів, які виникають у інспектора з працевлаштування при розподілі робочих місць на біржі труда. З метою покращення процесу та забезпечення ефективної відповідності між вимогами роботодавців та параметрами робітників, була розроблена інтелектуальна система розподілу робочих місць.

На перших етапах дослідження було зроблено аналіз вимог ринку праці та огляд існуючих рішень у сфері працевлаштування. На основі цього аналізу були визначені функціональні та технічні вимоги до системи, а також сплановано архітектуру програмного продукту.

Під час реалізації програмного продукту був розроблений інтерфейс користувача, база даних та знань для зберігання інформації про вакансії та робітників, система продукційних правил для автоматизованого розподілу робочих місць та можливості збіру відгуків для адаптації системи.

Після завершення розробки програмного продукту були проведені тестування та налагодження для виявлення та усунення помилок. Тестування показало, що система здатна ефективно здійснювати розподіл робочих місць на основі продукційних правил.

Результатом роботи є програмна система, яка забезпечує ефективний розподіл робочих місць на біржі праці та підвищує якість цього процесу завдяки використанню системи продукційних правил. Це дозволяє підвищити точність і швидкість прийняття рішень щодо відповідності робітників до вакансій.

Отже, розроблена система інтелектуального розподілу робочих місць може використовуватися як інструмент для покращення роботи біржі труда для забезпечення оптимального використання людських ресурсів у сучасному ринковому середовищі. Інтелектуальні можливості системи дозволяють їй адаптуватися до змін вимог ринку праці та автоматизувати складні процеси розподілу робочих місць.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Фединець Н.І. Соціологія ринку [Електронний ресурс] / Н.І. Фединець, Ю.Б. Миронов, М.Р. Гонська. – Режим доступу: <https://kerivnyk/sociologia-rynku6/>
2. Стаття 46. Підходяща робота для безробітного. Верховна Рада України – [Електронний ресурс]. – Режим доступу: <https://i.factor.ua/ukr/law-50/section-261/article-32582/>
3. LinkedIn Talent Solutions [Electronic resource]. – Access mode: <https://business.linkedin.com/talent-solutions/>
4. Greenhouse [Electronic resource]. – Access mode: <https://www.greenhouse.com/>
5. SAP SuccessFactors [Electronic resource]. – Access mode: <https://www.sap.com/products/hcm.html/>
6. Про затвердження тем та керівників дипломних кваліфікаційних робіт денної форми навчання освітнього ступеню "бакалавр": Наказ Національного університету "Запорізька політехніка" від 24.04.2024 №168. – С. 10
7. Субботін, С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навч. посіб. / С.О. Субботін. – Запоріжжя: ЗНТУ, 2008. – С. 8
8. What is Visual Studio? [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022/>
9. JetBrains IntelliJ IDEA [Electronic resource]. – Access mode: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html#coding-assistance/>
10. Why should Java developers consider using Eclipse IDE? [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/why-should-java-developers-consider-using-eclipse-ide-vinay-kumar/>
11. NetBeans [Electronic resource]. – Access mode: <https://netbeans.apache.org/front/main/index.html#:~:text=Apache%20NetBeans%20is>

%20much%20more,of%20handy%20and%20powerful%20tools.&text=Apache%20Net Beans%20provides%20editors%2C%20wizards,PHP%20and%20many%20other%20la nguages./

12. 15 reasons why you should learn C# in 2024 [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/15-reasons-why-you-should-learn-c-2023-andrea-angella/>

13. Top 10 Reasons to Learn Java in 2024 [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/reasons-to-learn-java/>

14. 12 Reasons Why You Should Learn Python (2024) [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/reasons-why-you-should-learn-python/>

15. Desktop Guide (Windows Forms [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/>

16. What are the Benefits of Using WPF? [Electronic resource]. – Access mode: <https://snapstack.cz/what-are-the-benefits-of-using-wpf/>

17. What is Actually the Universal Windows Platform [Electronic resource]. – Access mode: <https://www.thomasclaudiushuber.com/2021/02/05/what-is-actually-the-universal-windows-platform-and-what-is-winui-msix-and-project-reunion/>

18. Brembilla, Alice How to set up a database?—a five-step process / Alice Brembilla - Vol 10, Supplement 29: Journal of Thoracic Disease (Management of Database), 2018. – C. 12

ДОДАТОК А
ТЕКСТ ПРОГРАМИ

Form1.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace DiplomaAPP
{
    public partial class Form1 : Form
    {
        Interpreter interpreter = new Interpreter();

        private SqlConnection sqlConnection = null;

        private SqlCommandBuilder sqlBuilder1= null;
        private SqlDataAdapter sqlDataAdapter1= null;
        private DataSet dataSet1 = null;
        private bool newRowAdding1 = false;

        private SqlCommandBuilder sqlBuilder2 = null;
        private SqlDataAdapter sqlDataAdapter2 = null;
        private DataSet dataSet2 = null;
        private bool newRowAdding2 = false;

        private SqlCommandBuilder sqlBuilder3 = null;
        private SqlDataAdapter sqlDataAdapter3 = null;
        private DataSet dataSet3 = null;
        private bool newRowAdding3 = false;

        private SqlCommandBuilder sqlBuilder4 = null;
        private SqlDataAdapter sqlDataAdapter4 = null;
        private DataSet dataSet4 = null;
        private bool newRowAdding4 = false;

        private SqlCommandBuilder sqlBuilder5 = null;
        private SqlDataAdapter sqlDataAdapter5 = null;
        private DataSet dataSet5 = null;
        private bool newRowAdding5 = false;

        public Form1()
        {

```

```

    InitializeComponent();
}

public Form3 Form3
{
    get => default;
    set
    {
    }
}

public Form2 Form2
{
    get => default;
    set
    {
    }
}

private void Form1_Load(object sender, EventArgs e)
{
    DataGridView dataGridView1 = new DataGridView
    {
        Dock = DockStyle.Fill,
        AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
        DefaultCellStyle = { WrapMode = DataGridViewTriState.True },
        AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
    };

    DataGridView dataGridView2 = new DataGridView
    {
        Dock = DockStyle.Fill,
        AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
        DefaultCellStyle = { WrapMode = DataGridViewTriState.True },
        AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
    };

    DataGridView dataGridView3 = new DataGridView
    {
        Dock = DockStyle.Fill,
        AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
        DefaultCellStyle = { WrapMode = DataGridViewTriState.True },
        AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
    };
}

```

```

DataGridView dataGridView4 = new DataGridView
{
    Dock = DockStyle.Fill,
    AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
    DefaultCellStyle = { WrapMode = DataGridViewTriState.True },
    AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
};

DataGridView dataGridView5 = new DataGridView
{
    Dock = DockStyle.Fill,
    AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
    DefaultCellStyle = { WrapMode = DataGridViewTriState.True },
    AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
};

SqlConnection = new SqlConnection(@"Data Source=DESKTOP-
RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True");

SqlConnection.Open();

LoadData1();
LoadData2();
LoadData3();
LoadData5();
}
private void toolStripButton1_Click(object sender, EventArgs e)
{
    ReloadData1();
    ReloadData2();
    ReloadData3();
    ReloadData5();
}

private void LoadData1()
{
    try
    {
        sqlDataAdapter1= new SqlDataAdapter("SELECT ID, title, company, phone,
email, job_description, job_requirements, experience_requirements, salary, active,
category, 'Delete' AS [Command] FROM VACANCY", SqlConnection);

        sqlBuilder1= new SqlCommandBuilder(sqlDataAdapter1);
    }
    catch { }
}

```

```

sqlBuilder1.GetInsertCommand();
sqlBuilder1.GetUpdateCommand();
sqlBuilder1.GetDeleteCommand();

dataSet1 = new DataSet();

sqlDataAdapter1.Fill(dataSet1, "Vacancy");

dataGridView1.DataSource = dataSet1.Tables["Vacancy"];

for (int i = 0; i < dataGridView1.Rows.Count; i++)
{
    DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

    dataGridView1[11, i] = linkCell;
}

dataGridView1.Columns["category"].Visible = false;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}

private void ReloadData1()
{
    try
    {
        dataSet1.Tables["Vacancy"].Clear();

        sqlDataAdapter1.Fill(dataSet1, "Vacancy");

        dataGridView1.DataSource = dataSet1.Tables["Vacancy"];

        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

            dataGridView1[11, i] = linkCell;
        }
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        if (e.ColumnIndex == 11)
        {
            string task =
dataGridView1.Rows[e.RowIndex].Cells[11].Value.ToString();

            if (task == "Delete")
            {
                if (MessageBox.Show("Видалити цей запис?", "Видалення",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    int rowIndex = e.RowIndex;

                    dataGridView1.Rows.RemoveAt(rowIndex);

                    dataSet1.Tables["Vacancy"].Rows[rowIndex].Delete();

                    sqlDataAdapter1.Update(dataSet1, "Vacancy");
                }
            }
            else if (task == "Insert")
            {
                int rowIndex = dataGridView1.Rows.Count - 2;

                string category =
interpreter.AnalyzeCategory_vacancy(dataGridView1.Rows[rowIndex].Cells["job_desc
ription"].Value.ToString(),
dataGridView1.Rows[rowIndex].Cells["job_requirements"].Value.ToString());

                DataRow row = dataSet1.Tables["Vacancy"].NewRow();

                row["ID"] = dataGridView1.Rows[rowIndex].Cells["ID"].Value;
                row["title"] = dataGridView1.Rows[rowIndex].Cells["title"].Value;
                row["company"] =
dataGridView1.Rows[rowIndex].Cells["company"].Value;
                row["phone"] = dataGridView1.Rows[rowIndex].Cells["phone"].Value;
            }
        }
    }
    catch { }
}

```

```

        row["email"] = dataGridView1.Rows[rowIndex].Cells["email"].Value;
        row["job_description"]
dataGridView1.Rows[rowIndex].Cells["job_description"].Value;
        row["job_requirements"]
dataGridView1.Rows[rowIndex].Cells["job_requirements"].Value;
        row["experience_requirements"]
dataGridView1.Rows[rowIndex].Cells["experience_requirements"].Value;
        row["salary"] = dataGridView1.Rows[rowIndex].Cells["salary"].Value;
        row["active"] = dataGridView1.Rows[rowIndex].Cells["active"].Value;
        row["category"] = category;

        dataSet1.Tables["Vacancy"].Rows.Add(row);

dataSet1.Tables["Vacancy"].Rows.RemoveAt(dataSet1.Tables["Vacancy"].Rows.Count
- 1);

        dataGridView1.Rows.RemoveAt(dataGridView1.Rows.Count - 2);

        dataGridView1.Rows[e.RowIndex].Cells[11].Value = "Delete";

        sqlDataAdapter1.Update(dataSet1, "Vacancy");

        newRowAdding1 = false;
    }
    else if (task == "Update")
    {
        int r = e.RowIndex;

        string category
interpreter.AnalyzeCategory_vacancy(dataGridView1.Rows[r].Cells["job_description"]
.Value.ToString(),
dataGridView1.Rows[r].Cells["job_requirements"].Value.ToString());

        dataSet1.Tables["Vacancy"].Rows[r]["ID"]
dataGridView1.Rows[r].Cells["ID"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["title"]
dataGridView1.Rows[r].Cells["title"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["company"]
dataGridView1.Rows[r].Cells["company"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["phone"]
dataGridView1.Rows[r].Cells["phone"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["email"]
dataGridView1.Rows[r].Cells["email"].Value;

```

```

        dataSet1.Tables["Vacancy"].Rows[r]["job_description"] =
dataGridView1.Rows[r].Cells["job_description"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["job_requirements"] =
dataGridView1.Rows[r].Cells["job_requirements"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["experience_requirements"] =
dataGridView1.Rows[r].Cells["experience_requirements"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["salary"] =
dataGridView1.Rows[r].Cells["salary"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["active"] =
dataGridView1.Rows[r].Cells["active"].Value;
        dataSet1.Tables["Vacancy"].Rows[r]["category"] = category;

        dataGridView1.Rows[e.RowIndex].Cells[11].Value = "Delete";

        sqlDataAdapter1.Update(dataSet1, "Vacancy");
    }

    ReloadData1();
}
}
catch (Exception ex)
{
    MessageBox.Show("Не вдалося підключитися до бази даних!",
"Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void dataGridView1_UserAddedRow(object sender,
DataGridViewRowEventArgs e)
{
    try
    {
        if (newRowAdding1 == false)
        {
            newRowAdding1 = true;

            int lastRow = dataGridView1.Rows.Count - 2;

            DataGridViewRow row = dataGridView1.Rows[lastRow];

            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

            dataGridView1[11, lastRow] = linkCell;

            row.Cells["Command"].Value = "Insert";

```

```

    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

private void dataGridView1_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        DataGridView dataGridView = sender as DataGridView;
        if (dataGridView != null)
        {
            dataGridView.AutoResizeRow(e.RowIndex);
        }
        if (newRowAdding1 == false)
        {
            int rowIndex = dataGridView1.SelectedCells[0].RowIndex;

            DataGridViewRow editingRow = dataGridView1.Rows[rowIndex];

            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

            dataGridView1[11, rowIndex] = linkCell;

            editingRow.Cells["Command"].Value = "Update";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void LoadData2()
{

```

```

try
{
    sqlDataAdapter2 = new SqlDataAdapter("SELECT ID, full_name, birth_date,
address, phone, email, experience_in_years, education, skills, active, category, 'Delete'
AS [Command] FROM EMPLOYEE", sqlConnection);

    sqlBuilder2 = new SqlCommandBuilder(sqlDataAdapter2);

    sqlBuilder2.GetInsertCommand();
    sqlBuilder2.GetUpdateCommand();
    sqlBuilder2.GetDeleteCommand();

    dataSet2 = new DataSet();

    sqlDataAdapter2.Fill(dataSet2, "Employee");

    dataGridView2.DataSource = dataSet2.Tables["Employee"];

    for (int i = 0; i < dataGridView2.Rows.Count; i++)
    {
        DataGridViewLinkCell linkCell = new DataGridViewLinkCell();
        dataGridView2[11, i] = linkCell;
    }

    dataGridView2.Columns["category"].Visible = false;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
}

private void ReloadData2()
{
    try
    {
        dataSet2.Tables["Employee"].Clear();

        sqlDataAdapter2.Fill(dataSet2, "Employee");

        dataGridView2.DataSource = dataSet2.Tables["Employee"];

        for (int i = 0; i < dataGridView2.Rows.Count; i++)
        {

```

```

        DataGridViewLinkCell linkCell = new DataGridViewLinkCell();
        dataGridView2[11, i] = linkCell;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

private void dataGridView2_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        if (e.ColumnIndex == 11)
        {
            string task =
dataGridView2.Rows[e.RowIndex].Cells[11].Value.ToString();

            if (task == "Delete")
            {
                if (MessageBox.Show("Видалити цей запис?", "Видалення",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    int rowIndex = e.RowIndex;

                    dataGridView2.Rows.RemoveAt(rowIndex);

                    dataSet2.Tables["Employee"].Rows[rowIndex].Delete();

                    sqlDataAdapter2.Update(dataSet2, "Employee");
                }
            }
            else if (task == "Insert")
            {
                int rowIndex = dataGridView2.Rows.Count - 2;

                string category =
interpreter.AnalyzeCategory_employee(dataGridView2.Rows[rowIndex].Cells["skills"]
.Value.ToString());

                DataRow row = dataSet2.Tables["Employee"].NewRow();

```

```

        row["ID"] = dataGridView2.Rows[rowIndex].Cells["ID"].Value;
        row["full_name"]
dataGridView2.Rows[rowIndex].Cells["full_name"].Value;
        row["birth_date"]
dataGridView2.Rows[rowIndex].Cells["birth_date"].Value;
        row["address"]
dataGridView2.Rows[rowIndex].Cells["address"].Value;
        row["phone"] = dataGridView2.Rows[rowIndex].Cells["phone"].Value;
        row["email"] = dataGridView2.Rows[rowIndex].Cells["email"].Value;
        row["experience_in_years"]
dataGridView2.Rows[rowIndex].Cells["experience_in_years"].Value;
        row["education"]
dataGridView2.Rows[rowIndex].Cells["education"].Value;
        row["skills"] = dataGridView2.Rows[rowIndex].Cells["skills"].Value;
        row["active"] = dataGridView2.Rows[rowIndex].Cells["active"].Value;
        row["category"] = category;

        dataSet2.Tables["Employee"].Rows.Add(row);

dataSet2.Tables["Employee"].Rows.RemoveAt(dataSet2.Tables["Employee"].Rows.Count - 1);

        dataGridView2.Rows.RemoveAt(dataGridView2.Rows.Count - 2);

        dataGridView2.Rows[e.RowIndex].Cells[11].Value = "Delete";

        sqlDataAdapter2.Update(dataSet2, "Employee");

        newRowAdding2 = false;
    }
    else if (task == "Update")
    {
        int r = e.RowIndex;

        string category
interpreter.AnalyzeCategory_employee(dataGridView2.Rows[r].Cells["skills"].Value.ToString());

        dataSet2.Tables["Employee"].Rows[r]["ID"]
dataGridView2.Rows[r].Cells["ID"].Value;
        dataSet2.Tables["Employee"].Rows[r]["full_name"]
dataGridView2.Rows[r].Cells["full_name"].Value;
        dataSet2.Tables["Employee"].Rows[r]["birth_date"]
dataGridView2.Rows[r].Cells["birth_date"].Value;

```

```

        dataSet2.Tables["Employee"].Rows[r]["address"] =
dataGridView2.Rows[r].Cells["address"].Value;
        dataSet2.Tables["Employee"].Rows[r]["phone"] =
dataGridView2.Rows[r].Cells["phone"].Value;
        dataSet2.Tables["Employee"].Rows[r]["email"] =
dataGridView2.Rows[r].Cells["email"].Value;
        dataSet2.Tables["Employee"].Rows[r]["experience_in_years"] =
dataGridView2.Rows[r].Cells["experience_in_years"].Value;
        dataSet2.Tables["Employee"].Rows[r]["education"] =
dataGridView2.Rows[r].Cells["education"].Value;
        dataSet2.Tables["Employee"].Rows[r]["skills"] =
dataGridView2.Rows[r].Cells["skills"].Value;
        dataSet2.Tables["Employee"].Rows[r]["active"] =
dataGridView2.Rows[r].Cells["active"].Value;
        dataSet2.Tables["Employee"].Rows[r]["category"] = category;

        dataGridView2.Rows[e.RowIndex].Cells[11].Value = "Delete";

        sqlDataAdapter2.Update(dataSet2, "Employee");
    }

    ReloadData2();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

private void dataGridView2_UserAddedRow(object sender,
DataGridViewRowEventArgs e)
{
    try
    {
        if (newRowAdding2 == false)
        {
            newRowAdding2 = true;

            int lastRow = dataGridView2.Rows.Count - 2;

            DataGridViewRow row = dataGridView2.Rows[lastRow];

            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

```

```

        dataGridView2[11, lastRow] = linkCell;

        row.Cells["Command"].Value = "Insert";
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
}

private void dataGridView2_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        DataGridView dataGridView = sender as DataGridView;
        if (dataGridView != null)
        {
            dataGridView.AutoResizeRow(e.RowIndex);
        }
        if (newRowAdding2 == false)
        {
            int rowIndex = dataGridView2.SelectedCells[0].RowIndex;

            DataGridViewRow editingRow = dataGridView2.Rows[rowIndex];

            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

            dataGridView2[11, rowIndex] = linkCell;

            editingRow.Cells["Command"].Value = "Update";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

```

```

private void LoadData3()
{
    try
    {
        sqlDataAdapter3 = new SqlDataAdapter("SELECT ID, title, company, phone,
email, job_description, job_requirements, experience_requirements, salary, active,
category FROM VACANCY WHERE active = 1", sqlConnection);

        dataSet3 = new DataSet();
        sqlDataAdapter3.Fill(dataSet3, "Vacancy");

        DataGridViewLinkColumn linkColumn = new DataGridViewLinkColumn
        {
            HeaderText = "Command",
            Name = "Command",
            Text = "Обрати",
            UseColumnTextForLinkValue = true
        };

        dataGridView3.Columns.Add(linkColumn);

        dataGridView3.DataSource = dataSet3.Tables["Vacancy"];
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

```

```

private void ReloadData3()
{
    try
    {
        dataSet3.Tables["Vacancy"].Clear();

        sqlDataAdapter3.Fill(dataSet3, "Vacancy");

        dataGridView3.DataSource = dataSet3.Tables["Vacancy"];
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void dataGridView3_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        if (e.ColumnIndex == 0)
        {
            int need_exp = -1;
            var skillMatchPercentages = new List<(int ID, double Percentage)>();
            if
            (dataGridView3.Rows[e.RowIndex].Cells["experience_requirements"].Value != null
            && dataGridView3.Rows[e.RowIndex].Cells["experience_requirements"].Value !=
            DBNull.Value)
            {
                need_exp =
                int.Parse(dataGridView3.Rows[e.RowIndex].Cells["experience_requirements"].Value.T
                oString());
            }
            skillMatchPercentages =
            interpreter.CalculateSkillMatchPercentage(int.Parse(dataGridView3.Rows[e.RowIndex]
            .Cells["ID"].Value.ToString()),
            dataGridView3.Rows[e.RowIndex].Cells["job_description"].Value.ToString(),
            dataGridView3.Rows[e.RowIndex].Cells["job_requirements"].Value.ToString(),
            need_exp, dataGridView3.Rows[e.RowIndex].Cells["category"].Value.ToString());
            int vacancyID =
            int.Parse(dataGridView3.Rows[e.RowIndex].Cells["ID"].Value.ToString());
            Form2 form2 = new Form2(vacancyID, skillMatchPercentages);
            form2.Show();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void LoadData5()

```

```

    {
        try
        {
            sqlDataAdapter5 = new SqlDataAdapter("SELECT employeeID, vacancyID,
termID, Employee.full_name AS employee_name, Vacancy.title AS vacancy_title,
Vacancy.company AS vacancy_company, date, review_added " +
                "FROM LOGS_OF_DISTRIBUTION " +
                "JOIN Employee ON LOGS_OF_DISTRIBUTION.employeeID =
Employee.ID " +
                "JOIN Vacancy ON LOGS_OF_DISTRIBUTION.vacancyID =
Vacancy.ID"
                , sqlConnection);

            dataSet5 = new DataSet();
            sqlDataAdapter5.Fill(dataSet5, "Logs_of_distribution");

            DataGridViewLinkColumn linkColumn = new DataGridViewLinkColumn
            {
                HeaderText = "Command",
                Name = "Command",
                Text = "Відгук",
                UseColumnTextForLinkValue = true
            };

            dataGridView5.Columns.Add(linkColumn);

            dataGridView5.DataSource = dataSet5.Tables["Logs_of_distribution"];

            dataGridView5.Columns["employeeID"].Visible = false;
            dataGridView5.Columns["vacancyID"].Visible = false;
            dataGridView5.Columns["termID"].Visible = false;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

private void ReloadData5()
{
    try
    {
        dataSet5.Tables["Logs_of_distribution"].Clear();
    }
}

```

```

sqlDataAdapter5.Fill(dataSet5, "Logs_of_distribution");

dataGridView5.DataSource = dataSet5.Tables["Logs_of_distribution"];
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}

private void dataGridView5_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    try
    {
        if (e.ColumnIndex == 0 &&
int.Parse(dataGridView5.Rows[e.RowIndex].Cells["termID"].Value.ToString()) == 0)
        {
            MessageBox.Show("Роботодавець сам вказував потрібний досвід
роботи або не вказував посад чи термінів, що мають на увазі певний досвід
роботи!", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else if (e.ColumnIndex == 0 &&
Convert.ToBoolean(dataGridView5.Rows[e.RowIndex].Cells["review_added"].Value)
== false)
        {
            int vacancyID =
int.Parse(dataGridView5.Rows[e.RowIndex].Cells["vacancyID"].Value.ToString());
            int employeeID =
int.Parse(dataGridView5.Rows[e.RowIndex].Cells["employeeID"].Value.ToString());
            int termID =
int.Parse(dataGridView5.Rows[e.RowIndex].Cells["termID"].Value.ToString());
            string employeeName =
dataGridView5.Rows[e.RowIndex].Cells["employee_name"].Value.ToString();
            Form3 form3 = new Form3(employeeID, vacancyID, employeeName,
termID);
            form3.FormClosed += (obj, args) =>
            {
                ReloadData5();
            };
            form3.Show();
        }
    }
}

```

```

        else if (e.ColumnIndex == 0 &&
Convert.ToBoolean(dataGridView5.Rows[e.RowIndex].Cells["review_added"].Value)
== true)
        {
            MessageBox.Show("Відгук вже був доданий!", "Помилка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private void FilterData(string NameFilter)
{
    try
    {
        string query = "SELECT ID, title, company, phone, email, job_description,
job_requirements, experience_requirements, salary, active, category FROM
VACANCY WHERE company LIKE @Name";

        SqlDataAdapter filterAdapter = new SqlDataAdapter(query, sqlConnection);
        filterAdapter.SelectCommand.Parameters.AddWithValue("@Name", "%" +
NameFilter + "%");

        DataSet filteredDataSet = new DataSet();
        filterAdapter.Fill(filteredDataSet, "Vacancy");

        dataGridView3.DataSource = filteredDataSet.Tables["Vacancy"];
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private void button1_Click(object sender, EventArgs e)
{
    string NameFilter = textBox1.Text;
    FilterData(NameFilter);
}
}

```

Interpreter.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomaAPP
{
    internal class Interpreter
    {
        private Dictionary<string, (string skills, int experience, string category, int ID)>
knowledgeData;
        private Dictionary<string, (string terms, string skills)> categoryData;
        public Interpreter()
        {
            knowledgeData = LoadKnowledge_data();
            categoryData = LoadCategory_data();
        }

        public Form3 Form3
        {
            get => default;
            set
            {
            }
        }

        public Form1 Form1
        {
            get => default;
            set
            {
            }
        }

        static int CountOccurrences(string text, string phrase)
        {

```

```

        return Regex.Matches(text, @$@"\b{Regex.Escape(phrase)}\b",
RegexOptions.IgnoreCase).Count;
    }
    public void ChangeKnowledge_data(int ID)
    {
        int judge_weight;
        int experience_in_years;
        SqlConnection sqlConnection = null;
        sqlConnection = new SqlConnection(@"Data Source=DESKTOP-
RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True");
        using (sqlConnection)
        {
            if (sqlConnection.State == ConnectionState.Closed)
            {
                sqlConnection.Open();
            }

            string query = $"SELECT judge_weight FROM Knowledge_data WHERE ID
= {ID}";
            using (SqlCommand command = new SqlCommand(query, sqlConnection))
            {
                judge_weight = Convert.ToInt32(command.ExecuteScalar());
            }

            query = $"SELECT experience_in_years FROM Knowledge_data WHERE
ID = {ID}";
            using (SqlCommand command = new SqlCommand(query, sqlConnection))
            {
                experience_in_years = Convert.ToInt32(command.ExecuteScalar());
            }
            experience_in_years++;

            string updateQuery = $"UPDATE Knowledge_data " +
                $"SET judge_weight = " +
                $" CASE " +
                $"     WHEN DATEDIFF(YEAR, date_last_update, GETDATE())
>= 2 THEN 0 " +
                $"     ELSE judge_weight " +
                $" END, " +
                $" date_last_update = " +
                $" CASE " +
                $"     WHEN date_last_update IS NULL OR DATEDIFF(YEAR,
date_last_update, GETDATE()) >= 2 THEN GETDATE() " +
                $"     ELSE date_last_update " +

```

```

        $"    END " +
        $"WHERE ID = {ID}";

        using (SqlCommand updateCommand = new SqlCommand(updateQuery,
sqlConnection))
        {
            updateCommand.ExecuteNonQuery();
        }

        if (judge_weight == -5)
        {
            query = $"SELECT experience_in_years FROM Knowledge_data WHERE
ID = {ID}";
            using (SqlCommand command = new SqlCommand(query,
sqlConnection))
            {
                experience_in_years = Convert.ToInt32(command.ExecuteScalar());
            }
            experience_in_years++;

            updateQuery = $"UPDATE Knowledge_data SET judge_weight = {0}
WHERE ID = {ID}" +
                $"UPDATE Knowledge_data SET experience_in_years =
{experience_in_years} WHERE ID = {ID}";

            using (SqlCommand updateCommand = new SqlCommand(updateQuery,
sqlConnection))
            {
                updateCommand.ExecuteNonQuery();
            }
        }
        LoadKnowledge_data();
    }
    private Dictionary<string, (string skills, int experience, string category, int ID)>
LoadKnowledge_data()
    {
        var knowledge_data = new Dictionary<string, (string skills, int experience,
string category, int ID)>();

        string query = "SELECT ID, term, skills, experience_in_years, category FROM
Knowledge_data";

```

```

        using (SqlConnection connection = new SqlConnection(@"Data
Source=DESKTOP-RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True"))
        {
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string term = reader["term"].ToString();
                string skills = reader["skills"].ToString();
                int experience = Convert.IsDBNull(reader["experience_in_years"]) ? -1 :
Convert.ToInt32(reader["experience_in_years"]);
                string category = reader["category"].ToString();
                int ID = int.Parse(reader["ID"].ToString());

                knowledge_data.Add(term, (skills, experience, category, ID));
            }
        }

        return knowledge_data;
    }
    private Dictionary<string, (string terms, string skills)> LoadCategory_data()
    {
        var category_data = new Dictionary<string, (string terms, string skills)>();

        string query = "SELECT category, terms, skills FROM Category";

        using (SqlConnection connection = new SqlConnection(@"Data
Source=DESKTOP-RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True"))
        {
            SqlCommand command = new SqlCommand(query, connection);
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string category = reader["Category"].ToString();
                string terms = reader["Terms"].ToString();
                string skills = reader["Skills"].ToString();

                category_data.Add(category, (terms, skills));
            }
        }
    }

```

```

        return category_data;
    }

    private List<string> AnalyzeJob_skills(string jobDescription, string
jobRequirement, string category)
    {
        var requiredSkills = new HashSet<string>();

        string fullText = jobDescription + " " + jobRequirement;

        string[] allSkillsArray = categoryData[category].skills.Split(new[] { ';' },
StringSplitOptions.RemoveEmptyEntries);

        foreach (string skill in allSkillsArray)
        {
            if (fullText.Contains(skill))
            {
                requiredSkills.Add(skill.TrimStart());
            }
        }

        string[] words = fullText.Split(new[] { ' ', ',', '!', ':', ';', ':' },
StringSplitOptions.RemoveEmptyEntries);

        for (int i = 0; i < words.Length; i++)
        {
            string term = words[i].Trim();

            if (knowledgeData.ContainsKey(term))
            {
                string skills = knowledgeData[term].skills;

                string[] termSkills = skills.Split(new[] { ',', '!', ':', ';' },
StringSplitOptions.RemoveEmptyEntries);

                foreach (var skill in termSkills)
                {
                    requiredSkills.Add(skill.TrimStart());
                }
            }

            if (i < words.Length - 1)
            {
                term = string.Join(" ", words[i].Trim(), words[i + 1].Trim());
            }
        }
    }

```

```

        if (knowledgeData.ContainsKey(term))
        {
            string skills = knowledgeData[term].skills;

            string[] termSkills = skills.Split(new[] { ',', '!', ';', ':' },
StringSplitOptions.RemoveEmptyEntries);

            foreach (var skill in termSkills)
            {
                requiredSkills.Add(skill.TrimStart());
            }
        }
    }
    return requiredSkills.ToList();
}

private int AnalyzeJob_experience(int id, string jobDescription, string
jobRequirement, string category)
{
    int req_experience = 0;

    int experience_requirements_termID = -1;

    string fullText = " " + jobDescription + " " + jobRequirement;

    string[] words = fullText.Split(new[] { ' ', ',', '!', ';', ':' },
StringSplitOptions.RemoveEmptyEntries);

    for (int i = 0; i < words.Length; i++)
    {
        string term = words[i].Trim();

        if (knowledgeData.ContainsKey(term))
        {
            if (knowledgeData[term].experience > req_experience)
            {
                req_experience = knowledgeData[term].experience;
                experience_requirements_termID = knowledgeData[term].ID;
            }
        }

        if (i < words.Length - 1)
        {
            term = string.Join(" ", words[i].Trim(), words[i + 1].Trim());

```

```

        if (knowledgeData.ContainsKey(term))
        {
            if (knowledgeData[term].experience > req_experience)
            {
                req_experience = knowledgeData[term].experience;
                experience_requirements_termID = knowledgeData[term].ID;
            }
        }
    }
}
if(experience_requirements_termID != -1)
{
    using (SqlConnection connection = new SqlConnection(@"Data
Source=DESKTOP-RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True"))
    {
        connection.Open();

        string updateQuery = $"UPDATE Vacancy SET
experience_requirements_termID = {experience_requirements_termID} WHERE ID =
{id}";

        SqlCommand updateCommand = new SqlCommand(updateQuery,
connection);

        updateCommand.ExecuteNonQuery();
    }
}
else
{
    req_experience = 0;
}
return req_experience;
}
private List<string> AnalyzeSkills(string Skills, string category)
{
    var requiredSkills = new HashSet<string>();

    string fullText = " " + Skills;

    string[] allSkillsArray = categoryData[category].skills.Split(new[] { ';' },
StringSplitOptions.RemoveEmptyEntries);

    foreach (string skill in allSkillsArray)
    {

```

```

        if (fullText.Contains(skill))
        {
            requiredSkills.Add(skill.TrimStart());
        }
    }

    string[] words = fullText.Split(new[] { ' ', ',', '!', '!', '!', '!', '!' },
StringSplitOptions.RemoveEmptyEntries);

    for (int i = 0; i < words.Length; i++)
    {
        string term = words[i].Trim();

        if (knowledgeData.ContainsKey(term))
        {
            string skills = knowledgeData[term].skills;

            string[] termSkills = skills.Split(new[] { '!', '!', '!', '!' },
StringSplitOptions.RemoveEmptyEntries);

            foreach (var skill in termSkills)
            {
                requiredSkills.Add(skill.TrimStart());
            }
        }

        if (i < words.Length - 1)
        {
            term = string.Join(" ", words[i].Trim(), words[i + 1].Trim());

            if (knowledgeData.ContainsKey(term))
            {
                string skills = knowledgeData[term].skills;

                string[] termSkills = skills.Split(new[] { '!', '!', '!', '!' },
StringSplitOptions.RemoveEmptyEntries);

                foreach (var skill in termSkills)
                {
                    requiredSkills.Add(skill.TrimStart());
                }
            }
        }
    }

    return requiredSkills.ToList();

```

```

    }

    public List<(int ID, double Percentage)> CalculateSkillMatchPercentage(int id,
string jobDescription, string jobRequirements, int experience_in_years, string category)
    {
        DataTable dataTable = new DataTable();
        string query = "SELECT ID, skills, active, experience_in_years, category
FROM Employee";
        SqlConnection connection = new SqlConnection(@"Data Source=DESKTOP-
RB1SDJS;Initial Catalog=DiplomaDB;Integrated Security=True");
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            connection.Open();
            using (SqlDataReader reader = command.ExecuteReader())
            {
                dataTable.Load(reader);
            }
        }
        DataTable employeeTable = dataTable;

        var skillMatchPercentages = new List<(int ID, double Percentage)>();

        List<string>      requiredSkills      =      AnalyzeJob_skills(jobDescription,
jobRequirements, category);

        if (experience_in_years == -1)
        {
            experience_in_years      =      AnalyzeJob_experience(id,      jobDescription,
jobRequirements, category);
        }

        foreach (DataRow employeeRow in employeeTable.Rows)
        {
            if      (employeeRow["category"].ToString()      ==      category      &&
(bool)employeeRow["active"] == true)
            {
                if      (int.Parse(employeeRow["experience_in_years"].ToString())      >=
experience_in_years)
                {
                    string employeeSkillsString = employeeRow["skills"].ToString();
                    List<string> employeeSkills = AnalyzeSkills(employeeSkillsString,
category);
                    int      matchingSkillsCount      =
requiredSkills.Intersect(employeeSkills).Count();

```

```

        double matchPercentage = (double)matchingSkillsCount /
requiredSkills.Count * 100;
        if (matchPercentage > 60)
        {
            skillMatchPercentages.Add((Convert.ToInt32(employeeRow["ID"]),
matchPercentage));
        }
    }
}

return skillMatchPercentages.OrderByDescending(x =>
x.Percentage).Take(10).ToList();
}

```

```

public string AnalyzeCategory_vacancy(string jobDescription, string
JobRequirements)
{
    Dictionary<string, (int termCount, int skillCount)> result = new
Dictionary<string, (int termCount, int skillCount)>();

    foreach (var category in categoryData)
    {
        string categoryName = category.Key;
        string[] terms = category.Value.terms.Split(new[] { ',' },
StringSplitOptions.RemoveEmptyEntries);
        string[] skills = category.Value.skills.Split(new[] { ',' },
StringSplitOptions.RemoveEmptyEntries);

        int termCount = 0;
        int skillCount = 0;

        foreach (var term in terms)
        {
            termCount += CountOccurrences(jobDescription, term.Trim()) +
CountOccurrences(JobRequirements, term.Trim());
        }

        foreach (var skill in skills)
        {
            skillCount += CountOccurrences(jobDescription, skill.Trim()) +
CountOccurrences(JobRequirements, skill.Trim());
        }
    }
}

```

```

        result[categoryName] = (termCount, skillCount);
    }

    string maxCategory = null;
    int maxScore = 0;

    foreach (var category in result)
    {
        int totalScore = category.Value.termCount * 3 + category.Value.skillCount;
        if (totalScore > maxScore)
        {
            maxScore = totalScore;
            maxCategory = category.Key;
        }
    }

    return maxCategory;
}

public string AnalyzeCategory_employee(string education)
{
    Dictionary<string, (int termCount, int skillCount)> result = new
Dictionary<string, (int termCount, int skillCount)>();

    foreach (var category in categoryData)
    {
        string categoryName = category.Key;
        string[] terms = category.Value.terms.Split(new[] { ',' },
StringSplitOptions.RemoveEmptyEntries);
        string[] skills = category.Value.skills.Split(new[] { ',' },
StringSplitOptions.RemoveEmptyEntries);

        int termCount = 0;
        int skillCount = 0;

        foreach (var term in terms)
        {
            termCount += CountOccurrences(education, term.Trim());
        }

        foreach (var skill in skills)
        {
            skillCount += CountOccurrences(education, skill.Trim());
        }
    }
}

```

```
    result[categoryName] = (termCount, skillCount);
}

string maxCategory = null;
int maxScore = 0;

foreach (var category in result)
{
    int totalScore = category.Value.termCount * 3 + category.Value.skillCount;
    if (totalScore > maxScore)
    {
        maxScore = totalScore;
        maxCategory = category.Key;
    }
}

return maxCategory;
}
}
```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів
Дипломна кваліфікаційна робота бакалавра

Система інтелектуального розподілу робочих місць для біржі труда

Виконав:

ст. групи КНТ-130

Олексій ПАВЛЕЦЬ

Керівник:

к.т.н., доцент

Сергій СЕРДЮК

Рисунок Б.1 – Слайд № 1

Об'єкт, предмет та мета роботи

- ♦ Об'єкт дослідження – процес інтелектуального розподілу робочих місць;
- ♦ Предмет дослідження – комп'ютерні програми для інтелектуального розподілу робочих місць;
- ♦ Мета роботи – розробка програмного забезпечення для інтелектуального розподілу робочих місць на біржі труда з метою автоматизації процесу пошуку роботи для працівників біржи труда.

2

Рисунок Б.2 – Слайд № 2

Задачі, що вирішуються

Для досягнення поставленої мети роботи необхідно розв'язати такі задачі:

- ◆ проаналізувати діяльність інспектора з працевлаштування на біржі труда;
- ◆ дослідити існуючі аналоги систем розподілу робочих місць;
- ◆ здійснити проектування програмного забезпечення для системи інтелектуального розподілу робочих місць для біржі труда;
- ◆ розробити систему правил для інтелектуального розподілу робітників на вакансії;
- ◆ розробити базу даних системи інтелектуального розподілу робочих місць для біржі труда;
- ◆ створити програмне забезпечення для системи інтелектуального розподілу робочих місць для біржі труда;
- ◆ зробити тестування розробленого програмного забезпечення для системи інтелектуального розподілу робочих місць для біржі труда.

3

Рисунок Б.3 – Слайд № 3

Порівняння існуючих аналогів

Критерій порівняння	LinkedIn Talent Solutions	Greenhouse	SAP SuccessFactors	Мій застосунок
Інтелектуальність системи	-	-	-	+
Висока швидкість роботи	+	+	-	+
Низька ціна	-	-	-	+
Зручність та сучасність	+	+	-	+
Можливість інтеграції з іншими платформами	+	+	+	-
Надання спілкування між роботодавцем та робітником	+	+	-	-

4

Рисунок Б.4 – Слайд № 4

Порівняння середовищ розробки

Критерій	Microsoft Visual Studio	JetBrains IntelliJ IDEA
Підтримка .NET	Відмінна	Обмежена
Підтримка Java	Обмежена	Відмінна
Інструменти налагодження	Дуже хороші	Дуже хороші
Інтерфейс	Інтуїтивно зрозумілий	Інтуїтивно зрозумілий
Розширюваність	Дуже висока	Дуже висока
Вартість	Безкоштовно	Платно

5

Рисунок Б.5 – Слайд № 5

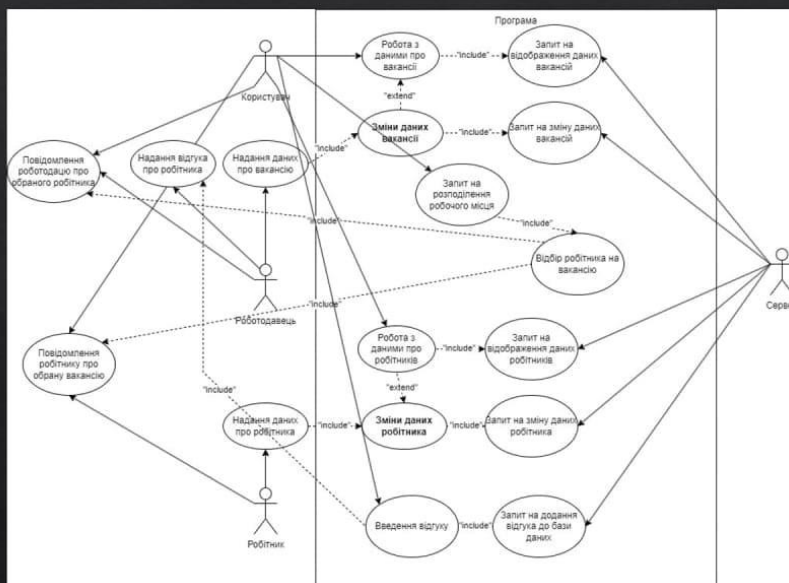
Вибір мови програмування

Критерій	C#	Java	Python
Широке використання	Так	Так	Так
Інтеграція з .NET	Висока	Немає	Немає
Крос-платформенність	Обмежена	Так	Так
Простота вивчення	Середня	Висока	Висока
Безпека	Висока	Висока	Низька
Швидкодія	Висока	Середня	Низька

6

Рисунок Б.6 – Слайд № 6

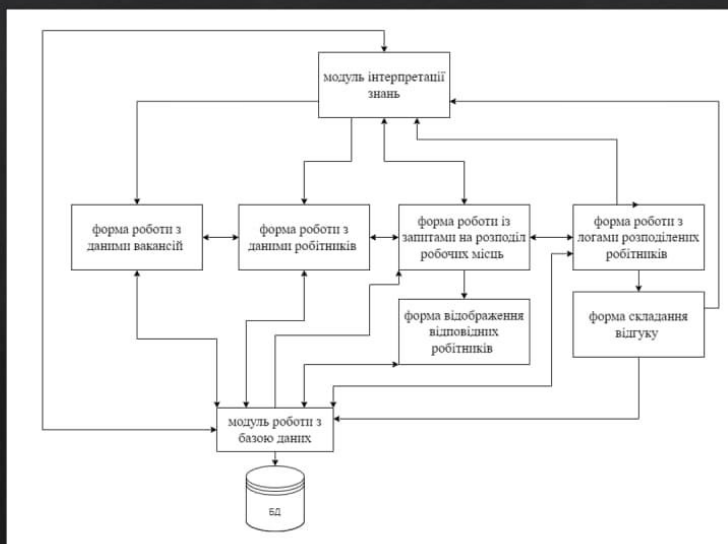
Прецедент «Розподіл робочих місць»:



7

Рисунок Б.7 – Слайд № 7

Схема структури розробленої програми



8

Рисунок Б.8 – Слайд № 8

Розробка системи правил

Продукційні правила в системі формалізують логіку ухвалення. Нижче наведено декілька прикладів правил системи.

П1. ЯКЩО «вакансія = містить Java junior» І «вакансія = НЕ містить вимог до досвіду роботи» ТО «вимоги до навичок = додати Java Basics, OOP Principles, Data Structures» І «вимоги до досвіду роботи = 2 роки».

П2. ЯКЩО «вакансія = містить Frontend developer» І «вакансія = НЕ містить вимог до досвіду роботи» ТО «вимоги до навичок = додати HTML, CSS, JavaScript» І «вимоги до досвіду роботи = 3 роки».

П3. ЯКЩО «вакансія = містить Python developer» І «вакансія = містить вимоги до досвіду роботи» ТО «вимоги до навичок = додати Python, Flask, SQL».

П10. ЯКЩО «негативних відгуків на термін General Practitioner на 5 більше ніж позитивних відгуків» І «оновлення терміну General Practitioner < 2 роки тому» ТО «вимоги до досвіду роботи General Practitioner = додати 1 рік» І «оновлення терміну General Practitioner = сьогодні».

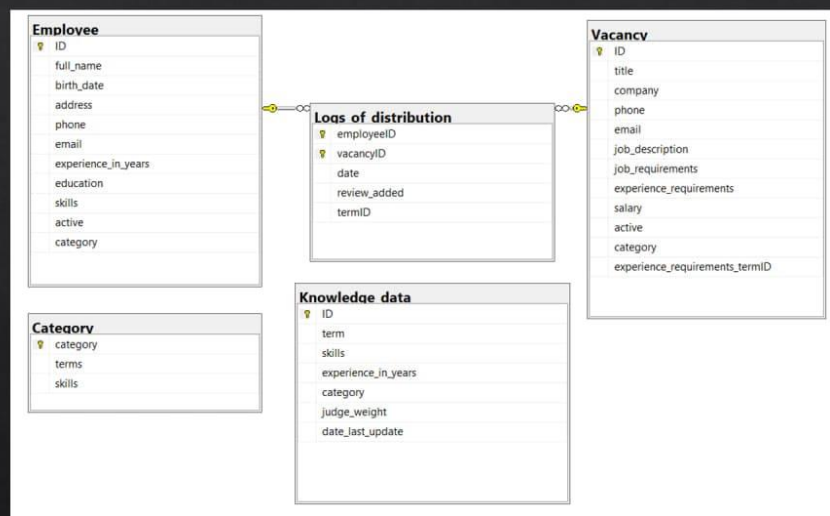
П11. ЯКЩО «негативних відгуків на термін Nurse на 5 більше ніж позитивних відгуків» І «оновлення терміну General Practitioner >= 2 роки тому» ТО «відгуків на термін Nurse = 0» І «оновлення терміну Nurse = сьогодні».

Ці правила дозволять системі ефективно підбирати працівників на основі аналізу наявних даних про вакансії та кандидатів, а також підтримувати актуальність знань у базі даних. Продукційні правила є основним інструментом прийняття рішень в системі інтелектуального розподілу робочих місць.

9

Рисунок Б.9 – Слайд № 9

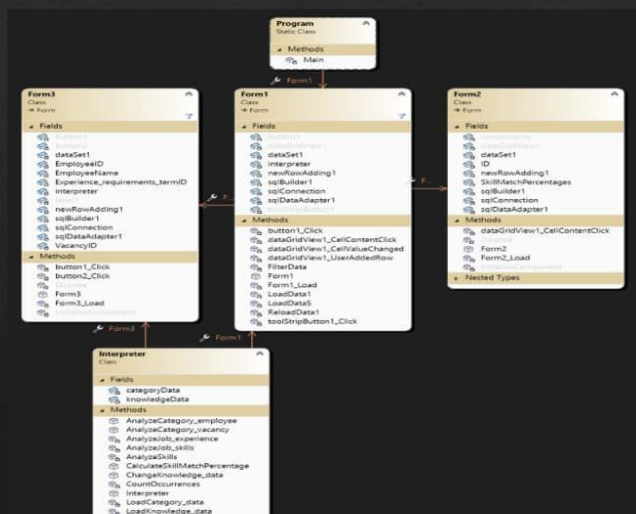
Схема структури бази даних



10

Рисунок Б.10 – Слайд № 10

Діаграма класів програми



11

Рисунок Б.11 – Слайд № 11

Інтерфейс основного вікна для роботи з програмою

Система інтелектуального розподілу робочих місць для біржі труда

Оновити

База даних вакансій База даних робітників Завіт на розподіл робочих місць Пути розподілення робочих місць

ID	title	company	phone	email	job_description	job_requirements	experience_require	salary	active
30	Software	Tech Solutions Inc.	+1 (555) 11111	hr@techsolutions.co	We are looking for a Software Engineer with 5+ years of experience in Java and Spring.	Proficient in HTML, CSS, JavaScript	5	\$5000.0000	☒
31	Data Analyst	Data Insights Corp.	+1 (555) 22222	careers@datainsight.com	We are seeking a Data Analyst to join our team and analyze customer behavior.	Experienced in Data Mining, SQL, and Tableau	4	\$5000.0000	☒
32	Network Security	SecureNet Ltd.	+1 (555) 33333	jobs@securenet.com	We need a Network Security Specialist to protect our infrastructure.	Skilled in Network Security, Firewalls, and Intrusion Detection	5	\$5000.0000	☒
33	Registered Nurse	HealthCare Partners	+1 (555) 44444	apply@healthcare.com	Hiring a Registered Nurse for our hospital's medical ward.	Proficient in Medical Procedures and Patient Care	2	\$5000.0000	☒
34	Financial Analyst	FinCorp	+1 (555) 55555	hr@fincorp.com	We are seeking a Financial Analyst to assist in investment decisions.	Skilled in Financial Modeling, Excel, and Market Analysis	3	\$5000.0000	☒
35	Physical Therapist	RehabWorks	+1 (555) 66666	careers@rehabworks.com	Looking for a Physical Therapist to join our rehabilitation center.	Expertise in Physical Therapy and Patient Assessment	4	\$5000.0000	☒
36	High School Teacher	EducationFirst	+1 (555) 77777	jobs@educationfirst.com	We need a High School Teacher for our Mathematics department.	Proficient in Lesson Planning and Classroom Management	2	\$5000.0000	☒
37	Investment Analyst	WealthBuilders	+1 (555) 88888	careers@wealthbuilders.com	Seeking an Investment Analyst to evaluate market opportunities.	Experienced in Financial Markets and Investment Strategies	5	\$5000.0000	☒
39	Cybersecurity	CyberGuard Inc.	+1 (555) 99999	hr@cyberguard.com	We are looking for a Cybersecurity Specialist to protect our data.	Proficient in Network Security, Encryption, and Incident Response	4	\$5000.0000	☒
40	Marketing Manager	MarketLeaders Co.	+1 (555) 00000	jobs@marketleaders.com	We are hiring a Marketing Manager to lead our digital marketing efforts.	Skilled in Strategic Marketing, Social Media, and Analytics	6	\$5000.0000	☒
42	Test Java junior	testtestcoo	test	test@testtest.com	We are looking for a Test Java junior to join our team.	Proficient in Java, JUnit, and Selenium	1	\$5000.0000	☒

12

Рисунок Б.12 – Слайд № 12

Висновки

В результаті виконання дипломної роботи було виконано такі задачі:

- проаналізовано діяльність інспектора з працевлаштування на біржі труда;
- досліджено існуючі аналоги систем розподілу робочих місць;
- спроектовано програмне забезпечення для системи інтелектуального розподілу робочих місць для біржі труда;
- розроблено систему продукційних правил для інтелектуального розподілу робітників на вакансії;
- розроблено базу даних системи;
- створено програмне забезпечення для системи інтелектуального розподілу робочих місць для біржі труда;
- зроблено тестування розробленого програмного забезпечення.

Розроблена система інтелектуального розподілу робочих місць може практично використовуватися в якості інструменту для покращення роботи біржі труда для забезпечення оптимального використання людських ресурсів. Інтелектуальні можливості системи дозволяють їй адаптуватися до змін вимог ринку праці та автоматизувати складні процеси розподілу робочих місць.

13

Рисунок Б.13 – Слайд № 13