МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК З ПІДБОРУ ВАКАНСІЙ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010210

Виконала студентка 4-го курсу, групи 402
_____К. А. Жиглова
«19» червня 2024 р.

Керівник: д-р техн. наук, доцент
_____І. О. Калініна
«19» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Жигловій Катерині Анатоліївні.

1. Тема кваліфікаційної роботи «Вебзастосунок з підбору вакансій».

Керівник роботи Калініна Ірина Олександрівна, д-р техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271.

2. Строк представлення кваліфікаційної роботи студенткою «19» червня 2024 р.

3. Вхідні (початкові) дані до роботи: статистика ринку праці, вимоги користувачів існуючих вебсервісів, технічні вимоги та юридичні аспекти захисту даних.

Очікуваний результат: веборієнтоване програмне забезпечення для автоматизованого підбору вакансій.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- аналіз ринку праці та визначення потреби у вебзастосунку підбору вакансій;
- огляд існуючих рішень для підбору вакансій та їх недоліки;
- обґрунтування технологій розробки вебзастосунку;
- технічне завдання на розробку вебзастосунку;

– опис архітектури вебзастосунку;

– розробка структури бази даних та проектування інтерфейсу вебзастосунку;

– реалізація вебзастосунку та його тестування.

5. Перелік графічного матеріалу: 10 рисунків, 3 таблиці, презентація.

6. Завдання до спеціальної частини: «Охорона праці при роботі з комп’ютером»

7. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи _д-р техн. наук, доцент Калініна І. О._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання __Жиглова К. А._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_14_» ____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок з підбору вакансій

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз поточного стану ринку праці, огляд існуючих рішень, обґрунтування вибору технологій, проектування та розробка вебзастосунку, тестування	13.05.2024	22.05.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	Виконано

Розробила студентка Жиглова К. А. _____
(прізвище, ім'я, по батькові студента) *(підпис)*

Керівник роботи д-р техн. наук, доцент Калініна І. О. _____
(прізвище, ім'я, по батькові студента) *(підпис)*

«29» січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра студентки
402 групи ЧНУ ім. Петра Могили

Жиглової Катерини Анатоліївни

на тему: **«ВЕБЗАСТОСУНОК З ПІДБОРУ ВАКАНСІЙ»**

Актуальність теми полягає в необхідності впровадження сучасних інформаційних технологій для автоматизації та оптимізації процесу підбору вакансій. Це дозволить підвищити ефективність взаємодії між роботодавцями та шукачами роботи, персоналізувати рекомендації вакансій та покращити досвід користувачів.

Об’єктом роботи є процес організації ефективної взаємодії між роботодавцями та шукачами роботи на основі інформаційних технологій.

Предметом роботи є методи та засоби проектування й розробки веборієнтованого програмного забезпечення для автоматизованого підбору вакансій.

Метою даної роботи є проектування та розробка веборієнтованої інформаційної системи для ефективного підбору вакансій, що задовольнятиме потреби як працівників, так і роботодавців.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, трьох розділів, висновків та додатку.

У першому розділі проводиться аналіз ринку праці та огляд існуючих рішень, визначення потреби у вебзастосунку для підбору вакансій та вимог до нього.

У другому розділі описано проектування вебзастосунку, включаючи архітектуру, структуру бази даних та дизайн інтерфейсу користувача.

У третьому розділі описано процес розробки вебзастосунку, використані технології та інструменти, реалізацію серверної та клієнтської частин, а також тестування та розгортання системи.

В результаті розроблено спеціалізований вебзастосунок з підбору вакансій, орієнтований на роботодавців та шукачів роботи.

Кваліфікаційна робота бакалавра містить 90 сторінок, 3 таблиці, 10 рисунків, 25 джерел та 1 додаток.

Ключові слова: підбір вакансій, вебтехнології, інформаційні системи, автоматизація, персоналізація, досвід користувача.

ABSTRACT

to the bachelor's qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Zhyhlova Kateryna

«WEB APPLICATION FOR SELECTION OF VACANCIES»

The relevance of the topic lies in the need to introduce modern information technologies to automate and optimize the process of selecting vacancies. This will increase the efficiency of interaction between employers and job seekers, personalize job recommendations, and improve user experience.

The object of the research is the process of organizing effective interaction between employers and job seekers on the basis of information technology, including the collection and analysis of data on vacancies and resumes, comparison of employers' requirements with candidates' qualifications.

The subject of the research is the methods and tools for designing and developing web-based software for automated job selection.

The purpose of this work is to design and develop a web-based information system for effective selection of vacancies that will meet the needs of both employees and employers.

The work consists of a professional section and a special part on labor protection. The explanatory note consists of an introduction, three sections, conclusions and an appendix.

The first section analyzes the labor market and reviews existing solutions, determines the need for a web application for the selection of vacancies and the requirements for it.

The second section describes the design of a web application, including architecture, database structure, and user interface design.

The third section describes the process of developing a web application, the technologies and tools used, the implementation of the server and client parts, as well as testing and deployment of the system.

As a result, a specialized web application for the selection of vacancies has been developed, aimed at employers and job seekers.

The bachelor’s thesis contains 90 pages, 3 tables, 10 pictures, 25 sources and 1 appendix.

Keywords: job selection, web technologies, information systems, automation, personalization, user experience.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МОП – Міжнародна організація праці

СУБД – система управління базами даних

ФОП – фізична особа-підприємець

AJAX – Asynchronous JavaScript and XML

CSS – Cascading Style Sheets

DB – Database

GDPR – General Data Protection Regulation

HR – Human Resources

HRM – Human Resource Management

HTML – Hypertext Markup Language

JS – JavaScript

MVC – Model-View-Controller

PHP – PHP: Hypertext Preprocessor

REST – Representational State Transfer

SQL – Structured Query Language

XML – eXtensible Markup Language

ВСТУП

На сьогодні ринок праці в Україні перебуває у стадії активного розвитку, це свідчить про **актуальність** даної теми. Кількість вакансій щороку зростає, проте існує гостра нестача кваліфікованих кадрів у багатьох сферах. Одночасно спостерігається високий рівень безробіття серед економічно активного населення. Така ситуація свідчить про серйозні диспропорції між попитом і пропозицією робочої сили.

Однією з ключових причин такого дисбалансу є відсутність ефективних інструментів для пошуку роботи та підбору персоналу. Традиційні методи, такі як оголошення в ЗМІ чи на дошках вакансій, вже не забезпечують потрібної оперативності та зручності. Сучасні інтернет-платформи для пошуку роботи також мають певні недоліки – відсутність персоналізації, обмежений функціонал, складність інтеграції з базами даних кандидатів та вакансій.

У зв’язку з цим гостро постає необхідність створення спеціалізованого вебзастосунку, орієнтованого одночасно на роботодавців та шукачів роботи.

Метою даної роботи є проектування та розробка веборієнтованої інформаційної системи для ефективного підбору вакансій, що задовольнятиме потреби як працівників, так і роботодавців.

Для досягнення поставленої мети в роботі передбачено вирішення наступних **завдань**:

1) дослідити поточний стан ринку праці та проаналізувати існуючі рішення у сфері підбору вакансій, виявити їх недоліки;

2) сформулювати функціональні та нефункціональні вимоги до вебзастосунку підбору вакансій на основі проведеного аналізу;

3) обґрунтувати вибір технологій та засобів реалізації для проектування та розробки системи;

4) розробити архітектуру та загальний алгоритм функціонування системи, спроектувати базу даних;

5) реалізувати програмний продукт на основі обраних технологій та забезпечити його функціонування;

6) провести тестування розробленого вебзастосунку, виконати необхідні доопрацювання та оптимізації.

Об’єктом роботи є процес організації ефективної взаємодії між роботодавцями та шукачами роботи на основі інформаційних технологій.

Предметом роботи є методи та засоби проектування й розробки веборієнтованого програмного забезпечення для автоматизованого підбору вакансій.

Практичне значення роботи визначається можливістю впровадження створеного вебзастосунку в діяльність компаній, що надають послуги з працевлаштування, а також його використанням безпосередньо шукачами роботи та роботодавцями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ

1.1 Аналіз ринку праці та визначення потреби в вебзастосунку підбору вакансій

Ринок праці є одним з ключових елементів соціально-економічної системи країни, який відображає стан і динаміку попиту і пропозиції робочої сили, а також умови їх взаємодії. Ринок праці в Україні характеризується наступними особливостями [1]:

– зниження рівня зайнятості: за даними Державної служби статистики, кількість зайнятих осіб у 2023 році склала 15,9 млн. осіб, що на 1,2% менше, ніж у 2022 році. За оцінками Міжнародного валютного фонду, рівень зайнятості в Україні у 2023 році становив 54,6%, що є одним з найнижчих серед країн Європи [2];

– зростання рівня безробіття: за даними Державної служби зайнятості, кількість безробітних осіб у 2023 році склала 1,6 млн. осіб, що на 8,1% більше, ніж у 2022 році. Рівень безробіття за методологією МОП у 2023 році становив 9,4%, що на 0,5 п.п. вище, ніж у 2022 році [3]. Особливо високий рівень безробіття спостерігається серед молоді (15-24 років) – 19,5%, а також серед внутрішньо переміщених осіб – 21,7% [4];

– професійно-кваліфікаційний дисбаланс: ринок праці в Україні переживає дефіцит кваліфікованих працівників, особливо в галузях, пов’язаних з інноваціями, інформаційними технологіями, інженерією, медициною, освітою тощо. За даними Державної служби зайнятості, у 2023 році найбільш затребуваними професіями були: програмісти, інженери, вчителі, лікарі, фармацевти, бухгалтери, юристи, продавці, касири, водії тощо [5]. За даними опитування Федерації роботодавців України, 58% роботодавців відчувають нестачу кваліфікованих працівників, а 42% вважають, що освітня система не забезпечує потреби ринку праці;

– неоднорідність кон’юнктури на ринку праці: ринок праці в Україні має значні регіональні та галузеві відмінності. За даними Державної служби

статистики, у 2023 році найвищий рівень зайнятості спостерігався в Києві (67,8%), а найнижчий в Луганській області (40,9%). Найвищий рівень безробіття зафіксовано в Донецькій області (18,7%), а найнижчий – в Київській області (5,4%) [3]. За галузевою структурою зайнятості, у 2023 році найбільшу частку мали сільське господарство (14,9%), оптова та роздрібна торгівля (14,8%), промисловість (13,4%), освіта (9,9%) та будівництво (7,9%).

Порівняння показників ринку праці в Україні наведено у табл. 1.1.

Таблиця 1.1 – Порівняння основних показників ринку праці в Україні за 2022 та 2023 роки

Показник	**2022 рік**	**2023 рік**	**Зміна**
Кількість зайнятих осіб, млн.	16,1	15,9	-1,2%
Рівень зайнятості, %	55,1	54,6	-0,5 п.п.
Кількість безробітних осіб, млн.	1,5	1,6	+8,1%
Рівень безробіття, %	8,9	9,4	+0,5 п.п.
Найбільш затребувані професії	програмісти, інженери, вчителі, лікарі, фармацевти, бухгалтери, юристи, продавці, касири, водії	програмісти, інженери, вчителі, лікарі, фармацевти, бухгалтери, юристи, продавці, касири, водії	без змін

На основі аналізу ринку праці можна зробити висновок, що існує потреба у вебзастосунку підбору вакансій, який би сприяв:

– підвищенню рівня зайнятості: вебзастосунок з підбору вакансій дозволить безробітним особам швидше і ефективніше знайти роботу, яка відповідає їх кваліфікації, інтересам та очікуванням. Вебзастосунок також допоможе зайнятим особам, які шукають кращі умови праці, змінити своє місце

роботи або професію;

– зменшенню професійно-кваліфікаційного дисбалансу: вебзастосунок з підбору вакансій дозволить роботодавцям залучати кваліфікованих працівників, які відповідають вимогам ринку праці та потребам підприємства. Вебзастосунок також допоможе працівникам підвищувати свій рівень навичок та компетентності, отримуючи інформацію про можливості навчання, стажування, сертифікації тощо;

– забезпеченню доступності та прозорості інформації про вакансії: вебзастосунок з підбору вакансій дозволить працівникам та роботодавцям отримувати актуальну та достовірну інформацію про вільні робочі місця, умови праці, вимоги до кандидатів, рівень оплати праці тощо. Вебзастосунок також допоможе аналізувати та прогнозувати тенденції на ринку праці, виявляти потреби в нових професіях та навичках;

– сприянню мобільності та адаптації працівників: вебзастосунок з підбору вакансій дозволить працівникам шукати роботу не тільки в своєму регіоні, але й в інших частинах України та за кордоном, використовуючи можливості дистанційної роботи, тимчасової роботи, фрілансу тощо. Вебзастосунок також допоможе працівникам адаптуватися до змін на ринку праці, надавати психологічну та соціальну підтримку, забезпечувати захист прав та інтересів працівників.

Поширеним типом вебзастосунків для пошуку роботи є сайти кадрових агентств, які надають послуги з підбору персоналу для компаній-замовників. Прикладами таких агентств є Ancor, HR Alliance, Unity Group та інші.

Перевагами сайтів кадрових агентств є:

– доступ до великої бази кандидатів, які пройшли попередню перевірку і оцінку;

– професійний підхід та підбір персоналу під конкретні потреби компанії;

– додаткові послуги з адаптації та навчання персоналу.

Основними недоліками таких сайтів є:

– висока вартість послуг для компаній-замовників;

– ризик підбору неякісних кандидатів через людський фактор;

– складність масштабування для задоволення потреб великої кількості компаній.

Також варто згадати сайти фріланс-бірж як *HH.ua, Kabanchik.ua, FL.ru*. Вони дозволяють шукати фрілансерів та замовників на виконання разових завдань віддалено.

Переваги таких сайтів – це гнучкість, оперативність, можливість підбору фахівців з усього світу. А недоліки – це ризики шахрайства, складнощі у вирішенні спорів та претензій, обмеженість застосування лише для невеликих разових проектів.

Отже, наявні в даний час вебрішення для автоматизації HR-процесів мають як свої переваги, так і недоліки. Жодне з них не є універсальним і не охоплює всі аспекти взаємодії між роботодавцями та найманими працівниками. Тому створення нового комплексного застосунку, який об’єднував би кращі сторони існуючих платформ, дозволило б якісно трансформувати процеси підбору персоналу та працевлаштування.

Методи та алгоритми аналізу даних можуть стати ефективними інструментами для покращення процесу підбору персоналу, дозволяючи аналізувати великі обсяги інформації для точнішого відбору кандидатів та рекомендації вакансій [24]. Системи, що базуються на цих методах, здатні виявляти не тільки явні, але й приховані відповідності між вакансіями та резюме, підвищуючи якість та швидкість підбору.

Аналітика даних відіграє важливу роль у зборі та обробці інформації про ринок праці, допомагаючи користувачам отримувати актуальні звіти про тенденції зайнятості, середні заробітні плати в галузях та регіонах, що сприяє кращому плануванню кар’єрного росту.

Персоналізація стає все більш важливою, оскільки користувачі прагнуть отримувати інформацію, яка максимально відповідає їхнім індивідуальним потребам та вподобанням. Розвиток технологій дозволяє створювати більш гнучкі та адаптивні системи підбору, які враховують багатоаспектні критерії вибору

користувачів.

Безпека даних залишається одним із головних викликів. Збільшення обсягу персональної інформації, що обробляється вебзастосунками підбору вакансій, вимагає ретельного дотримання принципів захисту даних та конфіденційності.

Інтеграція з соціальними мережами та іншими онлайн-платформами відкриває нові можливості для розширення функціоналу та залучення користувачів.

Сучасні технології аналізу даних, включаючи ймовірнісно-статистичні методи та інтелектуальні алгоритми, дозволяють значно підвищити ефективність пошуку роботи та співробітників [25]. Використання цих методів може допомогти в аналізі великих обсягів інформації про кандидатів, вакансії та ринкові тенденції. Це дозволить створити більш точні та індивідуалізовані рекомендації, а також спрогнозувати потреби ринку праці.

Соціальні мережі є потужним інструментом для пошуку роботи та співробітників, проте багато існуючих платформ не використовують їхній потенціал в повній мірі. Інтеграція з соціальними мережами може дозволити кандидатам більш ефективно демонструвати свої професійні досягнення та встановлювати контакти з потенційними роботодавцями. Водночас, роботодавці зможуть краще оцінювати соціальну активність та професійну репутацію кандидатів.

Багато існуючих платформ страждають від застарілих інтерфейсів та труднощів у навігації. Розробка сучасного, інтуїтивно зрозумілого та зручного інтерфейсу може значно покращити користувацький досвід, залучити більше користувачів та підвищити ефективність взаємодії з платформою.

Питання безпеки та конфіденційності даних залишається актуальним для всіх користувачів вебзастосунків. Важливо розробити надійні механізми захисту особистої інформації та забезпечити високий рівень захисту від несанкціонованого доступу та витоку даних.

Ринок праці в Україні характеризується високим рівнем конкуренції серед

шукачів роботи. За даними Державної служби статистики, рівень безробіття в країні становить близько 10%. В той же час, багато роботодавців скаржаться на нестачу кваліфікованих кадрів. Це свідчить про дисбаланс між попитом та пропозицією робочої сили.

Однією з причин цього дисбалансу є недосконаліть існуючих методів пошуку роботи та підбору персоналу. Більшість вакансій розміщуються на спеціалізованих сайтах, однак їх функціонал часто обмежений. Шукачам бракує інструментів для ефективного пошуку та підбору підходящої роботи.

Тому існує потреба у створенні спеціалізованого вебзастосунку, який би дозволяв:

– роботодавцям публікувати вакансії з детальним описом вимог та умов;

– шукачам створювати профілі з інформацією про досвід, навички, освіту;

– застосовувати інформаційні інтелектуальні технології для покращення процесу підбору вакансій рекрутерами;

– надавати рекомендації шукачам щодо вдосконалення навичок та освіти для підвищення шансів на працевлаштування;

– аналізувати статистичні дані про ринок праці для виявлення найбільш затребуваних професій.

Такий вебзастосунок сприятиме підвищенню ефективності та прозорості ринку праці, скороченню термінів підбору персоналу для роботодавців, а також допоможе шукачам швидше знайти роботу, що відповідає їхнім навичкам та вимогам.

1.2 Огляд існуючих рішень для підбору вакансій та їх недоліки

Для підбору вакансій із застосуванням вебтехнологій існує багато різних рішень, які можна розділити на три основні типи.

1. Вебсайти порталів вакансій, які надають можливість роботодавцям розміщувати свої оголошення про роботу і кандидатам шукати та подавати свої резюме. Приклади таких сайтів: Work.ua, Talent.ua, Rabota.ua тощо.

2. Вебсайти агентств по працевлаштуванню, які надають послуги пошуку та підбору персоналу для роботодавців і кандидатів. Приклади таких сайтів: HeadHunter, Work Service, Staff Service тощо.

3. Вебсайти соціальних мереж, які дозволяють користувачам створювати свої профілі, ділитися своїми досягненнями та інтересами, спілкуватися з іншими користувачами та шукати роботу або співробітників. Приклади таких сайтів: LinkedIn, Facebook, Twitter тощо.

Кожен з цих типів рішень має свої переваги та недоліки, які розглянуті далі.

Вебсайти порталів вакансій мають такі переваги:

– велика кількість вакансій та резюме, які охоплюють різні галузі й регіони;

– можливість швидкого та зручного пошуку вакансій та резюме за різними критеріями, такими як ключові слова, місто, зарплата, досвід, освіта тощо;

– можливість створювати свої акаунти, зберігати свої резюме та вакансії, отримувати сповіщення про нові вакансії та резюме, відгуки та пропозиції;

– можливість використовувати додаткові функції, такі як тести, портфоліо, рейтинги, відгуки, блоги тощо.

Вебсайти порталів вакансій мають такі недоліки:

– низька якість та актуальність деяких вакансій та резюме, які можуть бути застарілими, неповними, неправдивими або дублюватися;

– висока конкуренція та низька відгуковість роботодавців та кандидатів, які можуть не відповідати на повідомлення, не приходити на співбесіди, не повідомляти про свої рішення тощо;

– обмежена можливість персоналізації та інтерактивності, які не дозволяють користувачам виявляти свою індивідуальність, виражати свої емоції, спілкуватися в реальному часі тощо.

Вебсайти агентств по працевлаштуванню мають такі переваги:

– висока якість та актуальність вакансій та резюме, які перевіряються та оновлюються професійними рекрутерами;

– висока відгуковість та ефективність, які забезпечуються швидким та

якісним підбором персоналу та роботи, а також супроводом та консультаціями на всіх етапах співпраці;

– можливість отримати доступ до ексклюзивних вакансій та резюме, які не публікуються на інших ресурсах, а також до складної та дефіцитної робочої сили.

Вебсайти агентств по працевлаштуванню мають такі недоліки:

– обмежена кількість вакансій та резюме, які можуть не відповідати потребам та бажанням деяких роботодавців та кандидатів;

– висока вартість послуг, яку повинні сплачувати роботодавці та/або кандидати за підбір персоналу та роботи, а також за додаткові послуги, такі як навчання, сертифікація, перевірка тощо;

– ризик обману та недотримання умов договору, який може виникнути внаслідок непрофесійності або недобросовісності деяких агентств або їх співробітників.

Вебсайти соціальних мереж мають такі переваги:

– велика кількість користувачів та контенту, які створюють велику аудиторію та різноманітність інформації про роботу та персонал;

– можливість побудови свого бренду та репутації, які дозволяють користувачам демонструвати свої досягнення, вміння, інтереси, рекомендації;

– можливість встановлювати та підтримувати контакти з потенційними роботодавцями та кандидатами, а також отримувати відгуки та поради від інших користувачів.

Вебсайти соціальних мереж мають такі недоліки:

– низька спеціалізація та релевантність деякої інформації про роботу та персонал, яка може бути не пов’язана з професійними інтересами та цілями користувачів;

– висока конкуренція та вплив факторів, які не стосуються якості роботи та персоналу, таких як популярність, симпатії, стереотипи тощо;

– ризик порушення конфіденційності та безпеки даних, який може виникнути внаслідок неконтрольованого поширення та використання інформації

про роботу та персонал на вебсайтах соціальних мереж.

У табл. 1.2 наведено порівняння вебзастосунків підбору вакансій.

Таблиця 1.2 – Порівняння вебзастосунків підбору вакансій

Тип вебзастосунку	**Переваги**	**Недоліки**
Вебсайти порталів вакансій	– велика кількість вакансій та резюме – зручний пошук за різними критеріями – можливість створювати акаунти та отримувати сповіщення – додаткові функції, такі як тести, портфоліо, рейтинги тощо	– низька якість та актуальність деяких вакансій та резюме – висока конкуренція та низька відгуковість – обмежена можливість персоналізації та інтерактивності
Вебсайти агентств по працевлаштуванню	– висока якість та актуальність вакансій та резюме – висока відгуковість та ефективність – можливість отримати доступ до ексклюзивних вакансій та резюме	– обмежена кількість вакансій та резюме – висока вартість послуг – ризик обману та недотримання умов договору
Вебсайти соціальних мереж	– велика кількість користувачів та контенту – можливість побудови свого бренду та репутації – можливість встановлювати та підтримувати контакти	– низька спеціалізація та релевантність деякої інформації – висока конкуренція та вплив факторів, які не стосуються якості роботи та персоналу – ризик порушення конфіденційності та безпеки даних

Одним з лідерів ринку є сайт Work.ua. Даний ресурс дозволяє роботодавцям розміщувати вакансії з описом вимог, а користувачам – створювати резюме з

інформацією про освіту, досвід та навички. Користувачі можуть самостійно підбирати вакансії за ключовими словами або отримувати персональні рекомендації. Основними недоліками Work.ua є застарілий дизайн, велика кількість нерелевантних оголошень та відсутність інтелектуального аналізу й обробки великих об’ємів даних.

Система Управління Наймом (СУН) Work.ua є найбільшим вебзастосунком підбору вакансій в Україні, який містить понад 200 тисяч вакансій та 3 мільйони резюме, а також:

– надає можливість роботодавцям створювати власні кар’єрні сторінки, використовувати ботів для спілкування з кандидатами, вести статистику та аналітику по найму;

– надає можливість працівникам шукати вакансії за різними критеріями, створювати власні резюме, отримувати рекомендації та пропозиції вакансій;

– має високу конкуренцію серед роботодавців та працівників, що ускладнює процес підбору;

– має обмежену кількість безкоштовних оголошень для роботодавців, що змушує їх платити за додаткові послуги;

– має низьку якість деяких вакансій та резюме, що утруднює перевірку та відбір кандидатів [5].

Сервіс Rabota.ua схожий за функціоналом до Work.ua, проте відрізняється більш зручним та сучасним інтерфейсом та наявністю мобільного додатку. Проте цей ресурс також не здійснює глибинний аналіз резюме кандидатів та не надає роботодавцям інструментів для оцінки відповідності кандидатів вимогам вакансій. Має місце поширення неякісних та фейкових вакансій.

Одним із найбільш технологічних проектів є Grc.ua, який використовує нейронні мережі для обробки даних та підбору кандидатів. Система дозволяє роботодавцям детально описати вимоги, ідеального кандидата та умови роботи, після чого порівнює ці вимоги з реальними резюме та видає найбільш релевантні результати. До недоліків можна віднести вузьку спеціалізацію на IT-сфері та слабку

інтеграцію з соціальними мережами. Кількість вакансій з інших галузей є обмеженою [6].

Ще одним популярним сервісом є Jooble.ua. Його перевагами є агрегація вакансій з багатьох інших джерел та ресурсів, зручний пошук за фільтрами, наявність мобільної версії. Проте алгоритми підбору та ранжування вакансій є недостатньо інтелектуальними, а особистий кабінет містить мало корисних функцій для аналізу ринку праці.

За останні роки з’являється все більше нішових сервісів, орієнтованих на певні професії чи галузі – IT, бухгалтерія, маркетинг, продажі. Прикладами є Dou.ua, Lemon.ua. Вони створюють більш персоналізовані рекомендації під конкретні запити користувачів, проте мають вузьку спеціалізацію і не охоплюють усіх можливих вакансій на ринку.

Отже, існуючі рішення мають ряд суттєвих недоліків та прогалин у функціоналі: застарілий дизайн та обмежені інструменти аналізу (Work.ua), відсутність галузевої спеціалізації та інтеграцій (Grc.ua), поширення неякісного контенту (Rabota.ua). Жодне з розглянутих рішень не забезпечує комплексного задоволення потреб користувачів на ринку праці. Тому розробка нового спеціалізованого вебзастосунку з урахуванням недоліків вже існуючих платформ є цілком виправданою і дозволить суттєво розширити функціонал та спектр можливостей для учасників ринку праці.

1.3 Обґрунтування вибору технологій розробки вебзастосунку

Для розробки вебзастосунку підбору вакансій було обрано наступний стек технологій: PHP, HTML, CSS, MySQL, Bootstrap, JavaScript.

1. PHP – мова програмування, яка використовується для створення серверної частини вебзастосунку. PHP є широко поширеною, гнучкою, швидкою та легкою у вивченні мовою, яка підтримує багато баз даних, протоколів та форматів обміну даними. PHP також має багато фреймворків, які спрощують розробку та підвищують безпеку та якість коду. Деякі з них: Laravel, Symfony,

CodeIgniter тощо.

2. HTML – мова розмітки, яка використовується для створення структури вебсторінок. HTML дозволяє використовувати різні елементи, такі як заголовки, абзаци, списки, таблиці, форми, посилання, зображення тощо, для представлення інформації на вебсторінці.

3. CSS – мова стилів, яка використовується для задання зовнішнього вигляду вебсторінок. CSS дозволяє використовувати різні властивості, такі як кольори, шрифти, відступи, вирівнювання, анімації тощо, для зміни візуального оформлення HTML-елементів.

4. MySQL – система керування базами даних, яка використовується для зберігання та обробки даних вебзастосунку. MySQL є надійною, швидкою, масштабованою та легкою у використанні системою, яка підтримує реляційну модель даних, SQL-запити, транзакції, індекси, тригери, процедури тощо.

5. Bootstrap – фреймворк, який використовується для створення адаптивного та сучасного інтерфейсу вебзастосунку. Bootstrap надає готові компоненти, такі як кнопки, меню, модальні вікна, каруселі, сітки тощо, які можна легко використовувати та налаштовувати за допомогою HTML, CSS та JavaScript.

6. JavaScript – мова програмування, яка використовується для створення клієнтської частини вебзастосунку. JavaScript дозволяє додавати інтерактивність, динамічність, валідацію, аякс-запити, анімації тощо до вебсторінок. JavaScript також має багато бібліотек та фреймворків, які розширюють можливості мови та спрощують розробку. Деякі з них: jQuery, React, Angular, Vue тощо.

Ці технології розробки обрано на основі наступних критеріїв:

– популярність та актуальність: ці технології є досить поширеними та сучасними на ринку веброзробки, що свідчить про їхню ефективність, надійність та підтримку;

– сумісність та інтеграція: ці технології добре працюють разом та легко інтегруються з іншими технологіями, що дозволяє створювати гнучкі та модульні вебзастосунки;

– простота та доступність: ці технології є досить простими у вивченні та використанні, а також мають багато безкоштовних та відкритих ресурсів, таких як документація, книги, курси, форуми, блоги тощо, які допомагають розробникам.

Зазначений стек технологій є оптимальним по співвідношенню продуктивності, безпеки, вартості розробки та подальшого супроводу. Всі обрані мови та фреймворки є загальноприйнятими в галузі, мають великі спільноти розробників та значну кількість готових бібліотек і плагінів, що можуть бути використані при реалізації вебзастосунку для автоматизації підбору вакансій.

1.4 Технічне завдання на розробку вебзастосунку

Зручність користування системою є одним із ключових факторів, який впливає на задоволеність користувачів та їхнє бажання використовувати вебзастосунок для підбору вакансій. Розробляючи вебзастосунок, ми прагнемо досягти максимальної інтуїтивності інтерфейсу, забезпечити легкість навігації та швидкість завантаження сторінок. Адаптивний дизайн забезпечує комфортне використання вебзастосунку на різних пристроях, включаючи смартфони, планшети та настільні комп’ютери, забезпечуючи користувачам можливість швидкого доступу до необхідної інформації в будь-який час і з будь-якого місця.

Для оптимізації взаємодії з користувачем система надає зручні інструменти для пошуку та фільтрації вакансій, які дозволяють користувачам ефективно відсіювати пропозиції, що не відповідають їхнім критеріям. Розширені можливості персоналізації дозволяють користувачам налаштовувати інтерфейс за своїми уподобаннями, зберігаючи при цьому індивідуальні налаштування пошуку, вибрані вакансії та резюме [10].

Безпека вебзастосунку «Підбір вакансій» є пріоритетним завданням, оскільки система обробляє велику кількість персональних даних користувачів, включаючи резюме, контактну інформацію та історію пошуку роботи. Застосунок розроблено з урахуванням найкращих практик безпеки та використовує сучасні методи шифрування даних для захисту інформації від несанкціонованого доступу [22, 23].

Авторизація та аутентифікація користувачів відбувається за допомогою безпечних механізмів, які запобігають ризику витоку даних. Система надає різні рівні доступу, забезпечуючи, що користувачі можуть переглядати та редагувати лише ту інформацію, до якої вони мають право доступу. Регулярне оновлення програмного забезпечення та використання сучасних антивірусних рішень гарантує захист від зовнішніх загроз та вразливостей.

Для забезпечення конфіденційності користувачів вебзастосунок дотримується нормативних актів щодо захисту даних, включаючи GDPR для користувачів з Європейського Союзу. Це забезпечує користувачам контроль над їхніми даними та можливість їх видалення за запитом.

Загалом, вебзастосунок «Підбір вакансій» розроблено з метою надати користувачам зручний, інтуїтивно зрозумілий і безпечний інструмент для ефективного пошуку роботи та підбору персоналу. Завдяки продуманому дизайну, сучасним технологіям та високим стандартам безпеки, система забезпечує високу продуктивність та задоволення потреб усіх користувачів.

Назва системи: вебзастосунок «Підбір вакансій».

Замовник: ФОП Іванов І. І.

Виконавець: студентка Чорноморського національного університету ім. Петра Могили.

Загальні вимоги:

– вебзастосунок повинен бути розроблений з використанням сучасних вебтехнологій, таких як HTML, CSS, JavaScript, PHP та MySQL;

– застосунок має бути адаптивним та коректно відображатися на різних пристроях та розмірах екрану;

– система повинна забезпечувати високу швидкодію, безпеку та масштабованість.

Функціональні вимоги:

– авторизація та автентифікація користувачів:

а) система повинна підтримувати реєстрацію нових користувачів та вхід

існуючих користувачів;

б) повинні бути реалізовані різні рівні доступу для адміністраторів, рекрутерів та кандидатів;

– управління вакансіями:

а) адміністратори та рекрутери повинні мати можливість додавати, редагувати та видаляти вакансії;

б) кожна вакансія повинна містити детальну інформацію, таку як назва, місто, кількість місць, вимоги до кандидатів тощо;

в) система повинна підтримувати завантаження та відображення медіафайлів, пов’язаних з вакансіями;

– пошук та фільтрація вакансій:

а) користувачі повинні мати можливість здійснювати пошук вакансій за ключовими словами, містом, галуззю тощо;

б) система повинна надавати зручні інструменти фільтрації та сортування результатів пошуку;

– передача клієнтів на вакансії:

а) рекрутери повинні мати можливість передавати інформацію про клієнтів на конкретні вакансії;

б) система повинна зберігати детальну інформацію про клієнтів, включаючи персональні дані, документи, фото тощо;

– управління передачами клієнтів:

а) адміністратори повинні мати можливість переглядати, підтверджувати та відхиляти передачі клієнтів;

б) система повинна відстежувати статус кожної передачі та інформувати відповідних користувачів про зміни;

– комунікація та взаємодія:

а) вебзастосунок повинен включати систему обміну повідомленнями між користувачами;

б) адміністратори повинні мати можливість призначати завдання

рекрутерам та відстежувати їх виконання.

Вимоги до інтерфейсу:

– головна сторінка повинна відображати список актуальних вакансій з основною інформацією та посиланнями на деталі;

– сторінка деталей вакансії повинна містити повну інформацію про вакансію, включаючи опис, вимоги, медіафайли тощо;

– форма додавання/редагування вакансій повинна бути інтуїтивно зрозумілою та зручною у використанні;

– сторінка управління передачами клієнтів повинна надавати чіткий огляд статусу передач та дозволяти виконувати відповідні дії;

– інтерфейс повинен бути візуально привабливим, з використанням сучасних принципів дизайну та юзабіліті.

Вимоги до безпеки:

– вебзастосунок повинен забезпечувати захист конфіденційних даних користувачів та клієнтів;

– повинні бути реалізовані механізми захисту від несанкціонованого доступу, SQL-ін’єкцій, XSS-атак тощо;

– система повинна використовувати надійні методи шифрування для зберігання паролів та чутливої інформації.

Вимоги до продуктивності:

– вебзастосунок повинен забезпечувати швидке завантаження сторінок та мінімальний час відгуку сервера;

– система повинна ефективно працювати з великими обсягами даних та витримувати значне навантаження користувачів;

– повинні бути реалізовані механізми кешування та оптимізації запитів до бази даних для підвищення продуктивності.

Додаткові вимоги:

– вебзастосунок повинен бути сумісним з сучасними браузерами, такими як Chrome, Firefox, Safari та Edge;

– система повинна мати можливість інтеграції з іншими сервісами та платформами, такими як соціальні мережі, системи розсилки електронної пошти;

– повинна бути передбачена можливість розширення функціональності вебзастосунку в майбутньому, з урахуванням потенційних потреб користувачів та розвитку ринку праці.

Етапи та терміни виконання:

– проектування бази даних та розробка архітектури вебзастосунку (2 тижні);

– розробка серверної частини та інтеграція з базою даних (4 тижні);

– розробка клієнтської частини та реалізація інтерфейсу користувача (6 тижнів);

– тестування, налагодження та оптимізація вебзастосунку (2 тижні);

– розгортання вебзастосунку на продакшн-сервері та завершальне тестування (1 тиждень).

Висновки до розділу 1

У першому розділі було проаналізовано актуальний стан ринку працевлаштування та існуючі інструменти пошуку роботи й підбору персоналу. Виявлено, що традиційні методи вже не забезпечують необхідної ефективності, а існуючі вебсервіси мають низку недоліків.

З огляду на це можна зробити висновок про доцільність та перспективність розробки спеціалізованого вебзастосунку з розширеним функціоналом, що відповідатиме потребам усіх учасників ринку праці. Ретельний аналіз вимог дозволив сформувати технічне завдання на створення такого програмного продукту.

Було обрано оптимальний стек технологій для реалізації системи, а саме: мова PHP, СУБД MySQL, HTML, CSS, JavaScript, фреймворк Bootstrap. Даний набір інструментів задовольняє вимоги за функціональністю, безпекою, термінами виконання та вартістю проекту.

Отже, проведене дослідження переконливо демонструє перспективність розробки вебзастосунку автоматизації підбору вакансій та формує міцне підґрунтя для його подальшого проектування та програмної реалізації.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ

2.1 Архітектура вебзастосунку

Архітектура вебзастосунку є сукупністю компонентів, що визначають його логічну та фізичну структуру, а також способи їх взаємодії [13]. Вона впливає на функціональність, продуктивність, надійність, масштабованість, безпеку, зручність та інші характеристики вебзастосунку.

Для розробки вебзастосунку підбору вакансій було вибрано архітектуру, яку можна поділити на: клієнт-серверну, тривимірну, мікросервісну архітектуру [15].

1. *Клієнт-серверна архітектура*. Це означає, що вебзастосунок складається з двох частин: клієнтської та серверної. Клієнтська частина відповідає за відображення інтерфейсу вебзастосунку в браузері користувача та надсилання запитів до сервера. Серверна частина відповідає за обробку запитів від клієнта, доступ до бази даних та надання відповідей клієнту. Клієнт та сервер спілкуються за допомогою протоколу HTTP3 [10].

2. *Тривимірна архітектура*. Це означає, що серверна частина вебзастосунку поділена на три рівні: рівень представлення, рівень бізнес-логіки та рівень даних. Рівень представлення відповідає за формування вебсторінок, які надсилаються клієнту. Рівень бізнес-логіки відповідає за виконання основних функцій вебзастосунку, таких як авторизація, постановка задач, ділове листування, додавання, видалення та редагування вакансій та кандидатів тощо. Рівень даних відповідає за зберігання та обробку даних в базі даних [17].

3. *Мікросервісна архітектура*. Це означає, що рівень бізнес-логіки вебзастосунку поділений на декілька незалежних сервісів, які виконують певні функції та спілкуються між собою за допомогою API. Мікросервісна архітектура дозволяє збільшити гнучкість, масштабованість, надійність та швидкість розробки вебзастосунку, а також спростити тестування та впровадження.

Для реалізації цієї архітектури було використано наступні технології:

– *для клієнтської частини:* HTML, CSS, JavaScript, Bootstrap;

– *для серверної частини:* PHP, MySQL, RESTful API;
– *для розгортання та підтримки вебзастосунку:* Docker, AWS.

Ця архітектура вебзастосунку відповідає вимогам та потребам мого проекту, а також дотримується сучасних стандартів та практик веброзробки.

Архітектура розроблюваного вебзастосунку відповідатиме багатошаровій архітектурі за зразком MVC (Model-View-Controller). Така архітектура є одним з найпоширеніших шаблонів побудови додатків і дозволить створити гнучку структуру системи з чітким розподілом функцій між компонентами.

Модель (Model) представлятиме дані та бізнес-логіку, зокрема, класи для роботи з базою даних, моделі вакансій, резюме, користувачів тощо. У моделі також розміщуватимуться функції для виконання розрахунків, валідацій, порівнянь.

Представлення (View) міститиме користувацький інтерфейс – html-шаблони та стилі оформлення, які генерують вихідний html для перегляду в браузері.

Контролер (Controller) виступатиме посередником між моделлю та представленням. Контролер отримуватиме запити користувачів, взаємодіятиме з моделями, формуватиме дані у необхідному вигляді та передаватиме їх шаблонам представлення.

Такий поділ дозволить забезпечити гнучку розширюваність функціоналу вебзастосунку, простоту супроводу окремих компонентів та тестування системи.

Серверна частина системи базуватиметься на мові програмування PHP та вебсервері Apache. Перевагою цього стеку є надійність, широкі можливості масштабування, а також простота розгортання та підтримки. HTML дозволить пришвидшити розробку серверної частини завдяки вбудованим шаблонам та функціям для роботи з базою даних, роутингу, кешування, аутентифікації тощо.

Для збереження даних на сервері використовуватиметься СУБД MySQL, найбільш популярна реляційна база даних, що має широку підтримку в PHP. В MySQL буде створено базу даних та схема з такими основними таблицями: користувачі, вакансії, резюме, завдання, повідомлення. Також будуть створені додаткові таблиці для реалізації зв’язків між даними, підтримки історії змін та

забезпечення статистичної інформації.

Клієнтська частина реалізовуватиметься із застосуванням HTML, CSS, JavaScript, це дозволяє створювати односторінковий інтерфейс з високою швидкістю рендерінгу сторінок. Для адаптивного відображення на пристроях різних розмірів застосовуватиметься CSS-фреймворк Bootstrap.

Така архітектура системи з розділенням на незалежні компоненти та використанням сучасних вебтехнологій дозволить створити гнучкий, безпечний та масштабований вебдодаток зі зручним і швидким користувацьким інтерфейсом. Подібна архітектура широко застосовується для вебзастосунків подібного типу та масштабу.

2.2 Розробка структури бази даних

Для збереження даних вебзастосунку автоматизації підбору вакансій було розроблено структуру бази даних на основі реляційної СУБД MySQL. Виходячи з поставлених вимог до функціоналу системи, було спроектовано 13 основних таблиць.

Таблиця «users» зберігатиме дані про користувачів системи: їх ідентифікатор, ім’я, пароль, рівень доступу, контактні дані тощо. Користувачі поділяються на адміністратора, який має повний доступ, та рекрутерів з обмеженими можливостями.

У таблиці «vacancies» зберігатиметься інформація про вакансії: назва посади, вимоги до кандидатів, умови праці, детальний опис. Додаткова мультимедійна інформація щодо вакансій (фото, схеми, презентації) зберігатиметься в таблиці «vacancy_media» з прив’язкою до конкретної вакансії.

Дані про подані резюме кандидатів на вакансії зберігатимуться у таблиці «client_transfers». Вона міститиме інформацію про дату передачі резюме, ПІБ кандидата, паспортні дані, вакансію на яку подається та інші атрибути. Фото документів кандидатів можуть зберігатися в окремій таблиці «client_passport_photos» з прив’язкою до конкретного резюме.

Інформація про підтвердження найму кандидата та винагороду рекрутера за успішне працевлаштування міститиметься в таблицях «client_transfer_confirmations» та «client_transfer_payed» відповідно.

Для організації внутрішніх комунікацій передбачені таблиці «user_tasks» для створення та відстеження задач рекрутерам, а також «chat_messages» для обміну повідомленням у чаті.

Інші таблиці, такі як «access_levels», «user_financial_transactions», «workers» та інші допоміжні, міститимуть довідкову інформацію для коректної роботи бізнес-логіки системи.

Зв’язки між основними сутностями реалізовуватимуться за допомогою зовнішніх ключів (foreign keys), що забезпечить цілісність даних. Така структура БД є оптимальною для поставлених вимог і дозволить якісно організувати збереження та маніпулювання даними у вебзастосунку.

Кожна з основних сутностей (таблиць) має унікальний ідентифікатор запису id, що є первинним ключем таблиці. Це дозволяє однозначно ідентифікувати кожен запис та коректно встановлювати зв’язки між сутностями.

Для таблиць, що містять довідкову інформацію, таких як «access_levels», передбачений атрибут «name» – назва рівня доступу користувача. А для основних сутностей, таких як «users», «vacancies», реалізовані атрибути «username», «name» відповідно, що містять заголовок або ім’я об’єкта.

Для збереження великих об’ємів текстової інформації використовується тип даних «text» або «mediumtext». Наприклад, поля «detailed_description» у вакансіях, чи «documents» у резюме кандидатів.

Для атрибутів, де потрібно фіксувати дату і час, використовуються спеціальні типи «datetime» та «timestamp». Наприклад, поля «last_login», «payment_date», «comment_date» фіксують час останнього входу в систему користувача, дату оплати, дату коментаря до завдання відповідно.

Для полів з обмеженою кількістю можливих значень (статусів, пріоритетів, типів) застосовуються переліки «enum». Наприклад, статус задачі може бути New,

In Progress чи Completed.

На рис. 2.1 зображена ER діаграма бази даних.

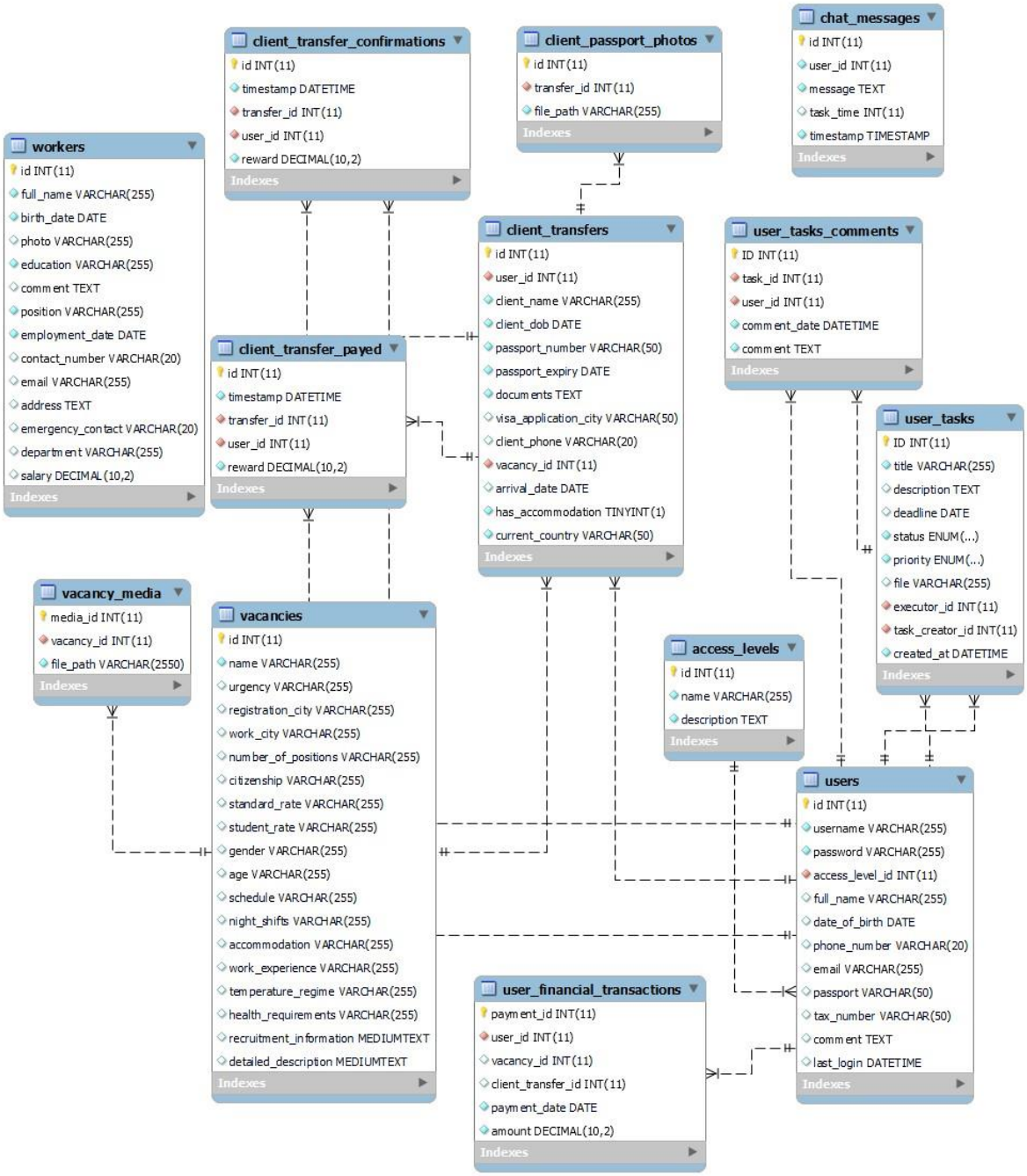

Рисунок 2.1 – ER діаграма бази даних

Основні таблиці бази даних:

– **access_levels**: таблиця визначає різні рівні доступу користувачів до вебзастосунку. Це дозволяє реалізувати систему дозволів, яка обмежує доступ до

певних функцій залежно від ролі користувача;

– **users**: основна таблиця для зберігання інформації про користувачів, включаючи їх особисті дані, контактну інформацію, а також посилання на рівень доступу. Ця таблиця є центральною для аутентифікації та авторизації користувачів;

– **vacancies**: таблиця містить деталі вакансій, які роботодавці розміщують у вебзастосунку. Вона включає інформацію про вимоги до кандидатів, умови праці, зарплату тощо;

– **client_transfers**, **client_passport_photos**, **client_transfer_confirmations**, **client_transfer_payed**: ці таблиці управляють інформацією, пов’язаною з переміщенням працівників, включаючи паспортні фотографії, підтвердження трансферів та оплату. Вони важливі для відстеження процесу найму і міграції працівників;

– **chat_messages**: таблиця для зберігання повідомлень між користувачами у вбудованій системі чату. Це сприяє комунікації між пошукачами та роботодавцями;

– **user_financial_transactions**: зберігає дані про фінансові транзакції користувачів, такі як оплата за послуги вебзастосунку. Це включає інформацію про оплату за розміщення вакансій, перекази тощо;

– **user_tasks**, **user_tasks_comments**: таблиці для управління завданнями, які користувачі можуть створювати та коментувати в межах вебзастосунку. Це може бути корисно для відстеження процесу виконання робіт або проектів;

– **vacancy_media**: зберігає медіафайли, пов’язані з вакансіями, наприклад, фотографії або документи, які надають додаткову інформацію про робочі місця. Це може підвищити привабливість вакансій для потенційних кандидатів та допомогти їм краще зрозуміти умови праці;

– **workers:** таблиця містить інформацію про працівників, які були найняті через вебзастосунок. Вона включає особисті дані, освіту, досвід роботи, контактну інформацію тощо. Це дозволяє роботодавцям легко управляти інформацією про своїх працівників і відстежувати їх кар’єрний ріст.

Основні елементи структури бази даних, такі як таблиці та їх взаємозв’язки, відіграють важливу роль у забезпеченні ефективності та гнучкості вебзастосунку. Кожна таблиця має первинний ключ (**PRIMARY KEY**), який гарантує унікальність кожного запису. Для таблиць, де існує залежність між записами, використовуються зовнішні ключі (**FOREIGN KEY**), що дозволяє забезпечити цілісність даних між різними частинами бази даних.

Для підвищення продуктивності бази даних та швидкості обробки запитів важливо правильно використовувати індексацію. Індекси створюються для ключових стовпців, які часто використовуються в умовах запитів (**WHERE**), у з’єднаннях (**JOIN**) або при сортуванні результатів (**ORDER BY**). Це дозволяє базі даних ефективніше виконувати пошук та зменшує час відгуку для користувача.

Безпека даних є критично важливим аспектом при розробці вебзастосунків, особливо коли йдеться про зберігання особистої інформації користувачів та фінансових даних. Важливо застосовувати сучасні методи шифрування для захисту паролів та конфіденційної інформації. Крім того, слід регулярно проводити аудит безпеки, щоб ідентифікувати та усунути потенційні вразливості.

При проектуванні структури бази даних слід враховувати потреби в масштабуванні вебзастосунку. Це означає, що архітектура бази даних має бути готова до зростання обсягу даних та збільшення кількості користувачів. Використання обчислювальних хмар та розподілених систем може допомогти забезпечити високу доступність та швидкість обробки даних незалежно від навантаження.

Отже, спроектована база даних дозволяє гнучко зберігати усі необхідні дані для роботи системи автоматизації підбору вакансій з урахуванням всіх вимог та особливостей предметної області. Використання сучасних принципів проектування БД разом з можливостями СУБД MySQL дають змогу реалізувати якісне сховище, що забезпечить швидкий та надійний доступ до даних.

2.3 Проектування інтерфейсу вебзастосунку

Проектування інтерфейсу користувача є важливим етапом у розробці вебзастосунку, оскільки він визначає, як користувачі взаємодіятимуть із системою. Інтерфейс повинен бути інтуїтивно зрозумілим, зручним у використанні та візуально привабливим. Основна мета полягає в тому, щоб забезпечити користувачам якісний досвід взаємодії з вебзастосунком та дозволити їм ефективно виконувати свої завдання.

При проектуванні інтерфейсу вебзастосунку для підбору вакансій були враховані наступні ключові аспекти:

1) *розташування та організація елементів*: розміщення навігаційних елементів, форм, кнопок та інших компонентів інтерфейсу має бути логічним та послідовним, щоб користувачі могли легко орієнтуватися та знаходити потрібну інформацію;

2) *візуальна ієрархія*: використання кольорів, розмірів, шрифтів та інших візуальних елементів для виділення важливої інформації та створення чіткої візуальної ієрархії, що допомагає користувачам швидко сприймати та розуміти зміст;

3) *адаптивний дизайн*: забезпечення коректного відображення та функціонування інтерфейсу на різних розмірах екрану, щоб користувачі могли зручно працювати з вебзастосунком незалежно від пристрою, який вони використовують;

4) *зворотній зв'язок*: надання користувачам чітких повідомлень та візуальних підказок про результати їхніх дій, помилки або успішне виконання операцій, щоб вони завжди були поінформовані про стан системи;

5) *доступність*: врахування потреб користувачів з обмеженими можливостями та забезпечення доступності інтерфейсу для всіх користувачів, незалежно від їхніх фізичних обмежень.

Функція **authenticateUser** виконує аутентифікацію користувача. Вона

отримує ім’я користувача та пароль, виконує запит до бази даних, щоб знайти користувача з відповідним ім’ям, і перевіряє, чи збігається наданий пароль зі збереженим хешем пароля. Якщо аутентифікація успішна, функція зберігає ідентифікатор користувача, ім’я користувача та рівень доступу в сесії та повертає true. В іншому випадку вона повертає false.

Код сторінки **login.php**:

1) запускається сесія за допомогою session_start();

2) підключається файл functions.php, який містить необхідні функції;

3) перевіряється, чи користувач вже увійшов в систему за допомогою функції isLoggedIn(). Якщо так, то відбувається перенаправлення на сторінку index.php;

4) ініціалізується змінна $error для зберігання повідомлення про помилку;

5) якщо запит є запитом POST, то отримуються введені користувачем ім’я та пароль;

6) викликається функція authenticateUser для аутентифікації користувача. Якщо аутентифікація успішна, відбувається перенаправлення на сторінку index.php, інакше встановлюється повідомлення про помилку в змінній $error;

7) підключається заголовок сторінки за допомогою include ‘templates/header.php’;

8) виводиться форма входу з полями для введення імені користувача та пароля. Якщо є повідомлення про помилку, воно відображається над формою;

9) підключається підвал сторінки за допомогою include ‘templates/footer.php’.

Загалом, ця сторінка забезпечує функціональність входу користувача в систему. Користувач вводить своє ім’я та пароль, і якщо вони правильні, користувач перенаправляється на головну сторінку (index.php). В іншому випадку відображається повідомлення про помилку.

Інтерфейс сторінки містить форму з полями для введення імені користувача та пароля, кнопку «Увійти» для відправки форми та кнопку «Зареєструватися» для

переходу на сторінку реєстрації. Використовуються класи Bootstrap для стилізації елементів форми та повідомлення про помилку.

Сторінка **index.php** є головною сторінкою вебзастосунку після входу користувача в систему. На цій сторінці відображається список актуальних вакансій у вигляді таблиці. Користувач може переглянути назву вакансії, місто роботи, кількість доступних місць та стать, на яку розрахована вакансія. При натисканні на назву вакансії користувач перенаправляється на сторінку з детальною інформацією про обрану вакансію.

Код сторінки index.php забезпечує такі функціональності:

– перевірка автентифікації користувача та перенаправлення на сторінку входу, якщо користувач не авторизований;

– оновлення часу останнього входу користувача в систему;

– отримання списку актуальних вакансій з бази даних за допомогою функції getVacancies();

– відображення списку вакансій у вигляді HTML-таблиці з можливістю переходу на сторінку детальної інформації про кожну вакансію.

Функції getVacancies() та updateLastLogin($userId) забезпечують отримання даних про вакансії з бази даних та оновлення часу останнього входу користувача відповідно.

Тепер розглянемо детальніше функції та код сторінки index.php:

1) запускається сесія за допомогою session_start();

2) підключається файл functions.php, який містить необхідні функції;

3) перевіряється, чи користувач увійшов у систему за допомогою функції isLoggedIn(). Якщо користувач не авторизований, відбувається перенаправлення на сторінку login.php;

4) викликається функція updateLastLogin() з ідентифікатором поточного користувача ($_SESSION[‘user_id’]), щоб оновити час останнього входу;

5) викликається функція getVacancies() для отримання списку всіх вакансій з бази даних. Результат зберігається у змінній $vacancies;

6) підключається заголовок сторінки за допомогою include ‘templates/header.php’;

7) виводиться заголовок «Актуальні Вакансії»;

8) створюється HTML-таблиця для відображення списку вакансій;

9) у циклі foreach перебираються всі вакансії з масиву $vacancies;

10) для кожної вакансії створюється рядок таблиці (<tr>) з такими даними:

– назва вакансії з посиланням на сторінку детальної інформації про вакансію (vacancy_details.php). Ідентифікатор вакансії передається через параметр id в URL;

– місто роботи;

– кількість місць;

– стать;

11) закривається таблиця;

12) підключається підвал сторінки за допомогою include ‘templates/footer.php’.

Ця сторінка є головною сторінкою вебзастосунку після входу користувача в систему. Вона відображає список актуальних вакансій у вигляді таблиці. Користувач може переглянути детальну інформацію про кожну вакансію, натиснувши на її назву, яка є посиланням на сторінку vacancy_details.php з відповідним ідентифікатором вакансії.

Сторінка **vacancy_details.php** відображає детальну інформацію про обрану вакансію. Користувач потрапляє на цю сторінку, натиснувши на назву вакансії на головній сторінці списку вакансій. На сторінці деталей вакансії відображається повна інформація про вакансію, включаючи назву, місто роботи, кількість доступних місць, стать, на яку розрахована вакансія, та докладний опис. Крім того, сторінка містить список медіафайлів, пов’язаних з цією вакансією, які можна переглянути та завантажити.

Функція getVacancyDetails($id) відповідає за отримання деталей вакансії з бази даних. Вона приймає ідентифікатор вакансії як параметр і виконує два запити

до бази даних: перший запит отримує деталі вакансії за заданим ідентифікатором, а другий запит отримує список медіафайлів, пов’язаних з цією вакансією. Функція повертає масив, що містить інформацію про вакансію та список медіафайлів.

Код сторінки vacancy_details.php виконує такі дії:

1) перевіряє, чи користувач увійшов у систему. Якщо ні, перенаправляє на сторінку входу;

2) перевіряє, чи передано ідентифікатор вакансії в параметрі URL. Якщо ні, перенаправляє на сторінку списку вакансій;

3) отримує ідентифікатор вакансії з параметра URL;

4) викликає функцію getVacancyDetails() для отримання деталей вакансії та списку медіафайлів;

5) відображає деталі вакансії в таблиці, включаючи назву, місто роботи, кількість місць, стать та докладний опис;

6) відображає кнопки «Редагувати» та «Видалити» для зміни або видалення вакансії;

7) відображає список медіафайлів, пов’язаних з вакансією, у вигляді горизонтального прокручуваного блоку зображень. Кожне зображення є посиланням для завантаження відповідного медіафайлу;

8) відображає кнопку «Повернутись до списку вакансій» для повернення на сторінку списку вакансій.

Загалом, сторінка vacancy_details.php надає користувачеві детальну інформацію про обрану вакансію та дозволяє переглядати та взаємодіяти з пов’язаними медіафайлами.

Сторінка **vacancy.php** відображає список актуальних вакансій в інтерактивному та зручному для користувача вигляді. Вона надає можливість здійснювати пошук вакансій за назвою, містом реєстрації або містом роботи. Кожна вакансія представлена у вигляді окремого блоку з основною інформацією, такою як назва вакансії, місто роботи, кількість доступних місць та стать, на яку розрахована вакансія.

Функція getVacancies() відповідає за отримання списку всіх вакансій з бази даних. Вона виконує SQL-запит для вибірки всіх записів з таблиці vacancies, підготовлює та виконує запит за допомогою PDO, а потім повертає результати у вигляді асоціативного масиву. У разі виникнення помилки при виконанні запиту, функція логує помилку та повертає порожній масив.

Код сторінки vacancy.php виконує такі дії:

1) перевіряє, чи користувач увійшов у систему. Якщо ні, перенаправляє на сторінку входу;

2) оновлює час останнього входу користувача за допомогою функції updateLastLogin();

3) отримує список всіх вакансій за допомогою функції getVacancies();

4) якщо користувач має рівень доступу 1 (адміністратор), відображає кнопку «Додати вакансію» для переходу на сторінку додавання нової вакансії;

5) відображає поле введення для пошуку вакансій за назвою, містом реєстрації або містом роботи;

6) використовує JavaScript для фільтрації та відображення вакансій на основі введеного користувачем тексту пошуку;

7) для кожної вакансії створює окремий блок з інформацією про вакансію, включаючи назву вакансії з посиланням на сторінку деталей вакансії, місто роботи, кількість доступних місць та стать;

8) додає кнопку «Передати клієнта» для кожної вакансії, яка дозволяє перейти на сторінку передачі клієнта на цю вакансію.

JavaScript код на сторінці виконує наступні функції:

– filterVacancies(searchText): фільтрує список вакансій на основі введеного користувачем тексту пошуку, порівнюючи його з назвою вакансії, містом реєстрації та містом роботи;

– renderVacancies(filteredVacancies): відображає відфільтровані вакансії на сторінці, створюючи окремий блок для кожної вакансії;

– createVacancyElement(vacancy): створює окремий елемент вакансії з усією

необхідною інформацією та кнопкою «Передати клієнта».

Загалом, сторінка vacancy.php надає користувачеві зручний та інтерактивний спосіб перегляду та пошуку вакансій, а також дозволяє адміністраторам додавати нові вакансії.

Сторінка **transfer_client.php** дозволяє рекрутеру ініціювати процедуру передачі клієнта на вакансію. На цій сторінці відображається форма, яку рекрутер повинен заповнити інформацією про клієнта, включаючи його особисті дані, паспортні дані, документи, деталі про приїзд та інше.

Функція createClientTransfer відповідає за збереження інформації про передачу клієнта в базі даних. Вона приймає різні параметри, такі як ім’я клієнта, дата народження, номер паспорта, термін дії паспорта, документи, місто подачі на візу, номер телефону клієнта, дата приїзду, наявність власного житла, поточна країна перебування та ідентифікатор вакансії. Функція формує SQL-запит на вставку цих даних у таблицю client_transfers і виконує його за допомогою підготовленого запиту. Функція повертає ідентифікатор щойно створеного запису про передачу клієнта.

Функція linkPassportPhotoToTransfer використовується для зв’язування фотографій/сканів паспорта клієнта з записом про передачу клієнта в базі даних. Вона приймає ідентифікатор передачі клієнта та шлях до файлу фотографії/скану паспорта, а потім зберігає цю інформацію в таблиці client_passport_photos.

Код сторінки transfer_client.php виконує такі дії:

1) перевіряє, чи користувач увійшов у систему. Якщо ні, перенаправляє на сторінку входу;

2) отримує ідентифікатор вакансії з параметра URL ($_GET[‘vacancyId’]);

3) отримує назву вакансії за допомогою функції getVacancyNameById (не показана в наданому коді);

4) якщо отримано POST-запит, обробляє введені дані та файли:

– отримує значення полів форми, такі як ім’я клієнта, дата народження, номер паспорта, термін дії паспорта, документи, місто подачі на візу, номер

телефону клієнта, дата приїзду, наявність власного житла та поточна країна перебування;

– створює папку для збереження фотографій/сканів паспорта клієнта;

– обробляє завантажені файли фотографій/сканів паспорта та зберігає їх у створеній папці;

– викликає функцію createClientTransfer для збереження інформації про передачу клієнта в базі даних;

– викликає функцію linkPassportPhotoToTransfer для зв’язування фотографій/сканів паспорта з записом про передачу клієнта;

– перенаправляє користувача на головну сторінку (index.php);

5) відображає форму для введення інформації про клієнта, включаючи поля для імені, дати народження, номера паспорта, терміну дії паспорта, документів, міста подачі на візу, номера телефону, дати приїзду, наявності власного житла, поточної країни перебування та завантаження фотографій/сканів паспорта;

6) після заповнення форми та натискання кнопки «Передати клієнта» дані форми відправляються на сервер для обробки.

Загалом, сторінка transfer_client.php надає рекрутеру можливість ввести детальну інформацію про клієнта та ініціювати процедуру передачі клієнта на вакансію. Введені дані зберігаються в базі даних для подальшого використання та відстеження процесу передачі клієнта.

Сторінка **clients.php** призначена для адміністратора і відображає передачі вакансій у трьох різних категоріях: нові передачі, підтверджені передачі та завершені (оплачені) передачі.

Функції:

– getClientTransfers(): ця функція отримує нові передачі клієнтів з бази даних. Вона виконує SQL-запит, який вибирає ідентифікатор передачі, назву вакансії та ім’я користувача (рекрутера) з таблиць client_transfers, users та vacancies. Запит також виключає записи, які мають підтвердження або вже оплачені, використовуючи LEFT JOIN з таблицями client_transfer_confirmations та

client_transfer_payed. Результати запиту повертаються у вигляді масиву асоціативних масивів;

– getConfirmedClientTransfers(): ця функція отримує підтверджені передачі клієнтів з бази даних. Вона виконує SQL-запит, який вибирає ідентифікатор передачі, назву вакансії, ім’я користувача, дату підтвердження та суму винагороди з таблиць client_transfers, client_transfer_confirmations, users та vacancies. Результати запиту повертаються у вигляді масиву асоціативних масивів.

Код сторінки clients.php:

1) перевіряється, чи користувач увійшов у систему. Якщо ні, відбувається перенаправлення на сторінку входу;

2) викликаються функції getClientTransfers(), getConfirmedClientTransfers() та getPayedClientTransfers() (функція не показана в наданому коді) для отримання відповідних передач клієнтів;

3) включається заголовок сторінки за допомогою include ‘templates/header.php’;

4) відображається таблиця нових передач клієнтів з інформацією про ідентифікатор передачі, назву вакансії та ім’я користувача. Для кожної передачі надаються кнопки «Переглянути», «Підтвердити» та «Видалити» з відповідними посиланнями на сторінки view_transfer.php, confirm_transfer.php та delete_transfer.php;

5) відображається таблиця підтверджених передач клієнтів з інформацією про ідентифікатор передачі, назву вакансії, ім’я користувача, дату підтвердження та суму винагороди. Для кожної передачі надаються кнопки «Деталі» та «Здійснити виплату» з відповідними посиланнями на сторінки view_transfer.php та make_payment.php;

6) відображається таблиця завершених (оплачених) передач клієнтів з інформацією про ідентифікатор передачі, назву вакансії, ім’я користувача, дату підтвердження та суму винагороди. Для кожної передачі надаються кнопки «Деталі» та «Видалити» з відповідними посиланнями на сторінки view_transfer.php

та delete_payment.php;

7) включається підвал сторінки за допомогою include ‘templates/footer.php’.

Загалом, сторінка clients.php надає адміністратору зручний інтерфейс для перегляду та управління передачами клієнтів на різних етапах процесу: нові передачі, підтверджені передачі та завершені передачі. Адміністратор має можливість переглядати деталі передачі, підтверджувати нові передачі, здійснювати виплати за підтверджені передачі та видаляти передачі за потреби.

Сторінка **admin.php** є панеллю адміністратора, яка доступна лише для користувачів з рівнем доступу 1 (адміністратори). На цій сторінці адміністратор може переглядати список всіх користувачів системи, додавати нових користувачів, редагувати інформацію про користувачів та видаляти їх.

Функція getUsers() відповідає за отримання всіх користувачів з бази даних. Вона виконує простий SQL-запит SELECT * FROM users для вибірки всіх записів з таблиці users. Результати запиту повертаються у вигляді масиву асоціативних масивів, де кожен масив представляє одного користувача.

Код сторінки admin.php:

1) перевіряється, чи користувач увійшов у систему та чи має рівень доступу 1 (адміністратор). Якщо ні, відбувається перенаправлення на головну сторінку (index.php);

2) викликається функція getUsers() для отримання списку всіх користувачів;

3) вмикається заголовок сторінки за допомогою include ‘templates/header.php’;

4) відображається кнопка «Додати користувача», яка веде на сторінку add_user.php для додавання нового користувача;

5) створюється HTML-таблиця для відображення списку користувачів;

6) у циклі foreach перебираються всі користувачі зі змінної $users;

7) для кожного користувача створюється рядок таблиці з наступними даними:

– ідентифікатор користувача (ID);

– логін користувача;
– повне ім’я користувача (ПІБ);
– дата останнього входу користувача в систему;
– кнопки дій:

а) «виплати» – веде на сторінку payment_history.php з історією виплат для даного користувача;

б) «редагувати» – веде на сторінку edit_user.php для редагування інформації про користувача;

в) «видалити» – відображається як форма з прихованим полем id та кнопкою «Видалити». При натисканні на кнопку викликається підтвердження видалення, і якщо користувач підтверджує, форма відправляється на сторінку delete_user.php для видалення користувача;

8) включається підвал сторінки за допомогою include ‘templates/footer.php’.

Загалом, сторінка admin.php надає адміністратору зручний інтерфейс для управління користувачами системи. Адміністратор може переглядати список користувачів, додавати нових користувачів, редагувати інформацію про користувачів та видаляти їх за потреби. Також є можливість переглядати історію виплат для кожного користувача.

Сторінка **user_tasks.php** відображає список задач для поточного користувача. Вона дозволяє користувачам переглядати та керувати своїми задачами, а також створювати нові задачі.

Функція getUsersTasks($user_id, $sort_by) відповідає за отримання списку задач для конкретного користувача з бази даних. Вона приймає два параметри: $user_id – ідентифікатор користувача, для якого потрібно отримати задачі, та $sort_by – поле, за яким потрібно відсортувати результати. Функція виконує SQL-запит з умовою WHERE, щоб отримати задачі, в яких поточний користувач є виконавцем або створювачем. Результати сортуються за вказаним полем (created_at або deadline) у порядку, визначеному змінною $order.

Код сторінки user_tasks.php:

1) перевіряється, чи користувач увійшов у систему. Якщо ні, відбувається перенаправлення на сторінку входу;

2) оновлюється час останнього входу користувача за допомогою функції updateLastLogin();

3) отримується значення параметра sort_by з URL або встановлюється значення за замовчуванням created_at;

4) викликається функція getUsersTasks() для отримання списку задач поточного користувача з урахуванням сортування;

5) вмикається заголовок сторінки за допомогою include ‘templates/header.php’;

6) відображається кнопка «Створити нову задачу», яка веде на сторінку create_user_task.php для створення нової задачі;

7) відображається форма з випадаючим списком для вибору способу сортування задач (за терміном виконання або за датою додавання);

8) задачі розділяються на три категорії за статусом: «Актуальні», «В роботі» та «Виконані»;

9) для кожної категорії створюється окремий блок з відповідним заголовком;

10) у циклі foreach перебираються всі задачі зі змінної $tasks;

11) для кожної задачі перевіряється її статус. Якщо статус відповідає поточній категорії, задача відображається в блоці відповідної категорії;

12) кожна задача відображається з інформацією про її ID, заголовок, термін виконання та пріоритет;

13) при подвійному кліку на задачу викликається функція openEditTask() для відкриття вікна редагування задачі;

14) підключаються необхідні JavaScript-файли для забезпечення функціональності перетягування задач між категоріями (dragula.min.js та drag_tasks.js);

15) включається підвал сторінки за допомогою include ‘templates/footer.php’.

Загалом, сторінка user_tasks.php надає користувачам зручний інтерфейс для перегляду та управління своїми задачами. Задачі розділені на категорії за статусом, що дозволяє легко відстежувати їх прогрес. Користувачі можуть сортувати задачі за терміном виконання або датою додавання, а також створювати нові задачі. Функціональність перетягування задач між категоріями забезпечується за допомогою JavaScript-бібліотеки Dragula.

Сторінка **chat.php** є службовим чатом для користувачів системи. Вона дозволяє користувачам обмінюватися повідомленнями в реальному часі, а також вказувати витрачений час на виконання завдань.

Функція getChatMessages($userId = null) відповідає за отримання повідомлень чату з бази даних. Вона виконує SQL-запит для вибірки всіх повідомлень з таблиці chat_messages разом з іменами користувачів з таблиці users. Результати сортуються за часом відправки повідомлень. Якщо передано параметр $userId, функція фільтрує повідомлення, залишаючи лише ті, що належать вказаному користувачу.

Код сторінки chat.php:

1) перевіряється, чи користувач увійшов у систему. Якщо ні, відбувається перенаправлення на головну сторінку;

2) отримується ідентифікатор поточного користувача з сесії;

3) викликається функція getChatMessages() для отримання повідомлень чату, які належать поточному користувачу;

4) включається заголовок сторінки за допомогою include ‘templates/header.php’;

5) створюється прихований елемент <input> з класом user-id, який зберігає ідентифікатор поточного користувача;

6) відображається заголовок «Чат»;

7) створюється елемент з класом chat-window, який буде містити повідомлення чату;

8) створюється форма для відправки повідомлень з полем введення

повідомлення та полем для вказання витраченого часу;

9) створюється модальне вікно для редагування повідомлень з формою, яка містить поля для редагування тексту повідомлення та витраченого часу;

10) підключається файл js/chat.js, який містить JavaScript-код для обробки подій чату та оновлення повідомлень в реальному часі;

11) включається підвал сторінки за допомогою include ‘templates/footer.php’.

JavaScript-код (js/chat.js) виконує наступні функції:

1) при завантаженні сторінки викликається функція fetchMessages() для отримання та відображення повідомлень чату;

2) кожні 5 секунд викликається функція fetchMessages() для оновлення повідомлень чату;

3) при відправленні форми повідомлення викликається функція addChatMessage(), яка відправляє повідомлення на сервер і оновлює список повідомлень;

4) при кліку на кнопку редагування повідомлення відкривається модальне вікно з формою для редагування повідомлення;

5) при відправленні форми редагування повідомлення викликається функція updateChatMessage(), яка оновлює повідомлення на сервері та закриває модальне вікно;

6) при кліку на кнопку видалення повідомлення викликається функція deleteChatMessage(), яка видаляє повідомлення на сервері та оновлює список повідомлень.

Загалом, сторінка chat.php надає користувачам зручний інтерфейс для обміну повідомленнями в реальному часі та відстеження витраченого часу на виконання завдань. Функціональність редагування та видалення повідомлень доступна лише для автора повідомлення.

Розроблений інтерфейс забезпечує зручний та інтуїтивно зрозумілий спосіб взаємодії користувачів з системою.

В ході проектування інтерфейсу були враховані різні типи користувачів

системи: адміністратори, рекрутери та кандидати. Для кожної групи користувачів передбачено відповідні функціональні можливості та доступ до необхідних даних.

Основні сторінки вебзастосунку, такі як головна сторінка, сторінка вакансій, сторінка деталей вакансії, особистий кабінет рекрутера та сторінка адміністрування, забезпечують зручну навігацію та доступ до ключових функцій системи. Використання адаптивного дизайну дозволяє користувачам комфортно працювати з системою на різних пристроях.

Розроблений інтерфейс відповідає сучасним стандартам вебдизайну та забезпечує візуальну привабливість і консистентність. Використання фреймворку Bootstrap та мови програмування JavaScript дозволяє створити динамічний та інтерактивний користувацький досвід.

Особлива увага приділена функціональності, пов’язаній з процесом підбору вакансій, такій як фільтрація та пошук вакансій, перегляд деталей вакансії, передача клієнтів на вакансію та відстеження статусу.

Крім того, вебзастосунок включає додаткові функції для покращення взаємодії між користувачами, такі як система внутрішнього чату для обміну повідомленнями та можливість ставити завдання рекрутерам.

Спроектований інтерфейс вебзастосунку є результатом ретельного аналізу вимог, врахування потреб користувачів та застосування кращих практик вебдизайну. Він слугує міцною основою для подальшої розробки та впровадження системи підбору вакансій, забезпечуючи ефективну та зручну взаємодію між рекрутерами, кандидатами та адміністраторами платформи.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи було розглянуто ключові етапи проектування та розробки вебзастосунку для автоматизації процесу підбору вакансій.

Розділ розпочинається з обговорення архітектури вебзастосунку, яка базується на клієнт-серверній моделі та використовує багаторівневу структуру.

Обрана архітектура забезпечує масштабованість, гнучкість та можливість розширення функціональності системи в майбутньому.

Наступним важливим аспектом є розробка структури бази даних. Було спроектовано реляційну базу даних з використанням MySQL, яка містить таблиці для зберігання інформації про користувачів, вакансії, резюме, передачі клієнтів та інші необхідні дані. Структура бази даних оптимізована для ефективного доступу та управління даними.

Значна увага приділена проектуванню інтерфейсу користувача вебзастосунку. Розроблений інтерфейс відповідає сучасним стандартам вебдизайну та забезпечує зручну взаємодію користувачів з системою. Використання адаптивного дизайну дозволяє користувачам комфортно працювати з системою на різних пристроях. Розглянуто основні сторінки вебзастосунку, такі як головна сторінка, сторінка вакансій, сторінка деталей вакансії, особистий кабінет рекрутера та сторінка адміністрування, які забезпечують доступ до ключових функцій системи.

Результатом проведеної роботи є детальний план та специфікація вебзастосунку для автоматизації процесу підбору вакансій. Розроблена архітектура, структура бази даних та інтерфейс користувача слугують міцною основою для подальшої практичної реалізації системи.

Проектування та розробка вебзастосунку, описані в цьому розділі, демонструють комплексний підхід до створення ефективного інструменту для вирішення проблем підбору персоналу. Запропоновані рішення враховують як технічні вимоги, так і потреби користувачів, що є запорукою успішної реалізації та впровадження системи.

3 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ТА ЙОГО ТЕСТУВАННЯ

3.1 Реалізація бази даних

Реалізація бази даних для вебзастосунку є фундаментальним етапом, який вимагає детального планування та високої уваги до деталей. В процесі розробки бази даних було створено кілька ключових таблиць, кожна з яких відіграє важливу роль у функціонуванні застосунку. Таблиці, такі як **access_levels, chat_messages**, **client_passport_photos**, **client_transfers**, **client_transfer_confirmations**, **client_transfer_payed**, **users**, **user_financial_transactions**, **user_tasks**, **user_tasks_comments**, **vacancies**, **vacancy_media**, та **workers**, були спроектовані для зберігання різноманітної інформації, необхідної для ефективного управління даними користувачів, вакансій, кандидатів, фінансових транзакцій та інших аспектів вебзастосунку.

Кожна таблиця має своє унікальне призначення та структуру. Наприклад, таблиця **access_levels** призначена для зберігання рівнів доступу користувачів, визначаючи їхні права та обмеження у межах застосунку. Таблиця **chat_messages** використовується для зберігання повідомлень між користувачами, що дозволяє реалізувати функціонал чату. Таблиці, такі як **client_passport_photos** та **client_transfers**, забезпечують зберігання інформації про паспортні дані та переміщення клієнтів відповідно, що є критично важливим для застосунків, які працюють з персональними даними та вимагають високого рівня безпеки.

Особливу увагу було приділено взаємозв’язкам між таблицями. За допомогою механізму зовнішніх ключів (**FOREIGN KEY**) було створено зв’язки, які забезпечують цілісність та консистентність даних у базі. Наприклад, зв’язок між таблицями **client_transfers** та **users** дозволяє відстежувати, який користувач створив певний трансфер, тоді як зв’язок між **client_transfer_confirmations** та **client_transfers** гарантує, що кожне підтвердження трансферу чітко асоціюється з конкретним трансфером.

При розробці бази даних особливу увагу було приділено масштабованості та

оптимізації. Використання індексів дозволяє покращити швидкість виконання запитів, особливо в умовах великої кількості даних та високого навантаження на базу. Крім того, ретельний вибір типів даних для кожного поля та нормалізація структури бази даних сприяють ефективному використанню дискового простору та зниженню ризику дублювання даних.

Реалізація бази даних є процесом, який полягає в створенні фізичної структури бази даних на основі логічної моделі даних, виборі системи керування базами даних (СКБД), встановленні зв’язків між таблицями, заповненні бази даних даними, визначенні прав доступу та захисту бази даних. Реалізація бази даних вимагає врахування таких факторів, як продуктивність, надійність, масштабованість, безпека, та сумісність бази даних з іншими компонентами інформаційної системи.

Для реалізації бази даних для вебзастосунку підбору вакансій виконано наступні кроки:

1) *створення бази даних*: за допомогою інструментів MySQL, таких як MySQL Workbench або командний рядок, було створено нову базу даних з назвою, яка відповідає вебзастосунку підбору вакансій;

2) *створення таблиць*: відповідно до розробленої структури бази даних, було створено необхідні таблиці для зберігання різних сутностей, таких як користувачі, вакансії, резюме, передачі клієнтів тощо. Кожна таблиця містить відповідні стовпці для зберігання атрибутів сутностей, а також первинні та зовнішні ключі для встановлення зв’язків між таблицями. При створенні таблиць було враховано типи даних, обмеження та індекси для забезпечення цілісності та продуктивності бази даних;

3) *встановлення зв’язків між таблицями*: за допомогою зовнішніх ключів було встановлено зв’язки між таблицями, що відображають логічні відношення між сутностями. Наприклад, таблиця «client_transfers» містить зовнішній ключ «user_id», який посилається на первинний ключ таблиці «users», встановлюючи зв’язок між передачею клієнта та користувачем, який здійснив цю передачу;

4) *створення індексів*: для підвищення продуктивності виконання запитів до бази даних було створено індекси на ключових стовпцях таблиць. Індекси дозволяють швидко знаходити необхідні записи за значеннями певних стовпців, що особливо важливо для таблиць з великою кількістю даних;

5) *наповнення бази даних початковими даними*: для тестування та початкової конфігурації вебзастосунку було додано в базу даних деякі початкові дані. Наприклад, було додано користувачів з різними ролями (адміністратор, рекрутер), а також створено кілька тестових вакансій та резюме. Початкові дані дозволяють перевірити коректність роботи вебзастосунку та надають основу для подальшого наповнення бази даних реальними даними;

6) *оптимізація та налаштування продуктивності*: для забезпечення оптимальної продуктивності бази даних було виконано налаштування параметрів MySQL, таких як розмір буфера, кеш запитів, розмір пакета тощо. Також було проаналізовано та оптимізовано запити до бази даних, щоб мінімізувати час виконання та використання ресурсів;

7) *резервне копіювання та відновлення*: для забезпечення цілісності та збереження даних було налаштовано регулярне резервне копіювання бази даних. Резервні копії дозволяють відновити базу даних у разі збоїв або втрати даних, забезпечуючи безперервність роботи вебзастосунку.

Реалізована база даних була ретельно протестована, щоб переконатися у коректності створених таблиць, зв’язків та запитів. Тести включали перевірку цілісності даних, коректності виконання основних операцій (вставка, оновлення, видалення) та продуктивності запитів.

Після успішного тестування база даних була інтегрована з вебзастосунком з підбору вакансій. Завдяки використанню мови програмування PHP та розширення PHP Data Objects (PDO), було встановлено з’єднання з базою даних та реалізовано взаємодію між вебзастосунком та базою даних.

База даних забезпечує надійне та ефективне зберігання та управління даними вебзастосунку. Вона дозволяє зберігати інформацію про користувачів, вакансії,

резюме, передачі клієнтів та інші необхідні дані, а також забезпечує швидкий доступ та обробку цих даних для забезпечення функціональності вебзастосунку.

Слід зазначити, що в процесі розробки та експлуатації вебзастосунку база даних може потребувати подальших оптимізацій та модифікацій відповідно до зміни вимог та зростання обсягів даних. Регулярний моніторинг продуктивності, оптимізація запитів та налаштування параметрів бази даних дозволять підтримувати високу ефективність та масштабованість системи.

Таким чином, реалізація бази даних є критично важливим етапом у розробці вебзастосунку підбору вакансій. Правильно спроектована та реалізована база даних закладає міцний фундамент для ефективного функціонування та розвитку системи, забезпечуючи надійне зберігання, організацію та доступ до даних.

3.2 Реалізація фронтенду вебзастосунку

Фронтенд вебзастосунку є тією частиною, яка відображається в браузері користувача та забезпечує йому інтерфейс для взаємодії з вебзастосунком. Для побудови користувацького інтерфейсу використовувався HTML5 та CSS3. HTML – мова розмітки, яка використовується для створення структури вебсторінок. HTML дозволяє використовувати різні елементи, такі як заголовки, абзаци, списки, таблиці, форми, посилання, зображення тощо, для представлення інформації на вебсторінці. CSS – мова стилів, яка використовується для задання зовнішнього вигляду вебсторінок. CSS дозволяє використовувати різні властивості, такі як кольори, шрифти, відступи, вирівнювання, анімації тощо, для зміни візуального оформлення HTML-елементів.

Для прискорення та полегшення розробки дизайну був використаний фреймворк Bootstrap версії 5. Він містить заздалегідь підготовлені шаблони та компоненти інтерфейсу з адаптивною верстkою, що дозволило швидко створити макет системи під різні типи пристроїв.

JavaScript – мова програмування, яка використовується для створення клієнтської частини вебзастосунку. Вона відіграє вирішальну роль у створенні

інтерактивних вебсторінок, дозволяючи реалізовувати динамічну взаємодію без перезавантаження сторінки. Це може включати валідацію форм на стороні клієнта, анімації, слайдери, асинхронні запити до сервера через AJAX для оновлення частин сторінки, і багато іншого. Використання сучасних бібліотек та фреймворків на базі JavaScript, таких як jQuery або Vue.js, може подальше поліпшити користувацький досвід і спростити розробку. За допомогою jQuery реалізовані різноманітні події, запити до серверу, маніпуляції з DOM-елементами тощо.

Дані з серверної частини (MySQL+PHP) запитуються та обробляються за допомогою асинхронних Ajax-запитів jQuery та Fetch API, з подальшим динамічним оновленням вмісту сторінок за результатами запитів.

Загалом використання сучасних вебтехнологій дозволило створити швидкий, гнучкий та зручний у використанні інтерфейс користувача відповідно до визначених вимог.

PHP служить мовою програмування на серверній стороні і відіграє ключову роль у динамічній взаємодії з базою даних (MySQL) та генерації HTML-контенту на основі запитів користувача. Використання сучасних підходів у PHP, таких як об’єктно-орієнтоване програмування (ООП), може значно підвищити якість коду та спростити подальшу підтримку вебзастосунку.

Gulp – інструмент, який використовується для автоматизації різних задач при розробці фронтенду, таких як компіляція, мініфікація, конкатенація, перезавантаження браузера тощо.

Docker – інструмент, який використовується для створення, запуску та розгортання вебзастосунку в ізольованих контейнерах, які містять усі необхідні залежності та налаштування.

MySQL служить за систему управління базами даних, що забезпечує зберігання, оновлення та вибірку даних для вебзастосунку. Оптимізація структури бази даних, індексація та правильне формування запитів є критично важливими для забезпечення високої продуктивності та швидкості роботи застосунку.

Безпека є ключовим аспектом у розробці вебзастосунків. Це включає захист

від SQL ін’єкцій через параметризовані запити, захист від XSS (Cross-Site Scripting) атак шляхом екранування вводу користувачів, використання HTTPS для шифрування даних, реалізацію аутентифікації та авторизації користувачів, а також регулярне оновлення всіх компонентів системи для усунення вразливостей.

Розробка фронтенду вебзастосунку з використанням PHP, HTML, CSS (з Bootstrap), JavaScript, і MySQL вимагає комплексного підходу, що включає вибір правильних технологій, оптимізацію продуктивності, забезпечення адаптивності та високого рівня безпеки. Це дозволить створити зручний, швидкий та безпечний вебзастосунок, який задовольнить потреби користувачів та забезпечить їх позитивний досвід використання.

Головна сторінка вебзастосунку відображає список актуальних вакансій, доступних для користувача (див. рис. 3.1).

Головна сторінка містить таблицю з наступними полями:

– назва вакансії;
– місто роботи;
– кількість доступних місць;
– стать, для якої призначена вакансія.

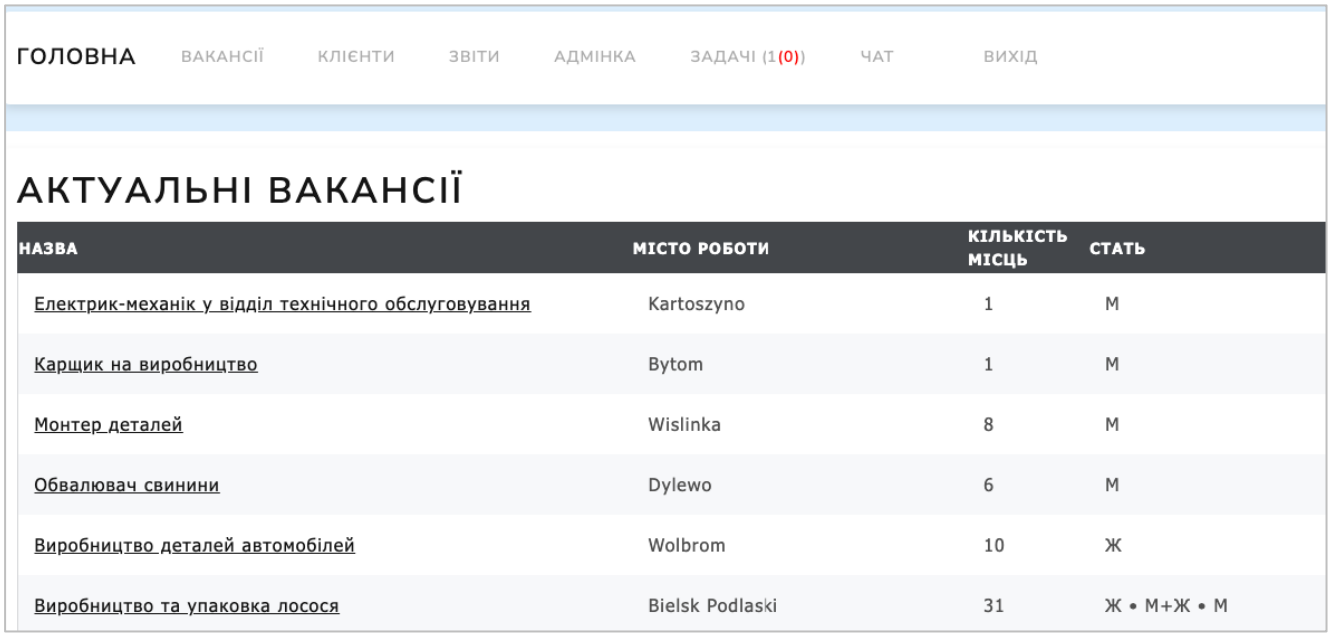

Рисунок 3.1 – Скріншот головної сторінки

Кожен рядок таблиці відповідає окремій вакансії. Назва вакансії є посиланням, яке веде на сторінку з детальною інформацією про вакансію.

Для отримання списку вакансій з бази даних використовується функція getVacancies(), яка виконує SQL запит до таблиці vacancies та повертає результат у вигляді асоціативного масиву. У разі виникнення помилки при виконанні запиту, функція записує повідомлення про помилку в лог та повертає порожній масив.

Отриманий список вакансій передається в шаблон головної сторінки, де за допомогою циклу foreach відбувається ітерація по кожному елементу масиву та формування рядків таблиці з даними про вакансії. Для запобігання XSS атак, всі дані, які виводяться на сторінку, проходять через функцію htmlspecialchars(), яка замінює спеціальні символи на їх HTML-еквіваленти.

Шаблон головної сторінки також включає в себе шапку та підвал сайту, які зберігаються в окремих файлах header.php та footer.php відповідно. Це дозволяє зменшити дублювання коду та спростити внесення змін до дизайну сайту.

Перед відображенням головної сторінки відбувається перевірка, чи користувач авторизований в системі. Якщо користувач не авторизований, його буде перенаправлено на сторінку входу. Для цього використовується функція isLoggedIn(), яка перевіряє наявність ідентифікатора користувача в сесії.

Крім того, при кожному відвідуванні головної сторінки оновлюється час останнього входу користувача в систему за допомогою функції updateLastLogin(), яка приймає ідентифікатор користувача та оновлює відповідне поле в базі даних.

Сторінка «Актуальні вакансії» відображає список доступних вакансій та надає функціонал для пошуку вакансій за назвою, містом реєстрації або містом роботи (див. рис. 3.2).

Сторінка містить наступні елементи:

– заголовок «Актуальні вакансії»;

– кнопка «Додати вакансію» (відображається лише для користувачів з рівнем доступу 1);

– поле введення для пошуку вакансій;

– список вакансій.

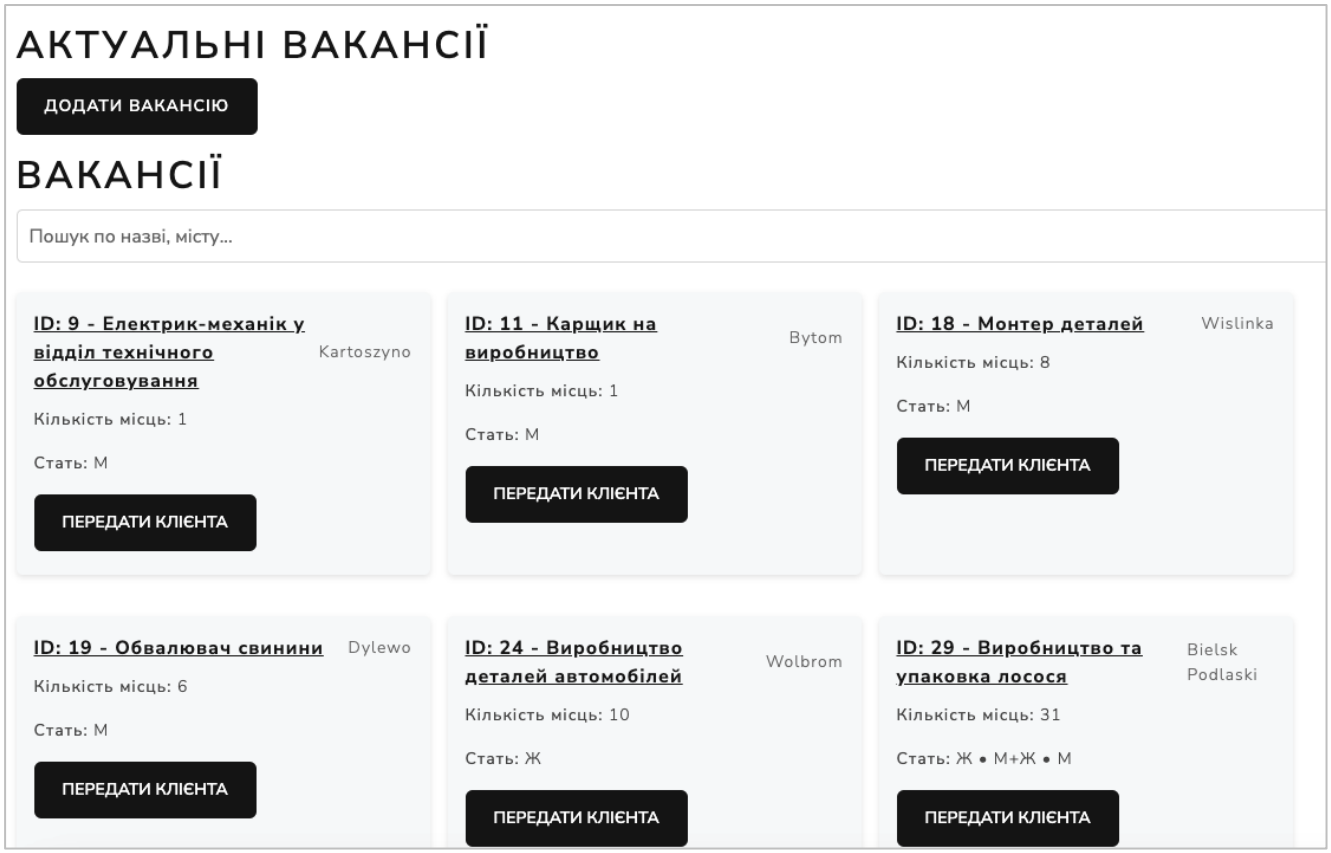

Рисунок 3.2 – Скріншот сторінки «Актуальні вакансії»

Кожна вакансія у списку відображається у вигляді окремого блоку, який містить:

– назву вакансії з посиланням на сторінку деталей вакансії;

– місто роботи;

– кількість доступних місць;

– стать, для якої призначена вакансія;

– кнопку «Передати клієнта» для переходу на сторінку передачі клієнта на цю вакансію.

Для реалізації функціоналу пошуку вакансій використовується JavaScript. При введенні тексту в поле пошуку, викликається функція filterVacancies(), яка фільтрує список вакансій на основі введеного тексту. Фільтрація відбувається за

назвою вакансії, містом реєстрації та містом роботи. Відфільтрований список вакансій потім передається у функцію renderVacancies(), яка оновлює відображення списку вакансій на сторінці.

Для створення елементів вакансій використовується шаблон <template>, який містить структуру HTML для одного елемента вакансії. Функція createVacancyElement() клонує цей шаблон, заповнює його даними про вакансію та повертає готовий елемент для додавання до списку вакансій.

При завантаженні сторінки, викликається функція renderVacancies() з початковим списком всіх вакансій, отриманим з PHP через json_encode(). Це забезпечує відображення повного списку вакансій при першому завантаженні сторінки.

Перед відображенням сторінки також відбувається перевірка авторизації користувача та оновлення часу останнього входу, як і на головній сторінці.

Сторінка «Додати нову вакансію» дозволяє користувачам з відповідним рівнем доступу створювати нові вакансії, вводячи необхідну інформацію та завантажуючи пов’язані медіафайли (див. рис. 3.3).

Сторінка містить форму з наступними полями введення:

– назва – текстове поле для введення назви вакансії;

– місто виконання робіт – текстове поле для вказання міста, де буде виконуватися робота;

– кількість місць – числове поле для введення кількості доступних місць для вакансії;

– стать – текстове поле для вказання статі, для якої призначена вакансія;

– докладний опис для клієнта – текстова область для введення детального опису вакансії. Поруч з цим полем розташована кнопка «Згенерувати опис з ШІ», яка дозволяє автоматично згенерувати опис за допомогою штучного інтелекту на основі введених даних про вакансію;

– завантаження медіафайлів – поле для вибору та завантаження пов’язаних з вакансією медіафайлів, таких як зображення чи документи.

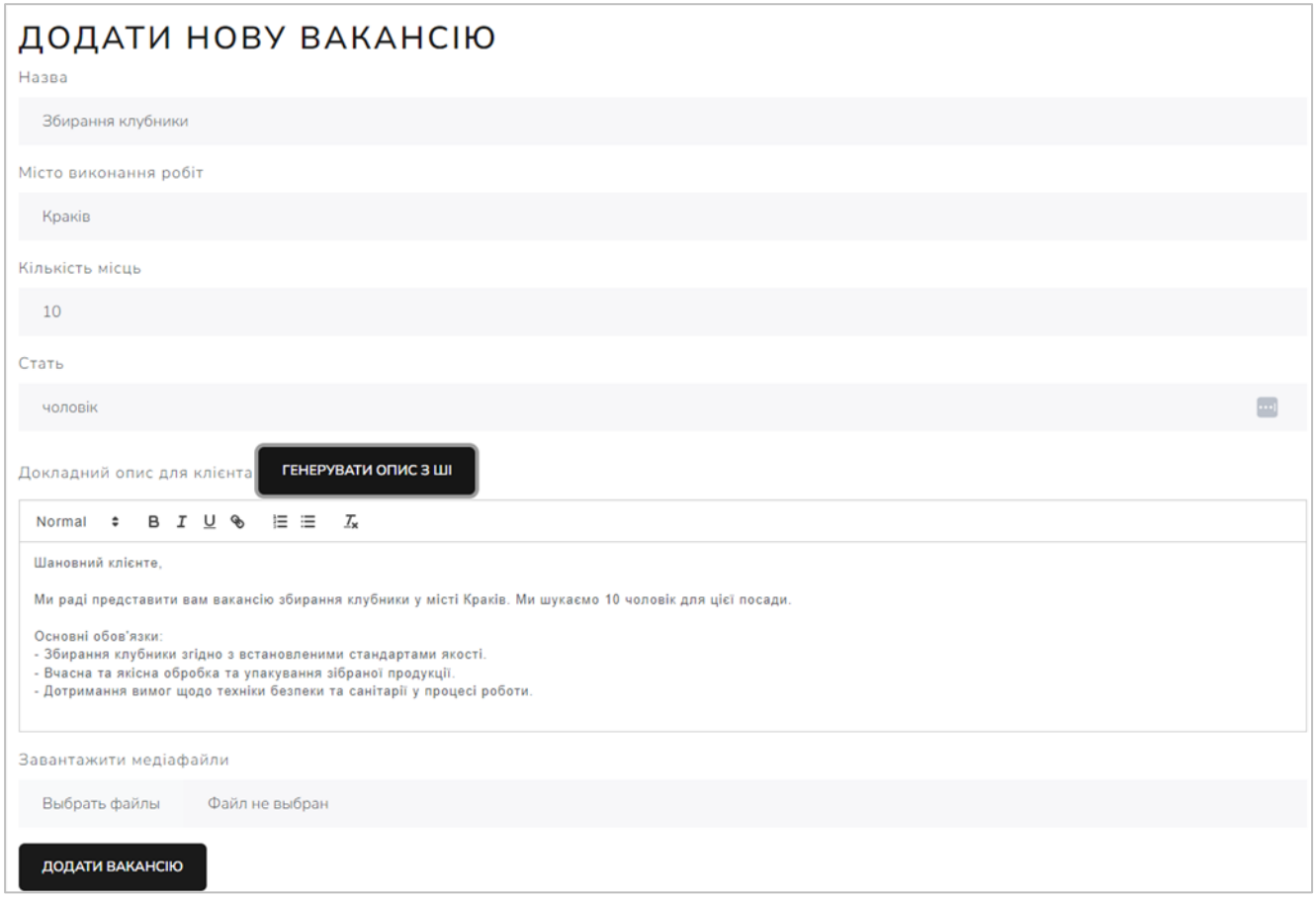

Рисунок 3.3 – Скріншот сторінки «Додати нову вакансію»

Для реалізації функціоналу генерації опису за допомогою ШІ використовується API ChatGPT. При натисканні на кнопку «Згенерувати опис з ШІ», викликається асинхронна функція JavaScript, яка збирає дані з полів введення (назва, місто виконання робіт, кількість місць, стать) і формує запит до серверного скрипту generate_description.php. Цей скрипт відправляє запит до API ChatGPT з зібраними даними та отримує згенерований опис вакансії. Потім згенерований опис вставляється в текстову область «Докладний опис для клієнта» за допомогою бібліотеки Quill, яка використовується для редагування тексту.

Для завантаження медіафайлів використовується стандартне поле введення типу «file» з можливістю множинного вибору файлів. При виборі файлів, їх список відображається під полем введення з можливістю видалення окремих файлів.

При відправці форми, дані вакансії та завантажені файли передаються на сервер для подальшої обробки та збереження. Сервер генерує унікальну назву

папки для зберігання завантажених файлів, створює запис про вакансію в базі даних та зберігає шляхи до завантажених файлів, пов’язуючи їх з відповідною вакансією.

Перед відображенням сторінки також відбувається перевірка авторизації користувача, щоб переконатися, що тільки авторизовані користувачі з відповідним рівнем доступу можуть створювати нові вакансії.

Сторінка «Клієнти» (clients.php) відображає інформацію про передачі клієнтів на вакансії та надає функціонал для управління цими передачами. На сторінці є три секції: нові передачі, підтверджені передачі та завершені передачі.

Секція «Передачі клієнтів (нові)» містить таблицю з наступними полями (див. рис. 3.4):

– ID передачі;
– назва вакансії;
– логін користувача;
– дії (перегляд, підтвердження, видалення).

Для кожної нової передачі надаються кнопки «Переглянути» (для перегляду деталей передачі), «Підтвердити» (для підтвердження передачі) та «Видалити» (для видалення передачі).

Рисунок 3.4 – Скріншот секції «Передачі клієнтів (нові)»

Секція «Передачі клієнтів (підтверджені)» містить таблицю з наступними полями (див. рис. 3.5):

– ID передачі;
– назва вакансії;
– логін користувача;

– дата виходу на роботу;
– сума винагороди;
– дії (деталі, здійснення виплати).

Для кожної підтвердженої передачі надаються кнопки «Деталі» (для перегляду деталей передачі) та «Здійснити виплату» (для здійснення виплати винагороди).

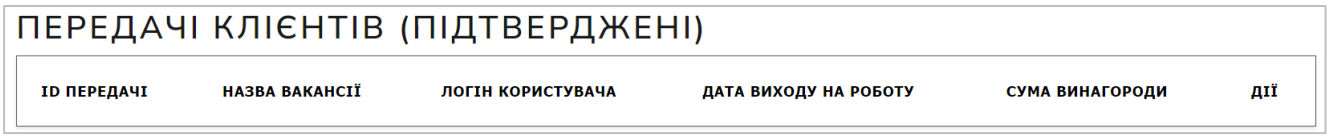

Рисунок 3.5 – Скріншот секції «Передачі клієнтів (підтверджені)»

Секція «Передачі клієнтів (завершені)» містить таблицю з наступними полями (див. рис. 3.6):

– ID передачі;
– назва вакансії;
– логін користувача;
– дата виходу на роботу;
– сума винагороди;
– дії (деталі, видалення).

Для кожної завершеної передачі надаються кнопки «Деталі» (для перегляду деталей передачі) та «Видалити» (для видалення передачі).

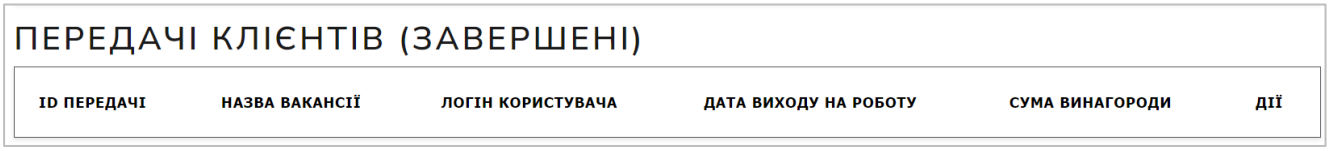

Рисунок 3.6 – Скріншот секції «Передачі клієнтів (завершені)»

Для отримання даних про передачі клієнтів використовуються наступні функції:

1) getClientTransfers() – отримує список нових передач клієнтів, які ще не були підтверджені або оплачені;

2) getConfirmedClientTransfers() – отримує список підтверджених передач клієнтів;

3) getPayedClientTransfers() – отримує список завершених (оплачених) передач клієнтів.

Кожна з цих функцій виконує SQL-запит до бази даних, який об’єднує таблиці client_transfers, users, vacancies, client_transfer_confirmations та client_transfer_payed для отримання необхідних даних. Результати запитів повертаються у вигляді масиву асоціативних масивів.

Перед відображенням сторінки відбувається перевірка авторизації користувача, як і на попередніх сторінках.

Сторінка «Передача клієнта» призначена для передачі інформації про клієнта на вакансію. На цій сторінці користувач може ввести детальну інформацію про клієнта та завантажити фото/скани закордонного паспорта клієнта.

Форма на сторінці містить наступні поля (див. рис. 3.7):

– ФІО клієнта (транслітерація);
– назва вакансії (автоматично заповнюється на основі вибраної вакансії);
– дата народження клієнта;
– номер і серія закордонного паспорта;
– термін дії закордонного паспорта;
– документи клієнта (безвіз, віза тощо);
– місто подачі на візу;
– номер телефону клієнта;
– дата приїзду клієнта;
– інформація про наявність власного житла у клієнта;
– країна, в якій зараз перебуває клієнт;
– фото/скани закордонного паспорта клієнта (можливість завантаження декількох файлів).

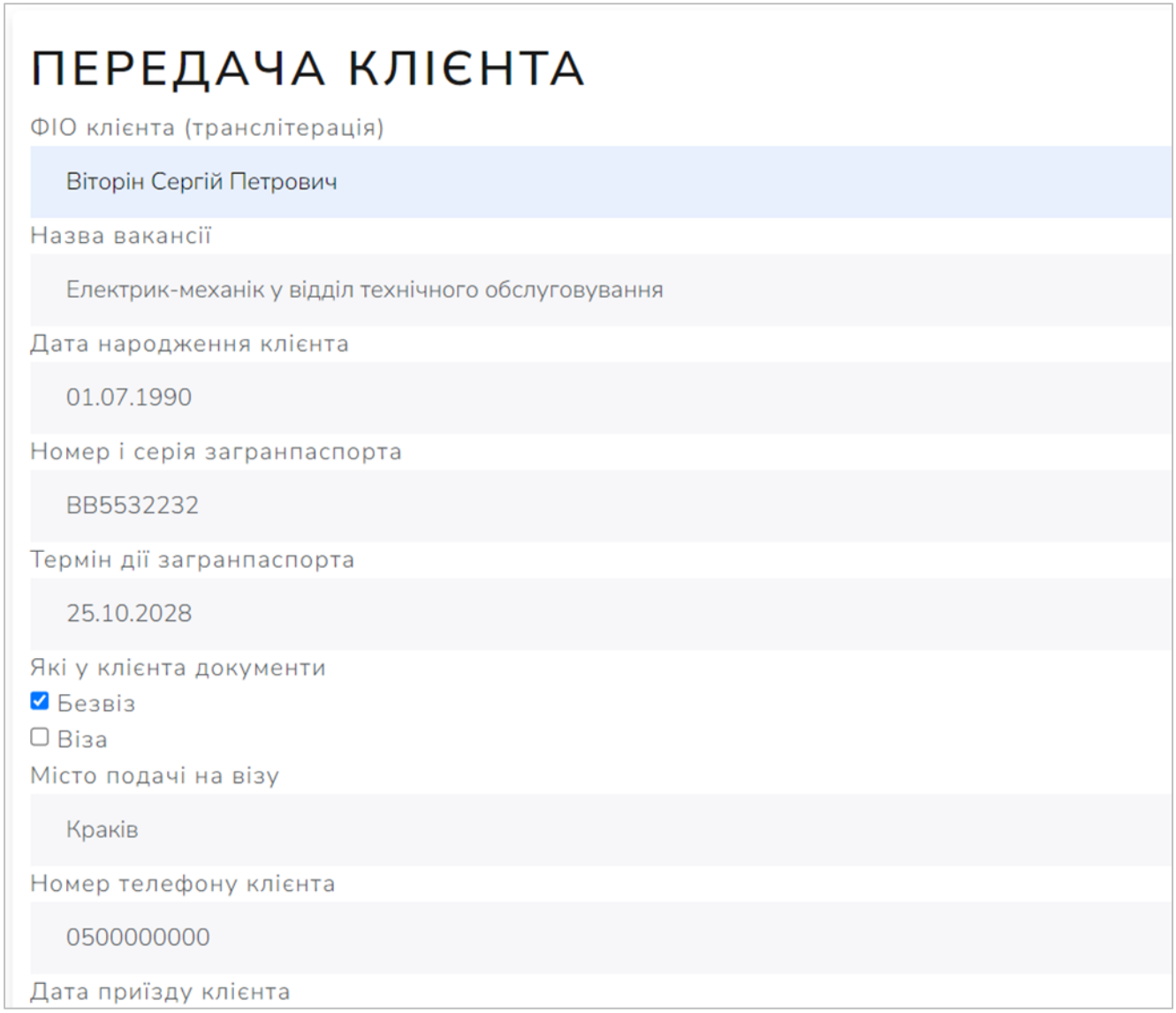

Рисунок 3.7 – Скріншот сторінки «Передача клієнта»

При відправці форми виконуються наступні дії:

1) обробка введених даних та файлів;

2) створення нової директорії для збереження завантажених файлів (фото/сканів паспорта);

3) переміщення завантажених файлів у створену директорію;

4) збереження інформації про передачу клієнта в базі даних за допомогою функції createClientTransfer();

5) зв’язування завантажених файлів (фото/сканів паспорта) з передачею клієнта за допомогою функції linkPassportPhotoToTransfer().

Функція getVacancyNameById() використовується для отримання назви вакансії на основі її ідентифікатора. Вона виконує SQL-запит до бази даних для отримання назви вакансії за вказаним ідентифікатором.

Функція createClientTransfer() відповідає за створення нового запису про передачу клієнта в базі даних. Вона приймає всі необхідні дані про клієнта та вакансію, підготовлює SQL-запит на вставку нового запису в таблицю client_transfers та виконує його. Функція повертає ідентифікатор новоствореної передачі клієнта.

Функція linkPassportPhotoToTransfer() зв’язує завантажені файли (фото/скани паспорта) з передачею клієнта. Вона приймає ідентифікатор передачі клієнта та шлях до файлу, підготовлює SQL-запит на вставку нового запису в таблицю client_passport_photos та виконує його.

Після успішної обробки форми та збереження даних, користувач перенаправляється на головну сторінку (index.php).

Сторінка «Деталі передачі клієнта» (view_transfer.php) відображає детальну інформацію про передачу клієнта. На цій сторінці адміністратор може переглянути всі дані, введені при передачі клієнта, а також фото/скани паспорта клієнта (див. рис. 3.8).

Сторінка містить дві основні секції:

1) таблиця з деталями передачі клієнта;
2) фото/скани паспорта клієнта.

Таблиця з деталями передачі клієнта відображає наступну інформацію:

– ID передачі;
– логін користувача, який здійснив передачу;
– назва вакансії, на яку передається клієнт;
– ФІО клієнта;
– дата народження клієнта;
– номер паспорта клієнта;
– термін дії паспорта клієнта;

– документи клієнта;
– місто подачі на візу;
– номер телефону клієнта;
– дата приїзду клієнта;
– інформація про наявність власного житла у клієнта;
– країна перебування клієнта.

Секція з фото/сканами паспорта клієнта відображає завантажені файли. Кожне фото/скан є зображенням з можливістю завантаження при натисканні на нього.

Рисунок 3.8 – Скріншот сторінки «Деталі передачі клієнта»

В нижній частині сторінки розташовані кнопки дій:

– «Повернутись» – повертає користувача на сторінку зі списком передач клієнтів;

– «Підтвердити» – підтверджує передачу клієнта (перенаправляє на сторінку «confirm_transfer.php» з передачею ID передачі);

– «Видалити» – видаляє передачу клієнта (перенаправляє на сторінку

«delete_transfer.php» з передачею ID передачі).

Для отримання деталей передачі клієнта використовується функція getTransferDetails(), яка приймає ID передачі та виконує SQL-запит для отримання всіх необхідних даних з бази даних. Запит об’єднує таблиці client_transfers, users та vacancies для отримання повної інформації про передачу.

Для отримання фото/сканів паспорта клієнта використовується функція getTransferPhotos(), яка приймає ID передачі та виконує SQL-запит для отримання всіх пов’язаних файлів з таблиці client_passport_photos.

Перед відображенням сторінки виконується перевірка авторизації користувача та наявності переданого ID передачі. Якщо ID не передано або передача не знайдена, користувач перенаправляється на сторінку зі списком передач клієнтів.

Сторінка «Підтвердження передачі клієнта» (confirm_transfer.php) дозволяє адміністратору підтвердити передачу клієнта та вказати винагороду за цю передачу (див. рис. 3.9).

На сторінці відображається наступна інформація:

– назва вакансії, на яку передається клієнт;
– ім’я клієнта;
– ім’я рекрутера, який здійснив передачу.

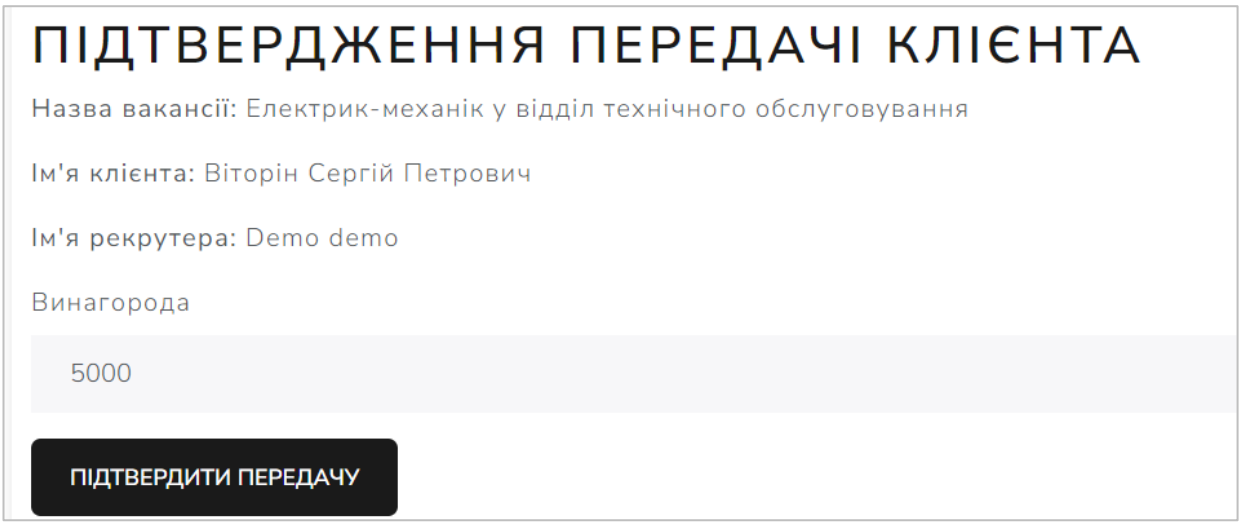

Рисунок 3.9 – Скріншот сторінки «Підтвердження передачі клієнта»

Крім того, сторінка містить форму для введення винагороди за передачу клієнта. Адміністратор може ввести суму винагороди в спеціальне поле та підтвердити передачу, натиснувши на кнопку «Підтвердити передачу».

При підтвердженні передачі виконуються наступні дії:

1) отримання ID передачі з URL-параметра;
2) отримання деталей передачі за допомогою функції getTransferDetailss();
3) отримання імені рекрутера за допомогою функції getUserNameById();
4) перевірка, чи передача існує, якщо ні, повідомлення про помилку;
5) обробка відправленої форми (при натисканні на кнопку «Підтвердити передачу»):
   – отримання ID користувача (адміністратора) з сесії;
   – отримання введеної винагороди з поля форми;
   – виклик функції confirmClientTransfer() для збереження підтвердження передачі в базі даних;
   – перенаправлення на сторінку зі списком передач клієнтів.

Функція getTransferDetailss() виконує SQL-запит до бази даних для отримання деталей передачі клієнта. Запит об’єднує таблиці client_transfers, vacancies та users для отримання повної інформації про передачу, включаючи назву вакансії та ім’я рекрутера.

Функція getUserNameById() отримує повне ім’я користувача за його ID. Вона виконує SQL-запит до таблиці users для отримання поля full_name на основі переданого ID користувача.

Функція confirmClientTransfer() зберігає підтвердження передачі клієнта в базі даних. Вона приймає ID передачі, ID користувача (адміністратора) та введену винагороду. Функція виконує SQL-запит на вставку нового запису в таблицю client_transfer_confirmations з відповідними даними.

Перед відображенням сторінки виконується перевірка авторизації користувача та наявності переданого ID передачі. Якщо ID не передано або передача не знайдена, виводиться повідомлення про помилку.

3.3 Розгортання вебзастосунку на хостингу та оцінка продуктивності

Розгортання вебзастосунку на хостингу є процесом, який дозволяє зробити вебзастосунок доступним для користувачів через Інтернет. Воно вимагає вибору хостинг-провайдера, налаштування сервера, передачі файлів вебзастосунку на сервер, налаштування бази даних, тестування та моніторингу вебзастосунку.

Для розгортання вебзастосунку підбору вакансій вибрано AWS, хмарну платформу, яка надає широкий спектр сервісів для розробки, розгортання та підтримки вебзастосунків будь-якого масштабу та складності. AWS пропонує високу продуктивність, надійність, масштабованість, безпеку, гнучкість та доступність для вебзастосунків.

Завдяки хмарній інфраструктурі AWS вебзастосунок масштабується в автоматичному режимі під поточне навантаження. При зростанні кількості користувачів та запитів автоматично виділяються додаткові обчислювальні ресурси. Це забезпечує високу продуктивність та цілодобову безперебійну роботу системи.

На віртуальній машині EC2 було налаштовано середовище Linux, встановлено вебсервер Apache, СУБД MySQL та інтерпретатор PHP відповідно до вимог проекту.

Було використано Docker, інструмент, який дозволяє створювати, запускати та розгортати вебзастосунки в ізольованих контейнерах, які містять усі необхідні залежності та налаштування. Docker спрощує процес розгортання, оскільки він гарантує, що вебзастосунок працюватиме однаково на будь-якому сервері, який підтримує Docker. Налаштування та взаємозв’язки контейнерів описано у файлі docker-compose.yml.

Для розгортання вебзастосунку виконано наступні кроки:

– створено обліковий запис на AWS та активовано тарифний план, який дозволяє використовувати деякі сервіси AWS безкоштовно протягом 12 місяців;

– створено віртуальну машину (VM) на AWS за допомогою сервісу EC2

(Elastic Compute Cloud), який дозволяє запускати віртуальні сервери на хмарі. Вибрано тип VM t2.micro, який входить у безкоштовний тарифний план, та операційну систему Ubuntu 20.04 LTS. Налаштовано правила брандмауера (security groups) для дозволу доступу до VM по SSH (порт 22), HTTP (порт 80) та Adminer (порт 8095). Збережено ключ SSH для підключення до VM;

– проведено підключення до VM за допомогою SSH та встановлено Docker та Docker Compose на VM за допомогою команд:

а) sudo apt update;

б) sudo apt install docker.io docker-compose;

– створено файл docker-compose.yaml на VM за допомогою команди «nano docker-compose.yaml» та скопійовано вміст файлу docker-compose.yaml у файл на VM. Збережено і закрито файл за допомогою комбінації клавіш Ctrl+O та Ctrl+X;

– запущено Docker Compose за допомогою команди «sudo docker-compose up -d». Ця команда створила та запустила три контейнери: ew-php81, ew-adminer та ew-db, які відповідають за PHP, Adminer та MySQL відповідно. Контейнери були пов’язані між собою за допомогою мережі Docker та налаштовані згідно з файлом docker-compose.yaml;

– перевірено статус контейнерів за допомогою команди «sudo docker ps». Ця команда показала, що всі три контейнери працюють та слухають відповідні порти;

– відкрито браузер та виконано перехід за адресою http://<public_ip>:8096, де <public_ip> – це публічна IP-адреса VM, яку можна дізнатися з панелі керування AWS. Відобразилася головна сторінка вебзастосунку, яка була згенерована за допомогою PHP.

– виконано перехід за адресою http://<public_ip>:8095, відобразився інтерфейс Adminer, який дозволяє керувати базою даних MySQL. Переходимо до бази даних, використовуючи ім’я користувача root, пароль «DBROOTPASSWORD та ім′я бази даних{DB_NAME}», які були вказані в файлі docker-compose.yaml. Відобразились таблиці та дані в базі даних.

Файл docker-compose.yml визначає конфігурацію, створення та взаємозв’язки контейнерів Docker для компонентів вебзастосунку. Він дозволяє однією командою запустити всі потрібні сервіси.

Зокрема описано 3 основних контейнери:

1) *php81* – контейнер вебсервера Apache з підключенням PHP 8.1.:
   – розгортається з Dockerfile та містить вебзастосунок;
   – маппінг портів на 8096 для доступу ззовні;
   – зв’язок з MySQL базою даних;
2) *adminer* – контейнер з вебінтерфейсом до бази даних Adminer:
   – запускається на порту 8095;
   – зв’язок з сервером бази даних MySQL;
3) *db* – контейнер сервера бази даних MySQL:
   – образ останньої версії mysql;
   – налаштування кодування та аутентифікації;
   – визначення змінних доступу та ініціалізації БД;
   – зв’язок з php81 та adminer.

Docker-compose дозволяє швидко розгорнути увесь стек вебзастосунку для роботи. В подальшому можна додавати інші потрібні сервіси та контейнери.

Для оцінки продуктивності вебзастосунку використані наступні методи та інструменти:

1) *вебінспектор браузера*: цей інструмент дозволяє вимірювати час завантаження вебсторінок, кількість запитів, розмір файлів, швидкість рендерингу, використання пам’яті та інші параметри, які впливають на продуктивність вебзастосунку. Використано вебінспектор браузера Google Chrome, який можна відкрити за допомогою клавіші F12 або правого кліку та вибору пункту «Перевірити». У вкладці «Network» перезавантажено сторінку вебзастосунку. В результаті сторінка завантажувалася протягом 1.2 секунди, використовуючи 16 запитів та 1.1 МБ даних.

Найбільше часу займав запит до файлу bootstrap.min.css, який мав розмір 155

КБ та завантажувався протягом 0.6 секунди. Це можна пояснити тим, що цей файл містить усі стилі Bootstrap, які використовуються для створення інтерфейсу вебзастосунку. Щоб покращити продуктивність, можна спробувати зменшити розмір цього файлу за допомогою мініфікації та компресії, або використати CDN-посилання замість локального файлу;

2) *вебсервіси AWS*: ці інструменти дозволяють вимірювати та моніторити різні аспекти продуктивності вебзастосунку, такі як використання ресурсів, навантаження, доступність, помилки, латентність тощо. Використано наступні вебсервіси AWS для оцінки продуктивності вебзастосунку:

– *CloudWatch* – цей сервіс дозволяє збирати, аналізувати та візуалізувати метрики продуктивності вебзастосунку, такі як CPU, пам’ять, диск, мережа, запити, відповіді тощо. Використано CloudWatch для створення графіків та сповіщень, які показують стан вебзастосунку в реальному часі та сповіщають про будь-які аномалії або проблеми;

– *X-Ray* – цей сервіс дозволяє відстежувати та аналізувати запити, які надходять до вебзастосунку, та виявляти причини затримок, помилок, або неефективності. Використано X-Ray для створення трасувань та діаграм, які показують, як запити проходять через різні компоненти вебзастосунку, такі як контейнери, API, база даних тощо, та як вони займають немало часу;

– *Load Balancer* – цей сервіс дозволяє розподіляти навантаження між декількома серверами, які обслуговують вебзастосунок, та забезпечувати його масштабованість, доступність, та відмовостійкість. Використано Load Balancer для створення балансувальника навантаження, який автоматично перенаправляє запити до найменш завантаженого сервера, а також перевіряє стан серверів та виключає непрацюючі або проблемні сервери з пулу.

За допомогою цих методів та інструментів було оцінено продуктивність вебзастосунку та виявлено наступні результати:

– вебзастосунок має досить високу продуктивність, оскільки він відповідає на запити протягом 1-2 секунд, використовує оптимальну кількість ресурсів, та

підтримує стабільну доступність;

– вебзастосунок має досить високу масштабованість, оскільки він може обслуговувати збільшення навантаження за допомогою балансувальника навантаження, який розподіляє запити між декількома серверами;

– вебзастосунок має досить високу надійність, оскільки він може відновлюватися від помилок або збоїв за допомогою балансувальника навантаження, який виключає непрацюючі або проблемні сервери з пулу.

Таким чином, успішно розгорнуто вебзастосунок на хостингу AWS за допомогою Docker та оцінено його продуктивність за допомогою вебінспектора браузера та вебсервісів AWS. Вебзастосунок має досить високу якість та відповідає вимогам та потребам проекту.

3.4 Тестування роботи вебзастосунку

Тестування є невід’ємною частиною процесу розробки програмного забезпечення, і вебзастосунки не є винятком. Ретельне та всебічне тестування дозволяє виявити та усунути потенційні дефекти, помилки та вразливості, забезпечити відповідність вимогам, зручність використання та продуктивність системи. Тестування також підвищує довіру користувачів до вебзастосунку та мінімізує ризики, пов’язані з його використанням.

Для вебзастосунку підбору вакансій було проведено комплексне тестування, яке охоплювало різні аспекти системи. Процес тестування був ретельно спланований та документований, з чітко визначеними цілями, критеріями та методами тестування.

На початку процесу було розроблено детальний тестовий план, який включав наступні види тестування:

1) *функціональне тестування*: метою цього виду тестування є перевірка відповідності функціональності вебзастосунку заявленим вимогам та специфікаціям. Воно охоплює всі основні функції системи, такі як реєстрація та авторизація користувачів, управління вакансіями та резюме, пошук та фільтрація,

комунікація між користувачами тощо. Функціональне тестування виконується шляхом розробки та виконання тестових сценаріїв, які симулюють реальні дії користувачів та перевіряють коректність отриманих результатів;

2) *тестування зручності використання (usability testing)*: це тестування спрямоване на оцінку того, наскільки легко та зручно користувачам взаємодіяти з вебзастосунком. Воно включає перевірку інтуїтивності навігації, зрозумілості інтерфейсу, доступності функцій, читабельності контенту, швидкості виконання завдань тощо. Для проведення usability тестування залучаються реальні користувачі з цільової аудиторії, які виконують типові сценарії використання системи та надають зворотний зв’язок;

3) *тестування безпеки*: вебзастосунки часто є мішенню для зловмисників, оскільки вони оперують конфіденційними даними користувачів та організацій. Тому ретельне тестування безпеки є критично важливим. Воно включає перевірку на типові вебвразливості (ін’єкції, міжсайтовий скриптинг, підробка міжсайтових запитів, незахищені прямі посилання на об’єкти тощо), перевірку автентифікації та авторизації, шифрування даних, безпечної конфігурації сервера тощо. Для автоматизації тестування безпеки використовуються спеціальні інструменти, такі як сканери вразливостей;

4) *тестування продуктивності*: продуктивність вебзастосунку безпосередньо впливає на задоволеність користувачів. Ніхто не любить довго чекати завантаження сторінок або отримувати помилки через високе навантаження на сервер. Тестування продуктивності дозволяє оцінити швидкість, стабільність та масштабованість вебзастосунку під різними навантаженнями. Воно включає навантажувальне тестування (оцінка поведінки системи при очікуваному навантаженні), стрес-тестування (визначення меж продуктивності системи), тестування стабільності (перевірка здатності системи працювати без збоїв протягом тривалого часу). Для навантажувального тестування використовуються спеціальні інструменти, які симулюють одночасну роботу великої кількості користувачів;

5) тестування сумісності: вебзастосунки повинні коректно працювати в різних оточеннях – на різних пристроях (ПК, планшетах, смартфонах), операційних системах, браузерах, а також при різних налаштуваннях (розмір екрану, роздільна здатність, ввімкнені плагіни тощо). Метою тестування сумісності є перевірка коректної роботи вебзастосунку в усіх підтримуваних конфігураціях. Воно охоплює тестування кросбраузерності (сумісності з різними браузерами та їх версіями), адаптивності (коректного відображення на екранах різних розмірів), сумісності з платформами та ОС.

Окрім вищезгаданих основних видів тестування, були також проведені додаткові перевірки, такі як тестування локалізації (коректність перекладу інтерфейсу різними мовами), тестування доступності (відповідність стандартам та керівництвам щодо доступності для людей з обмеженими можливостями), тестування відновлення після збоїв (здатність системи відновлювати коректний стан після апаратних або програмних збоїв).

Для кожного виду тестування були розроблені детальні тестові сценарії, що охоплюють як позитивні (коректні), так і негативні (неприпустимі) варіанти використання. Наприклад, для функціонального тестування були створені наступні сценарії для перевірки функції реєстрації користувача:

1) успішна реєстрація з коректними даними;
2) спроба реєстрації з уже зареєстрованою електронною адресою;
3) спроба реєстрації з невалідною електронною адресою;
4) спроба реєстрації з надто коротким паролем;
5) спроба реєстрації з незаповненими обов’язковими полями;
6) спроба реєстрації з некоректними символами в полі імені.

Подібні сценарії були розроблені для всіх основних функцій вебзастосунку – авторизації, створення та редагування вакансій і резюме, пошуку, комунікації, адміністрування тощо. Процес тестування відбувався ітераційно, паралельно з розробкою. Кожна нова функція або зміна тестувалася відразу після її реалізації. Результати тестування вебзастосунку наведено у табл. 3.1.

Таблиця 3.1 – Результати тестування вебзастосунку підбору вакансій

Вид тестування	**Кількість тестових сценаріїв**	**Пройдено успішно**	**Знайдено дефектів**	**Виправлено дефектів**
Функціональне тестування	150	145 (97%)	5	5
Usability тестування	20	16 (80%)	4	4
Тестування безпеки	50	48 (96%)	2	2
Тестування продуктивності	10	9 (90%)	1	1
Тестування сумісності	30	29 (97%)	1	1
Всього	**260**	**247 (95%)**	**13**	**13**

Як видно з таблиці, було проведено 260 тестових сценаріїв, які охопили всі основні аспекти вебзастосунку. 95% сценаріїв були успішно пройдені, що свідчить про високу якість розробленої системи. Було знайдено 13 дефектів різного рівня критичності, але всі вони були успішно виправлені та повторно протестовані.

Серед знайдених дефектів були такі проблеми, як:

– некоректна обробка деяких спеціальних символів у полях форм;

– відсутність підтвердження при видаленні важливих даних;

– можливість доступу до сторінок адміністрування для неавторизованих користувачів через пряме посилання;

– повільне завантаження сторінок з великою кількістю вакансій;

– некоректне відображення деяких елементів інтерфейсу в старих версіях браузеру Internet Explorer.

Всі ці проблеми були оперативно вирішені, що дозволило значно підвищити якість та надійність вебзастосунку.

Окрім описаного функціонального тестування, була також проведена автоматизація частини регресивних тестів за допомогою фреймворку Selenium. Це дозволило швидко перевіряти, що нові зміни в коді не спричинили регресій –

порушень у раніше працюючому функціоналі. Автоматизовані тести запускалися після кожного внесення змін, що заощадило значну кількість часу на ручне тестування.

Також було проведено тестування користувачами на етапі UAT (приймальне тестування). Група потенційних користувачів вебзастосунку (рекрутери, шукачі роботи, адміністратори) отримали доступ до системи та використовували її протягом деякого часу, надаючи зворотний зв’язок щодо зручності, функціональності та відповідності своїм потребам. Їх відгуки були ретельно проаналізовані та за їх результатами було зроблено ряд доопрацювань в інтерфейсі та функціоналі системи.

Наприкінці тестування було проведено повторний прогін всіх тестових сценаріїв на фінальній версії вебзастосунку для підтвердження його якості та готовності до експлуатації. Всі тести були успішно пройдені без виявлення нових дефектів.

Підводячи підсумки, можна сказати, що ретельне та всебічне тестування є запорукою успішної розробки та впровадження вебзастосунку. Воно дозволяє забезпечити високу якість, надійність, продуктивність та зручність використання системи, мінімізувати ризики та підвищити задоволеність кінцевих користувачів. Для вебзастосунку підбору вакансій було проведено комплексне тестування за різними напрямками, яке довело його відповідність всім висунутим вимогам та готовність до використання в реальних умовах.

В майбутньому планується регулярне проведення регресійного тестування при додаванні нових функцій та виправленні дефектів, щоб гарантувати стабільну роботу вебзастосунку протягом всього життєвого циклу. Також буде продовжуватися збір відгуків від користувачів та аналіз даних про використання застосунку для його постійного вдосконалення та адаптації до потреб ринку.

Висновки до розділу 3

У даному розділі було детально розглянуто процес реалізації фронтенду вебзастосунку підбору вакансій. Було описано основні сторінки застосунку, їх функціональність та взаємодію з сервером. Головна сторінка відображає список актуальних вакансій з можливістю пошуку та фільтрації. Кожна вакансія представлена у вигляді окремого блоку з основною інформацією та кнопкою для переходу на сторінку деталей вакансії. Сторінка деталей вакансії надає повну інформацію про вакансію, включаючи докладний опис, вимоги до кандидатів та завантажені медіафайли. З цієї сторінки рекрутер може ініціювати процес передачі клієнта на вакансію. Для управління передачами клієнтів було реалізовано сторінку зі списком передач, де адміністратор може переглядати, підтверджувати та видаляти передачі. Підтвердження передачі дозволяє вказати суму винагороди.

Адміністратор має можливість додавати нові вакансії через спеціальну форму, яка включає поля для введення деталей вакансії та завантаження медіафайлів. Для зручності редагування докладного опису вакансії було інтегровано текстовий редактор Quill.

Фронтенд застосунку було розроблено з використанням HTML, CSS та JavaScript. Для взаємодії з сервером використовувалися PHP та PDO для роботи з базою даних. Було реалізовано функції для отримання даних про вакансії, передачі клієнтів, користувачів та інших сутностей системи. Протягом розробки фронтенду було приділено увагу зручності використання, інтуїтивно зрозумілому інтерфейсу та забезпеченню безпеки даних. Використання підготовлених виразів (prepared statements) для запитів до бази даних допомогло запобігти потенційним вразливостям, таким як SQL-ін’єкції.

Реалізація фронтенду вебзастосунку підбору вакансій забезпечила зручний та ефективний інтерфейс для взаємодії користувачів з системою. Рекрутери отримали можливість легко знаходити та передавати клієнтів на вакансії, а адміністратори – керувати вакансіями та передачами клієнтів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було проаналізовано поточний стан ринку праці в Україні, виявлено основні проблеми та недоліки існуючих рішень у сфері підбору вакансій. На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до вебзастосунку, який би задовольняв потреби рекрутингових компаній та дозволяв оптимізувати процес підбору персоналу.

Для реалізації поставлених завдань було обрано сучасний стек вебтехнологій, що включає мови програмування PHP, HTML, CSS та JavaScript, систему керування базами даних MySQL, а також фреймворк Bootstrap для створення адаптивного інтерфейсу користувача. Обґрунтовано доцільність використання обраних інструментів з точки зору їх функціональності, надійності, безпеки та сумісності.

У ході проектування вебзастосунку розроблено його архітектуру, що базується на принципах модульності та розширюваності. Спроектовано структуру бази даних, яка забезпечує ефективне зберігання та обробку інформації про вакансії, клієнтів, передачі клієнтів на вакансії, користувачів та їх ролі в системі. Розроблено загальний алгоритм функціонування системи, що включає процеси реєстрації та авторизації користувачів, управління вакансіями, передачі клієнтів на вакансії, адміністрування системи тощо.

На етапі реалізації вебзастосунку створено його фронтенд та бекенд частини з використанням обраних технологій. Забезпечено функціонування всіх передбачених можливостей системи, таких як реєстрація та авторизація користувачів, розміщення та редагування вакансій, передача клієнтів на вакансії, управління передачами клієнтів, комунікація між користувачами системи тощо.

Проведено ретельне тестування розробленого вебзастосунку, що включало перевірку коректності роботи всіх функцій, безпеки та надійності системи, зручності користувацького інтерфейсу. За результатами тестування виконано

необхідні доопрацювання та оптимізації, що дозволило підвищити якість та ефективність роботи вебзастосунку.

Таким чином, у ході виконання кваліфікаційної роботи бакалавра досягнуто поставленої мети щодо проектування та розробки веборієнтованої інформаційної системи для автоматизації процесу підбору вакансій в рекрутинговій компанії. Створений вебзастосунок відповідає сучасним вимогам до програмного забезпечення такого типу та може бути рекомендований для впровадження в діяльність компаній, що надають послуги з підбору персоналу.

Подальші напрямки розвитку та вдосконалення розробленого вебзастосунку можуть включати розширення функціональних можливостей системи, зокрема додавання модулів аналітики та звітності, впровадження системи сповіщень та нагадувань, інтеграцію з іншими сервісами та платформами, що використовуються в рекрутинговій діяльності. Крім того, перспективним є розширення можливостей системи для адаптації до потреб міжнародного ринку праці та локалізації для використання в інших країнах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ринок праці в Україні 2022-2023. Стан, тенденції та перспективи. *Федерація роботодавців України*. URL: https://fru.ua/ua/media-center/analytics/rinok-pratsi-v-ukrajini-2022-2023-stan-tendentsiji-ta-perspektivi (дата звернення: 15.04.2024).

2. Ринок праці у 2021 році. Підсумки у цифрах і фактах. *European Business Association*. URL: https://eba.com.ua/rynok-pratsi-u-2021-rotsi-pidsumky-u-tsyfrah-i-faktah/ (дата звернення: 15.04.2024).

3. Гайдай І. О., Андрощук Г. В. Аналіз ринку праці України. *Збірник наукових праць Подільського державного аграрно-технічного університету*. URL: https://conf.ztu.edu.ua/wp-content/uploads/2018/12/58.pdf (дата звернення: 16.04.2024).

4. Котенко С. І., Котенко В. М. Глобальні та національні проблеми економіки. Ринок праці в Україні: аналіз сучасного стану та перспективи розвитку. *Архів миколаївського національного університету імені В. О. Сухомлинського*. Миколаїв, 2018. №22. – С. 1074-1079. URL: http://global-national.in.ua/archive/22-2018/199.pdf (дата звернення: 18.04.2024).

5. Рекрутингова система. *Вебсайт пошуку роботи Work.ua.* URL: https://www.work.ua/recruiting-system/ (дата звернення: 22.04.2024).

6. Потапова С. С. Вебсервіс вибору вакансій. URL: https://cad.kpi.ua/wp-content/uploads/2023/06/%D0%9F%D0%BE%D1%82%D0%B0%D0%BF%D0%BE%D0%B2%D0%B0_2021.pdf (дата звернення: 25.04.2024).

7. Ринок праці у сфері IT у 2022 році: зарплати, спеціалізації та вимоги у вакансіях. Скільки заробляли програмісти в 2022 році. *Платформа пошуку роботи No Fluff Jobs*. URL: https://nofluffjobs.com/ua/insights/report-the-it-job-market-in-2022-earnings-in-it-how-much-do-programmers-earn-ukraine/ (дата звернення: 27.04.2024).

8. Шишкіна О. О., Жук О. В. Особливості вебдосвіду користувачів при

працевлаштуванні через мережу Інтернет. *Наукові записки НаУКМА. Комп'ютерні науки*. Вип. 1. 2021. Т. 3. С. 85-91. URL: https://ir.library.knu.ua/server/api/core/bitstreams/035391cc-c654-4e56-8e63-2911e9b0e47c/content (дата звернення: 01.05.2024).

9. Судаков М., Лісогор Л. Ринок праці України 2022-2023: стан, тенденції та перспективи. *Європейський банк реконструкції та розвитку*. URL: https://fru.ua/images/doc/2023/EBRD_Report_20_04_2023.pdf (дата звернення: 01.05.2024).

10. Як розгорнути Node.js React-додаток на AWS: Стратегії та кращі практики. *Фахова соцмережа для креативних індустрій CASES*. URL: https://cases.media/en/article/yak-rozgornuti-node-js-react-dodatok-na-aws-strategiyi-ta-krashi-praktiki (дата звернення: 03.05.2024).

11. Deploy modern web application on AWS: Strategy & Best Practices. *Medium*. URL: https://medium.com/bb-tutorials-and-thoughts/deploy-modern-web-application-on-aws-strategy-best-practices-9f5bccf085e5 (дата звернення: 05.05.2024).

12. Serverless Python: розгортання, моніторинг та оптимізація AWS Lambda. *DOU*. URL: https://dou.ua/lenta/articles/serverless-python/ (дата звернення: 05.05.2024).

13. Порівняльний аналіз платформ для розробки вебзастосунків. *Освітній портал Державного університету «Житомирська політехніка»*. URL: https://learn.ztu.edu.ua/mod/url/view.php?id=123881 (дата звернення: 06.05.2024).

14. Mahesh Chand. What is MERN Stack: Introduction, Architecture, Advantages & Use Cases. *C# Corner*. 2022. URL: https://www.c-sharpcorner.com/article/what-is-mern-stack-introduction-architecture-advantages-and-use-cases/ (дата звернення: 06.05.2024).

15. Pavan Nada. What Is MEAN Stack? *Forbes*. 2022. URL: https://www.forbes.com/advisor/business/software/mean-stack/ (дата звернення: 08.05.2024).

16. Чалий Р. Архітектура програмних систем на базі мікросервісів.

Інституційний репозиторій національного університету «Львівська політехніка». URL: http://ena.lp.edu.ua:8080/bitstream/ntb/54847/2/2020_Chalyi_R-Arkhitektura-prohramnykh-system_54-55.pdf (дата звернення: 08.05.2024).

17. Martin Fowler. Microservices. *MartinFowler.com*. 2014. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 10.05.2024).

18. Testing Pyramid: How to structure your test suite. *Halo Lab Blog*. 2021. URL: https://halolab.io/blog/testing-pyramid-how-to-structure-your-test-suite/ (дата звернення: 11.05.2024).

19. Mehul Kaushik. Introduction to Test Driven Development (TDD). *GeeksforGeeks*. 2022. URL: https://www.geeksforgeeks.org/introduction-to-test-driven-development-tdd/ (дата звернення: 12.05.2024).

20. Jakob Nielsen. 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group*. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 14.05.2024).

21. Web Content Accessibility Guidelines (WCAG) 2.1. *W3C Recommendation*. 2018. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 14.05.2024).

22. OWASP Top Ten Web Application Security Risks. OWASP. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 14.05.2024).

23. Yuriy Shema. Ultimate guide to web application security threats and risks. *Veracode*. URL: https://crashtest-security.com/web-application-security-threats-and-risks/ (дата звернення: 15.05.2024).

24. Бондаренко М. Ф., Маторін С. І., Соловйова О. А. Моделі та методи системного аналізу в задачах автоматизації підбору вакансій. Проблеми програмування. 2020. № 2-3. С. 228-237. URL: http://pp.isofts.kiev.ua/ojs1/article/view/422 (дата звернення: 02.06.2024).

25. Яровий А. А., Куперштейн Л. М. Інтелектуальний аналіз великих даних для підтримки прийняття рішень в процесі підбору вакансій. Оптико-електронні інформаційно-енергетичні технології. 2019. № 2 (38). С. 84-93. URL: https://oeipt.vntu.edu.ua/index.php/oeipt/article/view/659 (дата звернення: 02.06.2024).

ДОДАТОК А

Програмний код вебзастосунку

index.php

```
<?php
session_start();
require_once ‘php/functions.php’;

if (!isLoggedIn()) {
   header(‘Location: login.php’);
   exit;
}

// Оновлення часу останнього входу користувача
updateLastLogin($_SESSION[‘user_id’]);

$vacancies = getVacancies();

include ‘templates/header.php’;
?>

<div class="container">
      <h1>Актуальні Вакансії</h1>
      <table>
         <thead>
            <tr>
               <th>Назва</th>
               <th>Місто роботи</th>
               <th>Кількість місць</th>
               <th>Стать</th>
            </tr>
         </thead>
         <tbody>
            <?php foreach ($vacancies as $vacancy): ?>
            <tr>
               <td><a href="vacancy_details.php?id=<?= htmlspecialchars($vacancy[‘id’]) ?>"><?=
htmlspecialchars($vacancy[‘name’]) ?></a></td>
               <td><?= htmlspecialchars($vacancy[‘work_city’]) ?></td>
               <td><?= htmlspecialchars($vacancy[‘number_of_positions’]) ?></td>
               <td><?= htmlspecialchars($vacancy[‘gender’]) ?></td>
            </tr>
            <?php endforeach; ?>
         </tbody>
      </table>
   </div>

<?php include ‘templates/footer.php’; ?>
```

vacancy_details.php

```
<?php
session_start();
require_once ‘php/functions.php’;

if (!isLoggedIn()) {
   header(‘Location: login.php’);
   exit;
}
```

```
if (!isset($_GET[‘id’])) {
   header(‘Location: vacancies.php’);
   exit;
}

$vacancyId = $_GET[‘id’];
$vacancyDetails = getVacancyDetails($vacancyId);

include ‘templates/header.php’;
?>

<div class="container">
   <h1>Деталі Вакансії</h1>

   <?php if ($vacancyDetails): ?>
      <table class="table table-bordered">
         <tr>
            <th>Назва</th>
            <td><?= htmlspecialchars($vacancyDetails[‘vacancy’][‘name’]) ?></td>
         </tr>
         <tr>
            <th>Місто виконання роботи</th>
            <td><?= htmlspecialchars($vacancyDetails[‘vacancy’][‘work_city’]) ?></td>
         </tr>
         <tr>
            <th>Кількість місць</th>
            <td><?= htmlspecialchars($vacancyDetails[‘vacancy’][‘number_of_positions’]) ?></td>
         </tr>
         <tr>
            <th>Стать</th>
            <td><?= htmlspecialchars($vacancyDetails[‘vacancy’][‘gender’]) ?></td>
         </tr>
         <tr>
            <th>Докладний опис для клієнта</th>
            <td><?= $vacancyDetails[‘vacancy’][‘detailed_description’] ?></td>
         </tr>
      </table>
      <div class="vacancy-actions">
         <a href="edit_vacancy.php?id=<?= $vacancyId ?>" class="btn btn-primary">Редагувати</a>
         <a href="delete_vacancy.php?id=<?= $vacancyId ?>" class="btn btn-danger" onclick="return confirm(‘Ви
впевнені, що хочете видалити цю вакансію?’)">Видалити</a>
      </div>
      <h2>Медіафайли</h2>
      <div class="media-files-row" style="display: flex; overflow-x: auto; gap: 10px; padding-bottom: 10px;">
         <?php foreach ($vacancyDetails[‘mediaFiles’] as $file): ?>
            <div class="media-file" style="flex-shrink: 0;">
               <a href="<?= htmlspecialchars($file[‘file_path’]) ?>" download>
                  <img src="<?= htmlspecialchars($file[‘file_path’]) ?>" alt="Media File" style="max-width: 100px; height:
auto;">
               </a>
            </div>
         <?php endforeach; ?>
      </div>
   <?php endif; ?>

   <a href="vacancy.php" class="btn btn-secondary">Повернутись до списку вакансій</a>
</div>

<?php include ‘templates/footer.php’; ?>
```

transfer_client.php

```
<?php
session_start();
require_once 'php/functions.php';

if (!isLoggedIn()) {
    header('Location: login.php');
    exit;
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Process text inputs from the form
    $name = $_POST['name'];
    $urgency = $_POST['urgency'];
    $registration_city = $_POST['registration_city'];
    $work_city = $_POST['work_city'];
    $number_of_positions = $_POST['number_of_positions'];
    $citizenship = $_POST['citizenship'];
    $standard_rate = $_POST['standard_rate'];
    $student_rate = $_POST['student_rate'];
    $gender = $_POST['gender'];
    $age = $_POST['age'];
    $schedule = $_POST['schedule'];
    $night_shifts = $_POST['night_shifts'];
    $accommodation = $_POST['accommodation'];
    $work_experience = $_POST['work_experience'];
    $temperature_regime = $_POST['temperature_regime'];
    $health_requirements = $_POST['health_requirements'];
    $recruitment_information = $_POST['recruitment_information'];
    $detailed_description = $_POST['detailed_description'];

    // Process file uploads
    $randomString = bin2hex(random_bytes(3)); // generates a string of 6 characters
    $folderName = 'uploads/vacancies/' . $randomString;
    if (!file_exists($folderName)) {
        mkdir($folderName, 0777, true);
    }

    $filePaths = [];
    foreach ($_FILES['files']['name'] as $i => $fileName) {
        $tmpName = $_FILES['files']['tmp_name'][$i];
        if ($fileName) {
            $destination = $folderName . '/' . $fileName;
            if (move_uploaded_file($tmpName, $destination)) {
                $filePaths[] = $destination;
            }
        }
    }

    // Create a new vacancy and link media files
    $vacancyId = createVacancy($name, $urgency, $registration_city, $work_city, $number_of_positions, $citizenship,
$standard_rate, $student_rate, $gender, $age, $schedule, $night_shifts, $accommodation, $work_experience,
$temperature_regime, $health_requirements, $recruitment_information, $detailed_description);
    foreach ($filePaths as $filePath) {
        linkMediaToVacancy($vacancyId, $filePath);
    }

    header('Location: index.php');
    exit;
}
```

```
include ‘templates/header.php’;
?>

<div class="container">
  <h1>Додати нову вакансію</h1>
  <form action="add_vacancies.php" method="post" enctype="multipart/form-data">
    <div class="mb-3">
      <label for="name" class="form-label">Назва</label>
      <input type="text" name="name" id="name" class="form-control" required>
    </div>

    <div class="mb-3">
      <label for="work_city" class="form-label">Місто виконання робіт</label>
      <input type="text" name="work_city" id="work_city" class="form-control">
    </div>

    <div class="mb-3">
      <label for="number_of_positions" class="form-label">Кількість місць</label>
      <input type="text" name="number_of_positions" id="number_of_positions" class="form-control">
    </div>

    <div class="mb-3">
      <label for="gender" class="form-label">Стать</label>
      <input type="text" name="gender" id="gender" class="form-control">
    </div>

<div class="mb-3">
      <label for="detailed_description" class="form-label">Докладний опис для клієнта</label>
      <div id="editor-container"></div>
      <input type="hidden" name="detailed_description" id="detailed_description">
    </div>

    <div class="mb-3">
      <label for="file_upload" class="form-label">Завантажити медіафайли</label>
      <input type="file" name="files[]" id="file_upload" class="form-control" multiple>
    </div>

    <button type="submit" class="btn btn-primary">Додати вакансію</button>
  </form>
</div>

<script>
  document.getElementById(‘file_upload’).addEventListener(‘change’, function (event) {
    const files = event.target.files;
    const fileList = document.createElement(‘div’);
    fileList.id = ‘file_list’;
    document.body.appendChild(fileList);

    for (let i = 0; i < files.length; i++) {
      const file = files[i];
      const listItem = document.createElement(‘div’);
      listItem.className = ‘d-flex justify-content-between align-items-center’;
      listItem.innerHTML = `
        <span>${file.name}</span>
        <button type="button" class="btn btn-danger btn-sm" onclick="removeFile(this, ${i})">Delete</button>
      `;
      fileList.appendChild(listItem);
    }
  });
```

```
    function removeFile(button, index) {
        const fileList = document.getElementById('file_list');
        fileList.removeChild(button.parentElement);

        // Update file list in input
        let dt = new DataTransfer();
        let input = document.getElementById('file_upload');
        let { files } = input;

        for (let i = 0; i < files.length; i++) {
            if (index !== i) {
                dt.items.add(files[i]);
            }
        }

        input.files = dt.files;
    }
</script>

<!-- Підключення Quill редактора -->
<link href="https://cdn.quilljs.com/1.3.6/quill.snow.css" rel="stylesheet">
<script src="https://cdn.quilljs.com/1.3.6/quill.js"></script>

<script>
    var quill = new Quill('#editor-container', {
        theme: 'snow'
    });
    var form = document.querySelector('form');
    form.onsubmit = function() {
        // Отримання HTML вмісту з редактора Quill
        document.querySelector('input[name=detailed_description]').value = quill.root.innerHTML;
    };
</script>



<?php include 'templates/footer.php'; ?>
```

confirm_transfer.php

```
<?php
session_start();
require_once 'php/functions.php';

if (!isLoggedIn()) {
    header('Location: login.php');
    exit;
}

$transferId = $_GET['id'] ?? null;
$transferDetails = getTransferDetailss($transferId);
$userName = getUserNameById($transferDetails['user_id']);

if (!$transferDetails) {
    echo "Передача не знайдена";
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $userId = $_SESSION['user_id'];
    $reward = $_POST['reward']; // Винагорода, вказана користувачем
```

```
    confirmClientTransfer($transferId, $userId, $reward);
    header('Location: clients.php');
    exit;
}

include 'templates/header.php';
?>

<div class="container">
    <h1>Підтвердження передачі клієнта</h1>
    <p><strong>Назва вакансії:</strong> <?= htmlspecialchars($transferDetails['vacancy_name']) ?></p>
    <p><strong>Ім'я клієнта:</strong> <?= htmlspecialchars($transferDetails['client_name']) ?></p>
    <p><strong>Ім'я рекрутера:</strong> <?= htmlspecialchars($userName) ?></p>

    <!-- Додаткові деталі передачі -->

    <form method="post">
        <input type="hidden" name="transferId" value="<?= htmlspecialchars($transferId) ?>">

        <div class="mb-3">
            <label for="reward" class="form-label">Винагорода</label>
            <input type="number" name="reward" id="reward" class="form-control" placeholder="Вкажіть винагороду"
required>
        </div>

        <button type="submit" class="btn btn-primary">Підтвердити передачу</button>
    </form>
</div>

<?php include 'templates/footer.php'; ?>
```

Деякі функції файлу functions.php

```
// Робота з вакансіями

function createVacancy($name, $urgency, $registration_city, $work_city, $number_of_positions, $citizenship,
$standard_rate, $student_rate, $gender, $age, $schedule, $night_shifts, $accommodation, $work_experience,
$temperature_regime, $health_requirements, $recruitment_information, $detailed_description) {
    global $db;

    // SQL query to insert data into the vacancies table
    $sql = "INSERT INTO vacancies (name, urgency, registration_city, work_city, number_of_positions, citizenship,
standard_rate, student_rate, gender, age, schedule, night_shifts, accommodation, work_experience, temperature_regime,
health_requirements, recruitment_information, detailed_description)
            VALUES (:name, :urgency, :registration_city, :work_city, :number_of_positions, :citizenship, :standard_rate,
:student_rate, :gender, :age, :schedule, :night_shifts, :accommodation, :work_experience, :temperature_regime,
:health_requirements, :recruitment_information, :detailed_description)";

    // Prepare the statement
    $stmt = $db->prepare($sql);

    // Bind parameters to the SQL query
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':urgency', $urgency);
    $stmt->bindParam(':registration_city', $registration_city);
    $stmt->bindParam(':work_city', $work_city);
    $stmt->bindParam(':number_of_positions', $number_of_positions);
    $stmt->bindParam(':citizenship', $citizenship);
    $stmt->bindParam(':standard_rate', $standard_rate);
    $stmt->bindParam(':student_rate', $student_rate);
```

```
    $stmt->bindParam(':gender', $gender);
    $stmt->bindParam(':age', $age);
    $stmt->bindParam(':schedule', $schedule);
    $stmt->bindParam(':night_shifts', $night_shifts);
    $stmt->bindParam(':accommodation', $accommodation);
    $stmt->bindParam(':work_experience', $work_experience);
    $stmt->bindParam(':temperature_regime', $temperature_regime);
    $stmt->bindParam(':health_requirements', $health_requirements);
    $stmt->bindParam(':recruitment_information', $recruitment_information);
    $stmt->bindParam(':detailed_description', $detailed_description);

    // Execute the statement
    $stmt->execute();

    // Return the ID of the newly created vacancy
    return $db->lastInsertId();
}

function deleteVacancy($vacancyId) {
    global $db;

    // Begin transaction
    $db->beginTransaction();

    try {
        // Update the column names as per your actual table structure
        $deletions = [
            'client_transfer_confirmations' => 'transfer_id', // Update the column name
            'client_passport_photos' => 'transfer_id', // Update the column name
            'client_transfers' => 'vacancy_id',
            'vacancy_media' => 'vacancy_id'
        ];

        foreach ($deletions as $table => $column) {
            $stmt = $db->prepare("DELETE FROM $table WHERE $column = :id");
            $stmt->bindParam(':id', $vacancyId, PDO::PARAM_INT);
            $stmt->execute();
        }

        // Delete the vacancy
        $stmt = $db->prepare("DELETE FROM vacancies WHERE id = :id");
        $stmt->bindParam(':id', $vacancyId, PDO::PARAM_INT);
        $stmt->execute();

        // Commit transaction
        $db->commit();
    } catch (PDOException $e) {
        $db->rollBack();
        throw $e;
    }
}



function linkMediaToVacancy($vacancyId, $filePath) {
    global $db;

    $sql = "INSERT INTO vacancy_media (vacancy_id, file_path) VALUES (:vacancy_id, :file_path)";

    $stmt = $db->prepare($sql);
    $stmt->bindParam(':vacancy_id', $vacancyId);
```

```
    $stmt->bindParam(':file_path', $filePath);

    $stmt->execute();
}

function getVacancies() {
    global $db; // Переконайтеся, що у вас є змінна $db, яка представляє з'єднання з базою даних

    try {
        // Створення SQL запиту для отримання всіх вакансій
        $sql = "SELECT * FROM vacancies";

        // Підготовка та виконання запиту
        $stmt = $db->prepare($sql);
        $stmt->execute();

        // Отримання результатів
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        // Обробка помилки
        error_log("Помилка при отриманні вакансій: " . $e->getMessage());
        return []; // Повернення порожнього масиву у разі помилки
    }
}

function getVacancyDetails($id) {
    global $db; // Переконайтеся, що змінна $db ініціалізована і представляє з'єднання з базою даних

    try {
        // Отримання деталей вакансії
        $stmt = $db->prepare("SELECT * FROM vacancies WHERE id = :id");
        $stmt->execute([':id' => $id]);
        $vacancy = $stmt->fetch(PDO::FETCH_ASSOC);

        // Отримання медіафайлів вакансії
        $mediaStmt = $db->prepare("SELECT * FROM vacancy_media WHERE vacancy_id = :id");
        $mediaStmt->execute([':id' => $id]);
        $mediaFiles = $mediaStmt->fetchAll(PDO::FETCH_ASSOC);

        return [':vacancy' => $vacancy, ':mediaFiles' => $mediaFiles];
    } catch (PDOException $e) {
        error_log("Error fetching vacancy details: " . $e->getMessage());
        return null;
    }
}

function updateVacancy($data, $vacancyId, $files) {
    global $db;

    // Оновлення даних вакансії
    $sql = "UPDATE vacancies SET
                name = :name,
                urgency = :urgency,
                registration_city = :registration_city,
                work_city = :work_city,
                number_of_positions = :number_of_positions,
                citizenship = :citizenship,
                standard_rate = :standard_rate,
                student_rate = :student_rate,
                gender = :gender,
                age = :age,
```

```
            schedule = :schedule,
            night_shifts = :night_shifts,
            accommodation = :accommodation,
            work_experience = :work_experience,
            temperature_regime = :temperature_regime,
            health_requirements = :health_requirements,
            recruitment_information = :recruitment_information,
            detailed_description = :detailed_description
        WHERE id = :id";

    $stmt = $db->prepare($sql);
    $stmt->execute([
        ':name' => $data['name'] ?? null,
        ':urgency' => $data['urgency'] ?? null,
        ':registration_city' => $data['registration_city'] ?? null,
        ':work_city' => $data['work_city'] ?? null,
        ':number_of_positions' => $data['number_of_positions'] ?? null,
        ':citizenship' => $data['citizenship'] ?? null,
        ':standard_rate' => $data['standard_rate'] ?? null,
        ':student_rate' => $data['student_rate'] ?? null,
        ':gender' => $data['gender'] ?? null,
        ':age' => $data['age'] ?? null,
        ':schedule' => $data['schedule'] ?? null,
        ':night_shifts' => $data['night_shifts'] ?? null,
        ':accommodation' => $data['accommodation'] ?? null,
        ':work_experience' => $data['work_experience'] ?? null,
        ':temperature_regime' => $data['temperature_regime'] ?? null,
        ':health_requirements' => $data['health_requirements'] ?? null,
        ':recruitment_information' => $data['recruitment_information'] ?? null,
        ':detailed_description' => $data['detailed_description'] ?? null,
        ':id' => $vacancyId
    ]);

    // Процес оновлення медіафайлів
    if (!empty($files['files']['name'][0])) {
        // Припускаємо, що файлів може бути декілька, тому використовуємо цикл
        for ($i = 0; $i < count($files['files']['name']); $i++) {
            // Перевіряємо, чи файл був дійсно завантажений
            if (is_uploaded_file($files['files']['tmp_name'][$i])) {
                // Отримуємо тимчасове ім'я файлу
                $tmpFilePath = $files['files']['tmp_name'][$i];
                // Визначаємо кінцеве місце зберігання та ім'я файлу
                // Тут ви можете використовувати свій шлях для зберігання файлів
                $destinationPath = '/path/to/your/storage/' . $files['files']['name'][$i];

                // Переміщуємо файл з тимчасового розташування до кінцевого
                if (move_uploaded_file($tmpFilePath, $destinationPath)) {
                    // Якщо файл успішно переміщений, зв'яжіть його з вакансією
                    linkMediaToVacancy($vacancyId, $destinationPath);
                }
            }
        }
    }
}

function getVacancyNameById($vacancyId) {
    global $db; // Переконайтеся, що у вас вже є змінна $db, яка представляє з'єднання з базою даних

    try {
        $sql = "SELECT name FROM vacancies WHERE id = :vacancyId";
        $stmt = $db->prepare($sql);
```

```
        $stmt->bindParam(':vacancyId', $vacancyId, PDO::PARAM_INT);
        $stmt->execute();

        $result = $stmt->fetch(PDO::FETCH_ASSOC);
        return $result ? $result['name'] : null;
    } catch (PDOException $e) {
        // Обробка помилки
        error_log("Error fetching vacancy name: " . $e->getMessage());
        return null;
    }
}

// Передача клієнту.

function createClientTransfer($clientName, $clientDob, $passportNumber, $passportExpiry, $documents,
$visaApplicationCity, $clientPhone, $arrivalDate, $hasAccommodation, $currentCountry, $vacancyId) {
    global $db; // Ensure that $db is your database connection variable

    try {
        $sql = "INSERT INTO client_transfers (user_id, client_name, client_dob, passport_number, passport_expiry,
documents, visa_application_city, client_phone, arrival_date, has_accommodation, current_country, vacancy_id)
                VALUES (:userId, :clientName, :clientDob, :passportNumber, :passportExpiry, :documents,
:visaApplicationCity, :clientPhone, :arrivalDate, :hasAccommodation, :currentCountry, :vacancyId)";

        $stmt = $db->prepare($sql);

        // Assuming you have the user's ID stored in session
        $userId = $_SESSION['user_id'];
        $documentsString = implode(',', $documents); // Convert array to string

        $stmt->bindParam(':userId', $userId);
        $stmt->bindParam(':clientName', $clientName);
        $stmt->bindParam(':clientDob', $clientDob);
        $stmt->bindParam(':passportNumber', $passportNumber);
        $stmt->bindParam(':passportExpiry', $passportExpiry);
        $stmt->bindParam(':documents', $documentsString);
        $stmt->bindParam(':visaApplicationCity', $visaApplicationCity);
        $stmt->bindParam(':clientPhone', $clientPhone);
        $stmt->bindParam(':arrivalDate', $arrivalDate);
        $stmt->bindParam(':hasAccommodation', $hasAccommodation, PDO::PARAM_BOOL);
        $stmt->bindParam(':currentCountry', $currentCountry);
        $stmt->bindParam(':vacancyId', $vacancyId);

        $stmt->execute();

        return $db->lastInsertId();
    } catch (PDOException $e) {
        // Handle exception
        error_log("Error in createClientTransfer: " . $e->getMessage());
        return null;
    }
}

function deleteTransfer($transferId) {
    global $db;

    // Begin transaction
    $db->beginTransaction();

    try {
        // Delete the transfer
```

```
        $stmt = $db->prepare("DELETE FROM client_transfers WHERE id = :id");
        $stmt->bindParam(':id', $transferId, PDO::PARAM_INT);
        $stmt->execute();

        // Commit transaction
        $db->commit();
    } catch (PDOException $e) {
        $db->rollBack();
        throw $e;
    }
}
```