МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри_____ Є. О. Давиденко
підпис
«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
Вебзастосунок пошуку роботи для ІТ-фахівців

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22011019

Здобувачка _____Ж. М. Руденко
підпис
«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____ Г. В. Горбань
підпис
«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
підпис
«__» _____ 2024 р.

Миколаїв 2024

ЗАТВЕРДЖУЮ

Зав. кафедри Є. О. Давиденко

«22» _грудня_2023р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студентці групи 408 факультету комп'ютерних наук

Руденко Жанні Михайлівні

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок пошуку роботи для ІТ-фахівців

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представленная кваліфікаційної роботи «___» червня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є вебзастосунок пошуку роботи для ІТ-фахівців

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до вебзастосунку;
- моделювання і проєктування користувацького інтерфейсу для вебзастосунку;
- проєктування бази даних;
- імплементація алгоритмів пошуку та фільтрації вакансій;
- тестування та відлагодження розробленого вебзастосунку.

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи ____канд. техн. наук, доцент Горбань Г. В._____
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_____Руденко Жанна Михайлівна_____

(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «22» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: _____Вебзастосунок пошуку роботи для ІТ-фахівців_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	15.12.2023	22.12.2023	Виконано
2.	Огляд літератури згідно теми КРБ	20.01.2024	30.01.2024	Виконано
3.	Складання календарного плану КРБ	01.02.2024	02.02.2024	Виконано
4.	Аналіз предметної області	05.02.2024	08.02.2024	Виконано
5.	Розробка проєктних рішень до КРБ	12.02.2024	15.02.2024	Виконано
6.	Проєктування інтерфейсу користувача	16.02.2024	17.03.2024	Виконано
7.	Проєктування бази даних	18.03.2024	30.03.2024	Виконано
8.	Імплементація алгоритмів пошуку та фільтрації вакансій	31.03.2024	15.04.2024	Виконано
9.	Тестування вебзастосунку за допомогою юніт-тестів.	16.04.2024	25.04.2024	Виконано
11.	Оформлення КРБ та презентації	26.04.2024	26.05.2024	Виконано
12.	Відгук керівника КРБ	27.05.2024	03.05.2024	Виконано
13.	Предзахист КРБ	04.06.2024	04.06.2024	
14	Розробка окремого модулю з охорони праці	05.06.2024	10.06.2024	Виконано
15.	Виправлення зауважень в результаті попереднього захисту КРБ	11.06.2024	12.06.2024	Виконано
16.	Подання рецезенту КРБ	13.06.2024	13.06.2024	Виконано
17.	Відгук рецезента КРБ	14.06.2024	17.06.2024	Виконано
18.	Захист КРБ	24.06.2024	24.06.2024	Виконано

Розробила здобувачка _____Руденко Жанна Михайлівна____ _____
(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи _____Горбань Гліб Валентинович_____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок пошуку роботи для ІТ-фахівців»

Здобувачка 408 гр.: Руденко Жанна Михайлівна

Керівник: канд. техн. наук, доцент Горбань Г. В.

Кваліфікаційна робота присвячена розробці вебзастосунку пошуку роботи для ІТ-фахівців, що буде забезпечувати швидкий та зручний доступ до актуальних вакансій у сфері ІТ, а також зробить процес пошуку роботи максимально ефективним та доступним для кожного користувача.

Об’єктом роботи є процес розробки вебзастосунку пошуку роботи для ІТ-фахівців.

Предметом кваліфікаційної роботи є інструментарій розробки вебзастосунку пошуку роботи для ІТ-фахівців.

Метою кваліфікаційної роботи є розробка вебзастосунку для ефективного пошуку роботи в сфері ІТ, спрямованого на полегшення процесу пошуку вакансій та взаємодії між роботодавцями та ІТ-фахівцями.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідження предметної галузі згідно отриманої теми КРБ;

2) проєктування інтерфейсу користувача з урахуванням зручності використання;

3) проєктування бази даних для зберігання інформації про вакансії та користувачів;

4) імплементація алгоритмів пошуку та фільтрації вакансій;

5) тестування вебзастосунку за допомогою юніт-тестів.

Структура кваліфікаційної роботи бакалавра включає вступ, чотири розділи, висновки та перелік джерел посилань.

У вступі визначається актуальність теми, мета, предмет та об’єкт дослідження.

У першому розділі проведено аналіз існуючих вебзастосунків-аналогів, визначення функціоналу, переваг, недоліків, технології за допомогою яких було

створено вебзастосунок. Формування та опис специфікації вимог вебзастосунку, що розробляється.

У другому розділі приведено моделювання та розробку структури застосунку, що розробляється.

У третьому розділі описується огляд мов, технологій та бібліотек, що використовуються для розробки, та процес проєктування програмного забезпечення.

У четвертому розділі описано процес розробки програмного забезпечення та його тестування.

У висновках проводиться аналіз виконаних робіт та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 78 сторінок, містить 4 розділи, 34 ілюстрації, 4 таблиці, 16 джерел в переліку посилань.

Ключові слова: *вебзастосунок, пошук роботи, програмування на Typescript, Angular фреймворк, рекрутер, вакансії.*

ABSTRACT

of the Bachelor's Thesis

"Web application for job search for IT professionals"

Student of group 408: Rudenko Zhanna Mykhailivna

Supervisor: Candidate of Technical Sciences, Associate Professor Gorban H. V.

This work is dedicated to the development of a web application for job search for IT professionals, which will provide fast and convenient access to current vacancies in the IT field, as well as make the job search process as efficient and accessible as possible for every user.

The object of the work is the process of developing a web application for job search for IT professionals.

The subject of the qualification work is the development toolkit of a web application for job search for IT professionals.

The goal of the qualification work is to develop a web application for effective job search in the IT field, aimed at facilitating the process of searching for vacancies and interaction between employers and IT professionals.

To achieve the stated goal, the following tasks need to be performed:

1) research of the subject area according to the received topic of the qualification work;
2) designing user interface considering usability;
3) designing a database for storing information about vacancies and users;
4) implementation of algorithms for searching and filtering vacancies;
5) testing the web application using unit tests.

The structure of the bachelor's qualification work includes an introduction, four chapters, conclusions, and a list of references.

The introduction defines the relevance of the topic, the goal, the subject, and the object of the research.

In the first chapter, an analysis of existing web applications-analogues is conducted, defining functionality, advantages, disadvantages, and technologies used to create the web application. Formation and description of the specification of requirements for the developed web application.

The second chapter presents modeling and development of the structure of the developed application.

The third chapter describes an overview of languages, technologies, and libraries used for development, and the software design process.

The fourth chapter describes the process of software development and its testing.

The conclusions analyze the work performed and the results obtained.

The bachelor's qualification work consists of 78 pages, including 4 chapters, 34 illustrations, 4 tables, and 16 sources in the reference list.

Keywords: *web application, job search, Typescript programming, Angular framework, recruiter, vacancies.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

БД – база даних

КРБ – кваліфікаційна робота бакалавра

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

ООМ – об’єктно-орієнтоване моделювання

JSON – JavaScript Object Notation

URL – Uniform Resource Locator

DFD – Data Flow Diagram

UI – User Interface

ВСТУП

В сучасному цифровому світі, де технології швидко розвиваються, індустрія інформаційних технологій займає центральне місце. Індустрія ІТ відіграє ключову роль у розвитку інших галузей економіки, що призводить до зростання попиту на кваліфікованих ІТ-фахівців. У зв'язку зі зростанням диджиталізації у всіх сферах бізнесу та життя, попит на кваліфікованих ІТ-фахівців стрімко зростає. Компанії активно шукають спеціалістів з різних галузей ІТ, починаючи від розробників програмного забезпечення до фахівців з аналізу даних та інженерії мереж. За останні роки способи пошуку роботи зазнали значних змін. Замість традиційних методів, таких як пошук вакансій у газетах або на сайтах роботодавців, більшість кандидатів зараз використовують вебзастосунки для пошуку роботи. Це обумовлено зручністю, швидкістю та широким доступом до інформації. Пандемія COVID-19 також суттєво вплинула на парадигму пошуку роботи. Багато компаній перейшли на дистанційну форму роботи, що розширило географію можливих місць праці для ІТ-фахівців. У зв'язку з цим, використання вебзастосунків для пошуку роботи стало ще більш актуальним. Прогресуюча цифрова трансформація у всіх сферах життя прогнозує подальше зростання попиту на ІТ-фахівців. Такі напрямки, як штучний інтелект, інтернет речей, кібербезпека та хмарні технології, будуть продовжувати збільшувати необхідність у висококваліфікованих спеціалістах у цих областях.

Об’єктом роботи є процес розробки вебзастосунку пошуку роботи для ІТ-фахівців.

Предметом кваліфікаційної роботи є інструментарій розробки вебзастостосунку пошуку роботи для ІТ-фахівців.

Метою кваліфікаційної роботи є розробка вебзастосунку для ефективного пошуку роботи в сфері ІТ, спрямованого на полегшення процесу пошуку вакансій та взаємодії між роботодавцями та ІТ-фахівцями.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1) дослідження предметної галузі згідно отриманої теми КРБ;

2) проєктування інтерфейсу користувача з урахуванням зручності використання;

3) проєктування бази даних для зберігання інформації про вакансії та користувачів;

4) імплементація алгоритмів пошуку та фільтрації вакансій;

5) тестування вебзастосунку за допомогою юніт-тестів.

Кваліфікаційна робота бакалавра викладена на 78 сторінок, містить 4 розділи, 34 ілюстрації, 4 таблиці, 16 джерел в переліку посилань.

Ключові слова: *вебзастосунок, пошук роботи, програмування на Typescript, Angular фреймворк, рекрутер, вакансії.*

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Пошук роботи – це комплексний процес, що включає вивчення різних аспектів, пов’язаних із працевлаштуванням. Цей процес спрямований на розуміння поточних тенденцій на ринку праці, вимог роботодавців, особливостей різних професій та галузей, а також на оволодіння навичками, необхідними для успішного працевлаштування.

По-перше, важливо розглянути сучасні тенденції на ринку праці. Ринок праці постійно змінюється під впливом технологічних інновацій, глобалізації та економічних факторів. Наприклад, розвиток інформаційних технологій призвів до появи нових професій, таких як спеціалісти з кібербезпеки, аналітики даних та розробники штучного інтелекту. Знання цих тенденцій допоможе кандидатам краще зрозуміти, які навички та компетенції є найбільш затребуваними. Окрім того, варто враховувати географічні особливості ринку праці, адже вимоги та можливості можуть суттєво відрізнятися залежно від регіону [1].

По-друге, вивчення вимог роботодавців є ключовим елементом підготовки до пошуку роботи. Роботодавці зазвичай мають конкретні очікування щодо кваліфікацій, досвіду роботи та особистих якостей кандидатів. Для успішного працевлаштування важливо вміти правильно презентувати свої навички та досвід, адаптуючи резюме та супровідні листи до вимог конкретних вакансій. Аналіз вакансій, які цікавлять кандидата, допоможе зрозуміти, які компетенції слід підкреслити у своїх документах та під час співбесіди. Також корисним може бути вивчення відгуків про компанії та їх корпоративні культури, що допоможе краще підготуватися до співбесіди.

Третім аспектом є розуміння особливостей різних професій та галузей. Кожна галузь має свої специфічні вимоги та перспективи розвитку. Наприклад, медична сфера вимагає високої кваліфікації та спеціалізованих знань, тоді як в креативних індустріях важливішими можуть бути творчі здібності та

інноваційний підхід. Проведення глибокого аналізу галузі, в якій кандидат планує працювати, допоможе краще орієнтуватися у професійних можливостях та вимогах. Важливо також досліджувати, які технології та методики використовуються в конкретній галузі, і бути готовим до їх освоєння.

Окрім цього, важливо розвивати навички ефективного пошуку роботи. Це включає вміння складати професійні резюме та супровідні листи, проводити успішні співбесіди, а також використовувати різноманітні платформи для пошуку роботи, такі як LinkedIn, Indeed, та спеціалізовані вебсайти. Навички нетворкінгу також є критично важливими, оскільки багато вакансій заповнюються через особисті зв'язки та рекомендації. Вміння будувати та підтримувати професійні зв'язки може суттєво збільшити шанси на успіх у пошуку роботи. Крім того, варто використовувати різні методи пошуку роботи, включаючи ярмарки вакансій, професійні конференції та спеціалізовані курси [2].

Важливою частиною підготовки до пошуку роботи є також саморефлексія та професійний розвиток. Кандидати повинні постійно оцінювати свої навички та визначати області для вдосконалення. Це може включати проходження додаткових курсів, отримання сертифікацій або участь у професійних тренінгах. Безперервне навчання та розвиток допоможуть кандидатам залишатися конкурентоспроможними на ринку праці. Також варто враховувати важливість особистісного розвитку, наприклад, розвивати комунікативні навички, емоційний інтелект та стресостійкість [2].

Таким чином, процес пошуку роботи – це багатогранний процес, що включає вивчення ринку праці, вимог роботодавців, особливостей різних професій та галузей, а також розвиток особистих та професійних навичок. Цей підхід допоможе кандидатам не лише знайти роботу, але й побудувати успішну кар'єру в довгостроковій перспективі. Комплексний підхід до пошуку роботи забезпечує глибоке розуміння власних можливостей і потреб ринку, що є запорукою успіху в сучасному конкурентному середовищі.

IT-ринок в Україні є одним з найбільш динамічних і швидкозростаючих в економіці країни. Українська IT-індустрія відзначається високим рівнем професіоналізму спеціалістів, значними обсягами експорту послуг та активним розвитком технологій. Станом на 2024 рік, IT-індустрія в Україні демонструє стабільне зростання. За даними Асоціації IT Ukraine, обсяг експорту IT-послуг за 2023 рік перевищив 7 мільярдів доларів США. Це свідчить про високий попит на українські IT-послуги з боку закордонних замовників, що є значним джерелом валютних надходжень для країни. В Україні працює понад 200 тисяч IT-спеціалістів, і цей показник постійно зростає завдяки активному розвитку освітніх програм та ініціатив з підготовки кадрів.

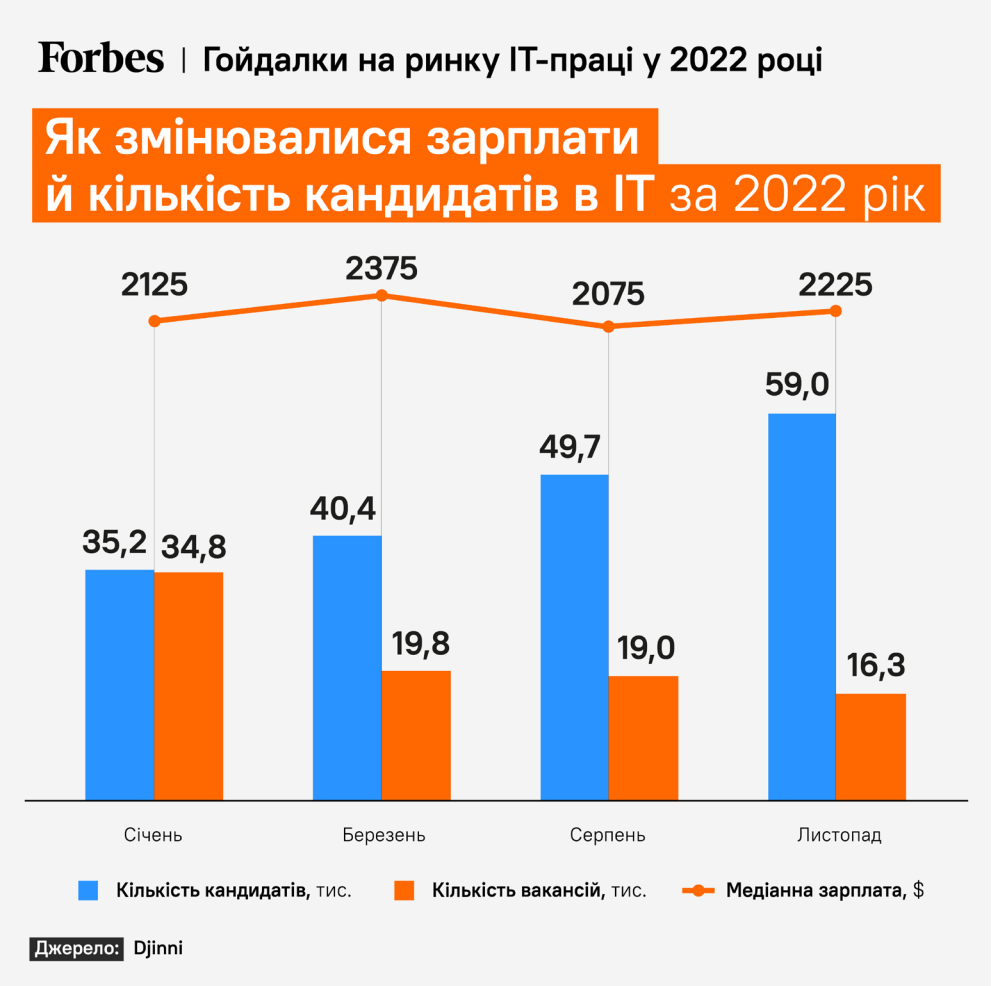

Рисунок 1.1 – Графік ринку праці в IT

Однією з ключових тенденцій на IT-ринку України є зростання популярності аутсорсингу. Багато міжнародних компаній обирають Україну як центр для розробки програмного забезпечення завдяки високій кваліфікації

місцевих спеціалістів та конкурентним цінам на їхні послуги. Крім того, зростає попит на спеціалістів у галузях штучного інтелекту, машинного навчання, кібербезпеки та аналізу даних. Також спостерігається активний розвиток стартапів, що отримують фінансову підтримку від венчурних інвесторів.

Рисунок 1.2 – Графік експорту в IT

Незважаючи на позитивну динаміку, український IT-ринок стикається з низкою викликів. Одним з основних є відтік кадрів за кордон, що пов'язано з кращими умовами праці та вищими зарплатами в інших країнах. Крім того, політична та економічна нестабільність в Україні може негативно впливати на інвестиційний клімат. Ще одним викликом є необхідність постійного вдосконалення освітніх програм для відповідності швидко змінюваним вимогам ринку [3].

Перспективи розвитку IT-ринку в Україні залишаються сприятливими. Очікується подальше зростання обсягів експорту IT-послуг та збільшення кількості фахівців у галузі. Важливим фактором буде розвиток інноваційних

технологій та підтримка стартап-екосистеми. Збільшення інвестицій у дослідження та розробки, а також підтримка державою ІТ-сектору можуть сприяти подальшому зростанню галузі. Крім того, розширення співпраці з міжнародними партнерами та участь у глобальних проєктах можуть підсилити позиції України на світовому ІТ-ринку [4].

В цілому, ІТ-ринок України характеризується високим потенціалом для зростання та розвитку. Завдяки значному кадровому потенціалу, активній підтримці з боку держави та міжнародних інвесторів, Україна має всі шанси стати одним з провідних гравців на глобальному ринку інформаційних технологій.

1.2 Огляд застосунків-аналогів

Вебзастосунок пошуку роботи для ІТ-фахівців – це онлайн-сервіс, розроблений для сприяння пошуку вакансій та залучення талановитих ІТ-професіоналів до різних компаній. Цей застосунок створений з метою спростити процес пошуку роботи та підвищити ефективність знаходження відповідних вакансій для індивідуальних потреб кожного ІТ-спеціаліста. В сучасному світі, де ІТ-індустрія швидко розвивається та зростає, важливо мати засіб, який допоможе знаходити вакансії та залучати талановитих фахівців. Застосунок пошуку роботи для ІТ фахівців актуальний, оскільки він надає зручний і ефективний спосіб знаходження відповідних робочих місць для індивідуальних потреб.

Головною метою вебзастосунка для пошуку роботи для ІТ-фахівців є забезпечення зручного та ефективного інструменту для знаходження вакансій та пропозицій роботи в ІТ-сфері. Ця платформа повинна мати зручний користувацький інтерфейс з можливістю пошуку, фільтрації та детального аналізу вакансій, а також систему реєстрації та авторизації користувачів для збереження профілів та налаштувань пошуку. Крім того, важливими аспектами є адаптивність для різних пристроїв, безпека даних користувачів та ефективна архітектура з можливістю масштабування для оптимальної роботи платформи при збільшенні обсягу даних та користувачів [4].

Існують різні вебзастосунки для пошуку роботи в IT-сфері, такі як LinkedIn, Indeed, Glassdoor, Stack Overflow Jobs, Djinni, Dou тощо. Кожен з них має свої особливості та переваги, але загальна мета всіх одна – допомогти знаходити та отримувати вакансії для IT-фахівців [5].

DOU

DOU (Distributed Open University) – це популярний український вебзастосунок, який поєднує в собі ресурси для пошуку роботи, інформаційний портал та спільноту IT-спеціалістів. DOU став відомим завдяки своїм новинам, статтям, блогам, форумам та іншим інформаційним ресурсам для професіоналів у сфері інформаційних технологій в Україні [6].

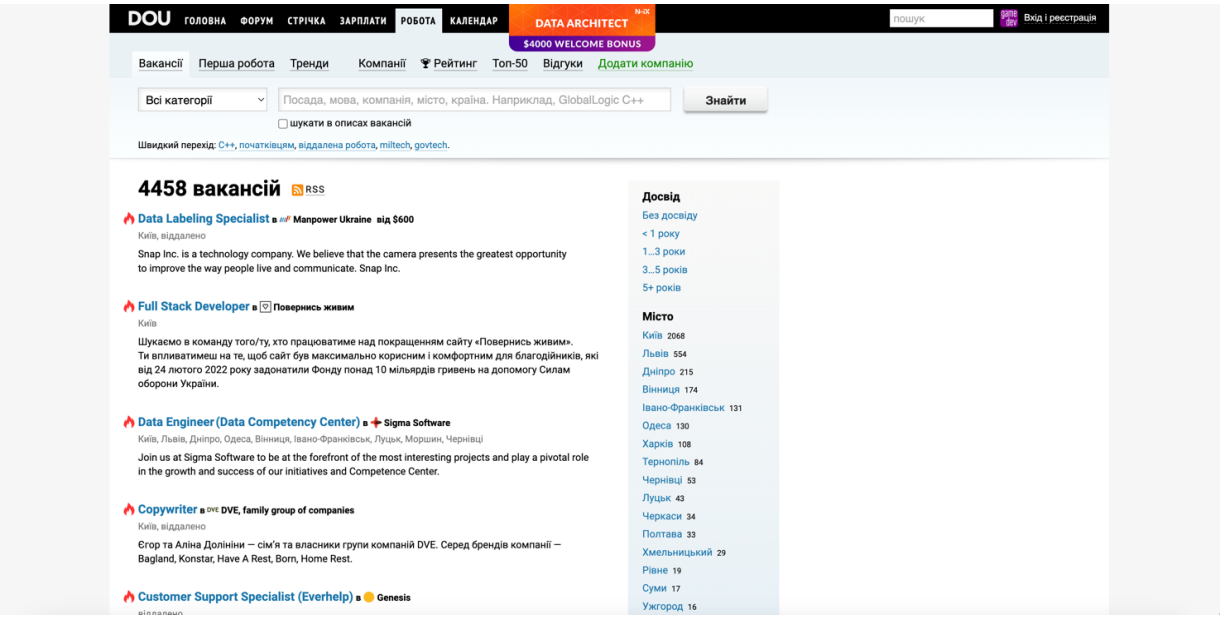

Рисунок 1.3 – Скріншот вебсайту jobs.dou.ua

Таблиця 1.1 – Опис вебзастосунку DOU

Назва	**jobs.dou.ua**
Архітектура	Клієнт-сервер
Виробник	Компанія «DOU» (ДОУ)
Мова реалізації	PHP, Laravel, Vue.js

Кінець таблиці 1.1

Функції	– пошук вакансій; – створення резюме; – контакти з роботодавцями; – аналітика ринку праці; – допомога в пошуку роботи.
Переваги	– велика кількість вакансій; – якість вакансій; – зручний пошук; – безкоштовний доступ.
Недоліки	– наявність спаму; – не завжди актуальна інформація; – не завжди зручний інтерфейс.
Посилання	jobs.dou.ua: https://jobs.dou.ua/

Djinny

Djinny – це український вебзастосунок для пошуку роботи та розвитку кар'єри в сфері ІТ. Створений командою фахівців, які розуміють потреби і особливості індустрії, Djinny поєднує в собі елементи пошуку вакансій, освітні ресурси та можливості розвитку.

З урахуванням швидкого розвитку технологій та популярності індустрії ІТ, вебзастосунок Djinny має великий потенціал та актуальність. Він допомагає спрощувати пошук роботи, підвищувати кваліфікацію та будувати успішну кар'єру у сфері інформаційних технологій [7].

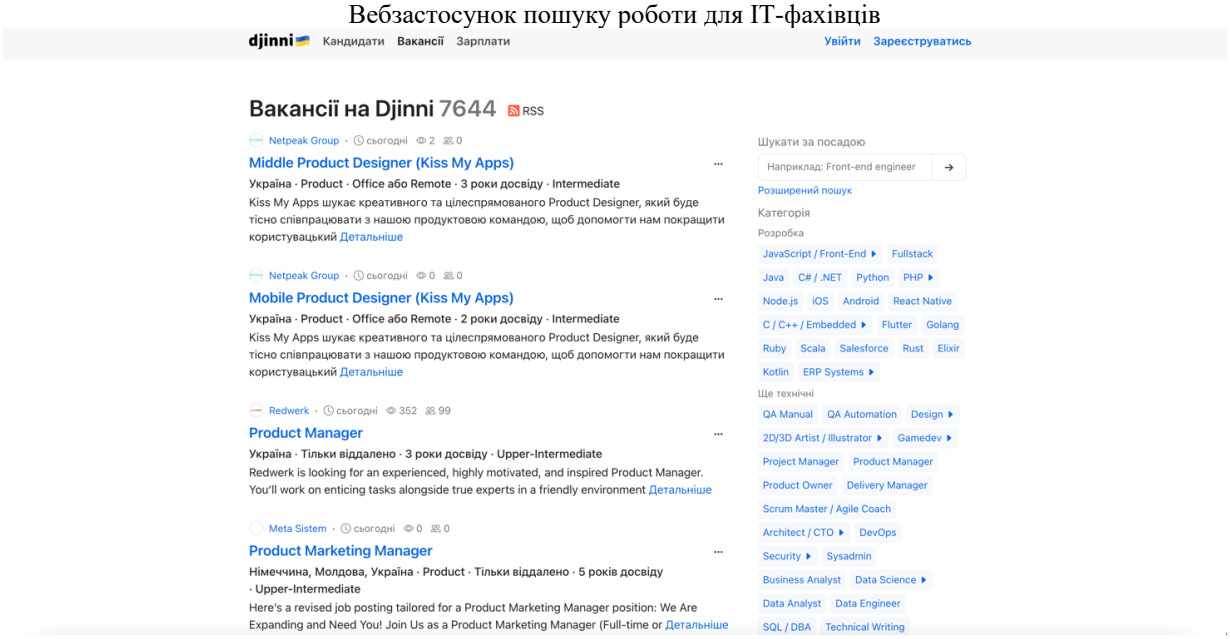

Рисунок 1.4 – Скріншот вебсайту djinni.co

Таблиця 1.2 – Опис вебзастосунку Djinny

Назва	**djinni.co**
Архітектура	Клієнт-сервер
Виробник	Компанія «Djinni»
Мова реалізації	React.js, Next,js
Функції	– пошук вакансій; – створення резюме; – контакти з роботодавцями; – аналітика ринку праці; – допомога в пошуку роботи.
Переваги	– велика кількість вакансій; – якість вакансій; – зручний пошук; – безкоштовний доступ; – анонімний пошук.
Недоліки	– наявність спаму; – не завжди актуальна інформація.
Посилання	djinni.co: https://djinni.co/

Linkedin

LinkedIn – це найбільша світова соціальна мережа для професійних контактів та розвитку кар'єри. Заснований у 2003 році, LinkedIn став основним інструментом для підтримки професійних зв'язків, пошуку роботи, рекрутингу, освіти та бізнесу [8].

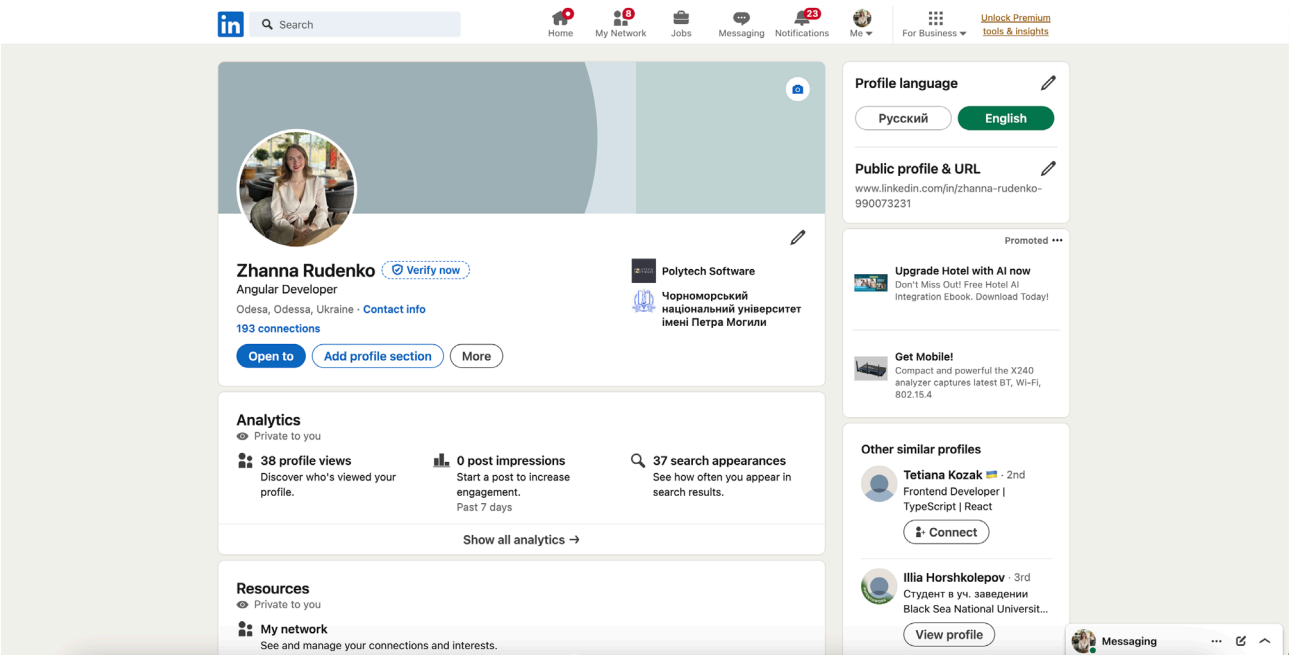

Рисунок 1.5 – Скріншот вебсайту linkedin.com

Таблиця 1.3 – Опис вебзастосунку Linkedin

Назва	**linkedin.com**
Архітектура	Клієнт-сервер
Виробник	Компанія «LinkedIn Corporation»
Мова реалізації	Java, Spring
Функції	– пошук вакансій; – створення резюме; – контакти з роботодавцями; – аналітика ринку праці; – допомога в пошуку роботи.

Кінець таблиці 1.3

Переваги	– велика база користувачів; – професійні мережі; – інформація про ринок праці; – допомога в пошуку роботи.
Недоліки	– платні послуги; – застарілий інтерфейс; – не зручний пошук вакансій.
Посилання	linkedin.com: https://www.linkedin.com

1.3 Специфікація вимог

Призначення проєкту

Призначення системи: розробка вебзастосунку для пошуку роботи для ІТ-фахівців з метою полегшення пошуку вакансій та сприяння кар'єрному зростанню.

Межі проєкту:

– крайня дата завершення робіт: 17.06.2024р.;

– технології: Angular для фронтенду, Express.js для бекенду, MongoDB для зберігання даних.

Загальний опис:

Сфера застосування: вебзастосунок призначений для об'єднання вакансій для ІТ-фахівців з різних джерел та підвищення доступності цієї інформації.

Функції системи:

1) пошук вакансій

– опис: користувачі можуть шукати вакансії за ключовими словами, локацією, зарплатним діапазоном тощо;

– вхідна інформація: параметри пошуку;

– вихідна інформація: список вакансій, що відповідають критеріям пошуку.

2) фільтрація результатів

– опис: можливість фільтрувати вакансії за різними категоріями (типом роботи, рівнем досвіду тощо);

– вхідна інформація: список вакансій;

– вихідна інформація: відфільтрований список вакансій.

3) авторизація та реєстрація користувачів

– опис: користувачі можуть створювати облікові записи та авторизуватися для збереження налаштувань пошуку та перегляду історії.

Вимоги до інформаційного забезпечення:

– джерела даних: інформація про вакансії буде отримуватися з різних вебсайтів та API;

– зберігання та обробка даних: MongoDB використовуватиметься для зберігання та управління даними.

Вимоги до технічного забезпечення:

– операційна система: будь-яка, що підтримує Node.js та MongoDB;

– ресурси: система повинна мати достатньо ресурсів для виконання Node.js, MongoDB та Angular.

Вимоги до програмного забезпечення:

– архітектура програмної системи: клієнт-серверна архітектура з Angular на фронтенді, Express.js на бекенді та MongoDB як база даних;

– мови та технології розробки: Angular, Express.js, MongoDB;

– серверне програмне забезпечення: Node.js для запуску Express.js.

Вимоги до зовнішніх інтерфейсів:

– інтерфейс користувача: зручний та інтуїтивно зрозумілий інтерфейс користувача, розроблений з використанням Angular та Bootstrap.

Властивості програмного забезпечення:

– доступність: застосунок має бути доступним для користувачів за будь-яких умов, за наявності Інтернет-з'єднання;

– супровід: мінімальний супровід від розробника, за винятком необхідних оновлень та підтримки;

– переносимість: сумісність з будь-якою ОС, що підтримує Node.js та MongoDB;

– продуктивність: застосунок повинен бути ефективним та швидким у відгуках на запити користувачів;

– надійність: захищений доступ до адміністративних функцій та даних користувачів;

– безпека: забезпечення безпеки даних за допомогою аутентифікації та авторизації користувачів.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи було проаналізовано предметну сферу, існуючі застосунки-аналоги, після чого було створено специфікацію вимог, у якій було зазначено наступні вимоги, функції та властивості ПЗ:

– призначення та межі проєкту;

– загальний опис;

– функції системи;

– вимоги до інформаційного забезпечення;

– вимоги до технічного забезпечення;

– вимоги до програмного забезпечення;

– вимоги до зовнішніх інтерфейсів;

– властивості програмного забезпечення.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Створення сценаріїв

Usecase – це метод моделювання в системному аналізі та проєктуванні програмного забезпечення, який описує поведінку системи з точки зору зовнішнього користувача. Use case представляє собою специфікацію того, як система взаємодіє з зовнішніми агентами для виконання певних дій або досягнення певних цілей [9].

Usecase допомагає узагальнити вимоги користувачів та визначити функціональність системи з точки зору їхнього використання. Вони є важливим інструментом для розробки програмного забезпечення, оскільки допомагають розуміти потреби користувачів і визначити, як система повинна поводитися в різних ситуаціях [10].

Коротка форма:

Usecase: Зареєструватися в системі як ІТ-фахівець

Актор: ІТ-фахівець

Приклад:

ІТ-фахівець, хоче знайти роботу за допомогою вебзастосунку пошуку роботи для ІТ-фахівців. Він відвідує вебсайт системи і натискає кнопку «Зареєструватися». Він заповнює форму реєстрації, вказавши своє ім'я, електронну адресу та пароль. Він підтверджує свою електронну адресу, натиснувши посилання, яке було надіслано йому електронною поштою. Після цього ІТ-фахівець успішно зареєстрований в системі і отримує доступ до своїх облікових даних.

Поверхнева форма:

Usecase: Пошук роботи для ІТ-фахівців

Суть: ІТ-фахівець шукає роботу за допомогою вебзастосунку.

Користувачі: ІТ-фахівець

Передумови: ІТ-фахівець має реєстрацію в вебзастосунку.

Основний успішний сценарій:

1) ІТ-фахівець відкриває вебзастосунок;

2) ІТ-фахівець переходить на сторінку пошуку роботи;

3) ІТ-фахівець вводить у пошукову форму бажану посаду, місцезнаходження, зарплату та інші критерії;

4) ІТ-фахівець натискає кнопку «Пошук»;

5) система відображає список вакансій, які відповідають критеріям пошуку;

6) ІТ-фахівець переглядає список вакансій;

7) ІТ-фахівець може натиснути на посилання на вакансію, щоб переглянути її детальніше;

8) ІТ-фахівець може зберегти вакансію;

9) якщо ІТ-фахівець зацікавлений у вакансії, він може надіслати своє резюме роботодавцю.

Альтернативні сценарії

Сценарій 1: ІТ-фахівець не має реєстрації в вебзастосунку. У цьому випадку він повинен створити обліковий запис, перш ніж зможе використовувати функцію пошуку роботи.

Сценарій 2: ІТ-фахівець не вводить у пошукову форму жодних критеріїв. У цьому випадку система відображає список всіх вакансій, які є в базі даних.

Сценарій 3: ІТ-фахівець ввів у пошукову форму невірні дані. У цьому випадку система відображає повідомлення про помилку.

Сценарій 4: ІТ-фахівець не знайшов жодної вакансії, яка відповідає його критеріям. У цьому випадку система відображає повідомлення про те, що вакансій не знайдено.

Сценарій 5: ІТ-фахівець не зацікавлений у жодній з вакансій, які він знайшов. У цьому випадку він може закрити вебзастосунок.

Повна форма:

Таблиця 2.1 – Повна форма

Usecase section	**Comment**
Use Case Name	Створити профіль ІТ-фахівця
Scope	System
Level	User-goal
Primary Actor	ІТ-фахівець
Stakeholders and interests	1) ІТ-фахівець: хоче створити профіль у вебзастосунку пошуку роботи, щоб знайти роботу; 2) вебзастосунок пошуку роботи: підвищення клієнтури вебзастосунку, надання можливості ІТ-фахівцям знаходити роботу.
Preconditions	1) ІТ-фахівець не має створеного профілю в вебзастосунку пошуку роботи; 2) ІТ-фахівець має акаунт у вебзастосунку пошуку роботи.
Success guarantee	1) Після успішного завершення usecase у вебзастосунку пошуку роботи буде створений профіль ІТ-фахівця.
Main Success Scenario	1) ІТ-фахівець відкриває вебзастосунок пошуку роботи; 2) ІТ-фахівець натискає кнопку «Створити профіль»; 3) ІТ-фахівець заповнює форму для створення профілю 4) ІТ-фахівець натискає кнопку «Створити»; 5) вебзастосунок перевіряє, чи всі обов'язкові поля в формі для створення профілю були заповнені; 6) якщо всі обов'язкові поля були заповнені, вебзастосунок перевіряє, чи дані, введені в форму для створення профілю, є вірними; 7) якщо дані, введені в форму для створення профілю, є вірними, вебзастосунок створює новий профіль ІТ-фахівця; 8) вебзастосунок зберігає дані профілю ІТ-фахівця в базі даних; 9) вебзастосунок підтверджує створення профілю;

Продовження таблиці 2.1

Extensions	1) якщо ІТ-фахівець не заповнив усі обов'язкові поля в формі для створення профілю, вебзастосунок виведе повідомлення про помилку; 2) якщо ІТ-фахівець вводить невірні дані в форму для створення профілю, вебзастосунок виведе повідомлення про помилку; 3) якщо ІТ-фахівець вже має створений профіль у вебзастосунку пошуку роботи, вебзастосунок виведе повідомлення про те, що профіль вже існує; 4) якщо ІТ-фахівець не має акаунту у вебзастосунку пошуку роботи, вебзастосунок виведе повідомлення про те, що необхідно створити акаунт; 5) якщо ІТ-фахівець не має доступу до вебзастосунку пошуку роботи, вебзастосунок виведе повідомлення про помилку.
Special Requirements	1) вебзастосунок повинен забезпечувати безпечне зберігання даних профілю ІТ-фахівця; 2) вебзастосунок повинен забезпечувати можливість створення профілю ІТ-фахівця різними мовами.
Technology and Data Variations List	1) вебзастосунок може використовувати технологію штучного інтелекту для аналізу даних профілю ІТ-фахівця та рекомендації вакансій, які відповідають його кваліфікації; 2) вебзастосунок може використовувати технологію машинного навчання для навчання моделі, яка прогнозує, чи буде ІТ-фахівець зацікавлений у конкретній вакансії; 3) вебзастосунок може використовувати дані з інших джерел, наприклад, з державних реєстрів, навчальних закладів, компаній; 4) вебзастосунок може використовувати дані з соціальних мереж, щоб отримати більш повну картину про ІТ-фахівця.
Frequency of Ocurrence	Usecase може виконуватися один раз або кілька разів на день.

Кінець таблиці 2.1

Miscellaneous	1) вебзастосунок повинен надавати можливість ІТ-фахівцеві редагувати або видаляти свій профіль; 2) вебзастосунок повинен надавати можливість ІТ-фахівцеві додавати або видаляти інформацію в свій профіль.

2.2 Створення діаграми варіантів використання

Діаграма використання (Use Case Diagram) – це вид діаграми в Unified Modeling Language (UML), яка використовується для візуалізації взаємодії між акторами та функціональними можливостями системи. Вона дозволяє моделювати функціональність системи з точки зору користувачів та зовнішніх систем [11].

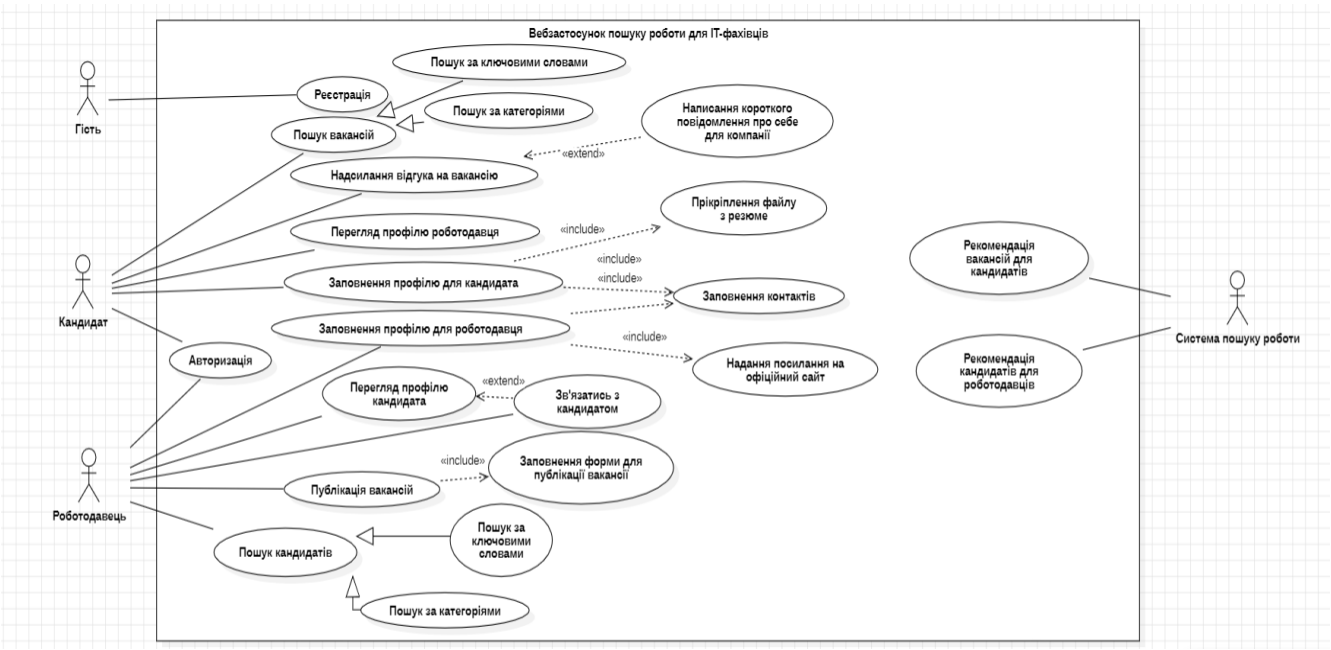

Рисунок 2.1 – Діаграма варіантів використання

Актори включають:

– гість (Guest);
– кандидат (Candidate);
– роботодавець (Employer);

– вебзастосунок пошуку роботи для IT-фахівців (Web Application for IT Professionals Job Search).

Випадки використання включають:

– реєстрація (Registration);

– пошук за ключовими словами (Search by Keywords);

– пошук за категоріями (Search by Categories);

– пошук вакансій (Search Jobs);

– надсилання відгука на вакансію (Apply for a Job);

– написання короткого повідомлення про себе для компанії (Write a Short Introduction for the Company);

– прікріплення файлу (Attach File);

– заповнення профілю для кандидата (Fill Candidate Profile);

– заповнення профілю для роботодавця (Fill Employer Profile);

– публікація вакансій (Publish Jobs);

– пошук кандидатів (Search Candidates);

– зв'язатись з кандидатом (Contact Candidate);

– надання посилання на офіційний сайт (Provide Link to Official Website);

– рекомендація кандидатів для роботодавців (Recommend Candidates for Employers).

Під час створення діаграми були використані всі види зв’язків: unidirectional association, generalization, extend relationship, include relationship.

2.3 Створення діаграм діяльності

Діаграма діяльності – це вид структурної діаграми, що використовується для моделювання послідовності дій або процесів в системі. Вона відображає різні дії та взаємозв'язки між ними, які відбуваються під час виконання конкретного завдання або процесу.

Діаграми діяльності широко використовуються в сферах програмування, бізнес-аналізу, проєктуванні програмного забезпечення, моделюванні бізнес-процесів та інших областях для візуалізації та розуміння порядку виконання

дій. Вони дозволяють чітко представити послідовність кроків у різних сценаріях та ідентифікувати можливі проблеми чи оптимізації в процесі.

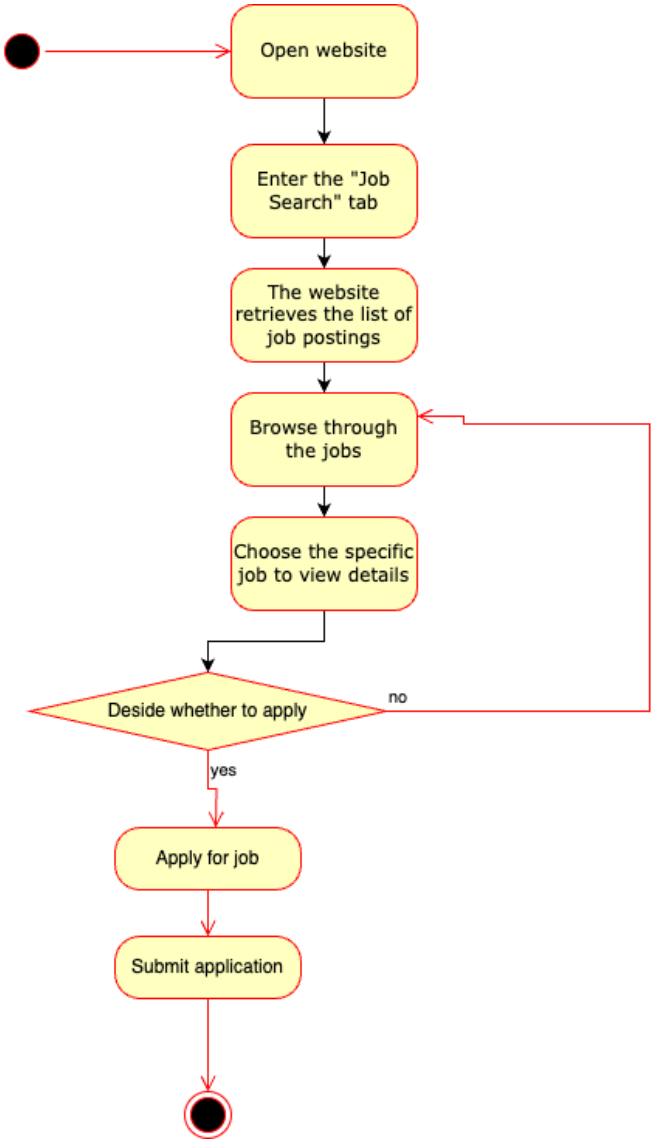

Рисунок 2.2 – Діаграма діяльності для usecase «Пошук роботи»

Діаграма діяльності для usecase «Пошук роботи» (рис. 2.2) починається з кроків «Відкрити вебсайт» та «Увійти в розділ Пошук роботи», що показують взаємодію користувача з вебсайтом пошуку роботи. Після цього вебсайт отримує список вакансій, і користувач переглядає вакансії. Після цього користувач обирає конкретну вакансію для перегляду деталей і вирішує, чи подати заявку. Якщо користувач вирішує подати заявку, він подає заявку на роботу і надсилає заявку.

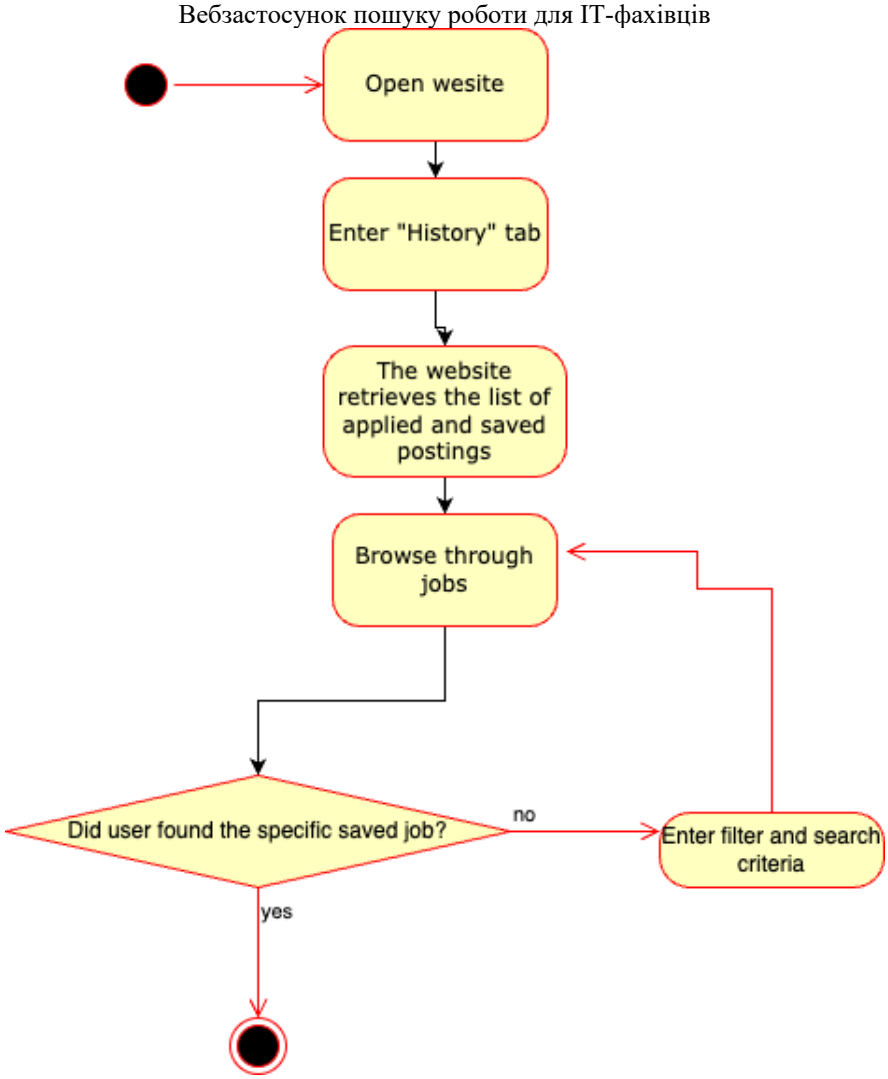

Рисунок 2.3 – Діаграма діяльності для usecase «Пошук збереженої вакансії»

Діаграма діяльності для usecase «Пошук збереженої вакансії» (рис. 2.3) починається з того, що користувач відкриває вебсайт, де можна шукати роботу. Далі користувач переходить в розділ «Історія», щоб переглянути раніше подані та збережені вакансії. Вебсайт отримує список поданих та збережених оголошень для користувача. Користувач переглядає список оголошень про роботу. Якщо користувач шукає конкретну збережену вакансію, він може ввести критерії фільтрації та пошуку, щоб знайти її. Потім користувач може виконати додаткові дії з оголошенням, такі як подання або редагування своєї заявки.

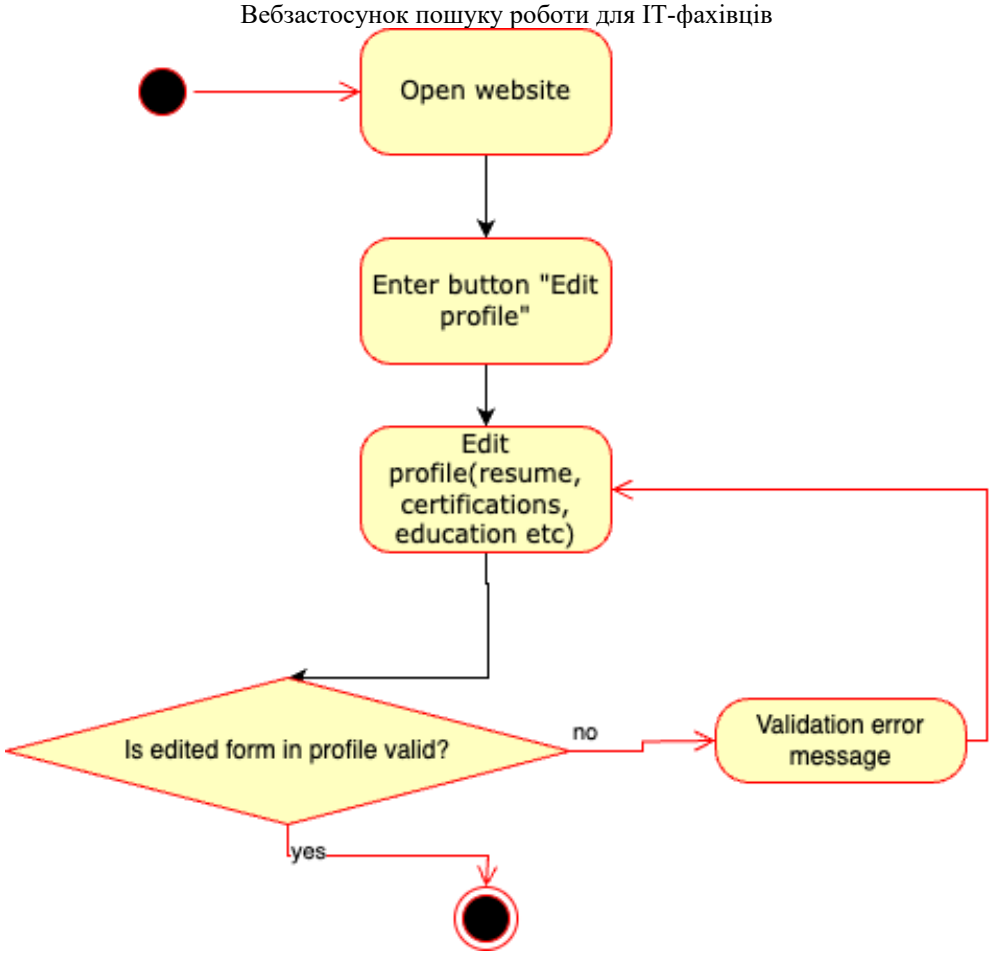

Рисунок 2.4 – Діаграма діяльності для usecase «Редагування профілю»

Діаграма діяльності для usecase «Редагування профілю» (рис. 2.4) починається з того, що користувач відкриває вебсайт, де вони можуть шукати роботу. Користувач переходить в розділ «Історія», щоб переглянути свої раніше подані та збережені оголошення про роботу. Вебсайт отримує список поданих та збережених оголошень про роботу для користувача. Користувач переглядає список оголошень про роботу. Якщо користувач шукає конкретну збережену вакансію, вони можуть ввести критерії фільтрації та пошуку, щоб знайти її. Потім користувач може виконати додаткові дії з оголошенням про роботу, такі як подання або редагування своєї заявки.

2.4 Створення DFD

DFD (Data Flow Diagram) – це діаграма, яка відображає потоки даних та процеси, що їх обробляють, в інформаційній системі або бізнес-процесі. DFD складається з різних елементів, таких як процеси, потоки даних, сховища даних та зовнішні сутності.

На діаграмі потоків даних елементи системи зображуються у вигляді процесів, зовнішніх сутностей, даних та зв'язків між ними. Процеси обробляють дані, а дані переміщуються між процесами та зовнішніми сутностями. Стрілки вказують напрямок потоку даних.

DFD широко використовується при аналізі та проєктуванні інформаційних систем для виявлення потоків даних, виділення функцій системи та виявлення взаємозв'язків між елементами системи. Це допомагає створити більш чітке уявлення про те, як працює система, та ідентифікувати можливі покращення.

Була створена DFD нульового рівня (рис. 2.5) для вебсайту пошуку роботи. Вона призначена для швидкого огляду кваліфікацій, заробітних плат та резюме, які відображають систему як єдиний процес високого рівня із зв'язком із зовнішніми об'єктами: роботою, кандидатом та компанією. Це має бути легко зрозумілим для широкої аудиторії, включаючи робітників, компанії та кваліфікації [12].

Об’кти та потік процесів системи пошуку роботи:

1) керування всіма вакансіями;
2) керування всіма заявками;
3) керування всіма компаніями;
4) керування всіма заявками на вакансії;
5) керування всіма кваліфікаціями;
6) керування всіма зарплатами;
7) керування всіма резюме.

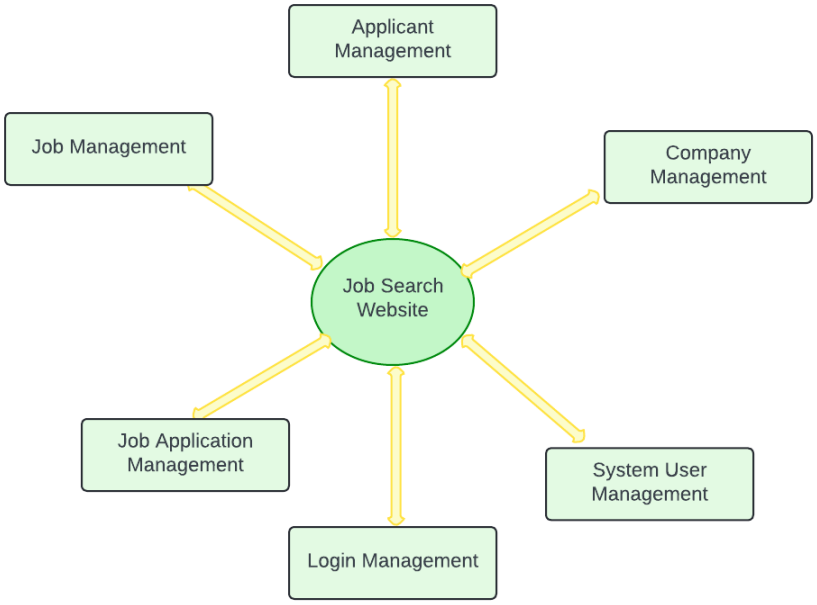

Рисунок 2.5 – DFD нульового рівня

DFD першого рівня (рис 2.6) для вебсайту пошуку роботи може виглядати наступним чином:

1) **реєстрація користувача та увійти в систему**:

– зовнішній агент: користувач;

– процес: авторизація та аутентифікація;

– потік даних: дані користувача, включаючи ім'я, електронну адресу та пароль;

– внутрішнє сховище даних: база даних користувачів.

2) **пошук вакансій**:

– зовнішній агент: користувач;

– процес: пошук відповідних вакансій;

– потік даних: критерії пошуку (наприклад, регіон, галузь, рівень зарплати).

– внутрішнє сховище даних: база даних вакансій.

3) **подання заявок на вакансії**:

– зовнішній агент: користувач;

– процес: подання заявки на вакансії;

– поток даних: вибрана вакансія, дані про користувача;

– внутрішнє сховище даних: база даних заявок.

4) **додавання вакансій компанією**:

– зовнішній агент: компанія;

– процес: додавання нових вакансій до системи;

– потік даних: інформація про вакансію (наприклад, назва посади, опис роботи);

– внутрішнє сховище даних: база даних вакансій.

5) **управління резюме користувачів**:

– зовнішній агент: користувач;

– процес: додавання, редагування та видалення резюме;

– потік даних: інформація про резюме користувача;

– внутрішнє сховище даних: база даних резюме.

Ці основні функції покривають ключові аспекти роботи і забезпечують взаємодію між користувачами та системою для ефективного пошуку роботи та кандидатів [9].

Об’кти та потік процесів системи пошуку роботи:

1) обробка записів вакансій та отримання вакансій;

2) обробка записів аплікантів та отримання всіх аплікантів;

3) обробка записів компаній та отримання всіх компаній;

4) обробка записів вакансій аплікантів та отримання всіх вакансій аплікантів;

5) обробка записів кваліфікацій та отримання всіх кваліфікацій;

6) обробка записів заробітної плати та отримання всіх записів заробітної плати;

7) обробка записів резюме та отримання всіх резюме.

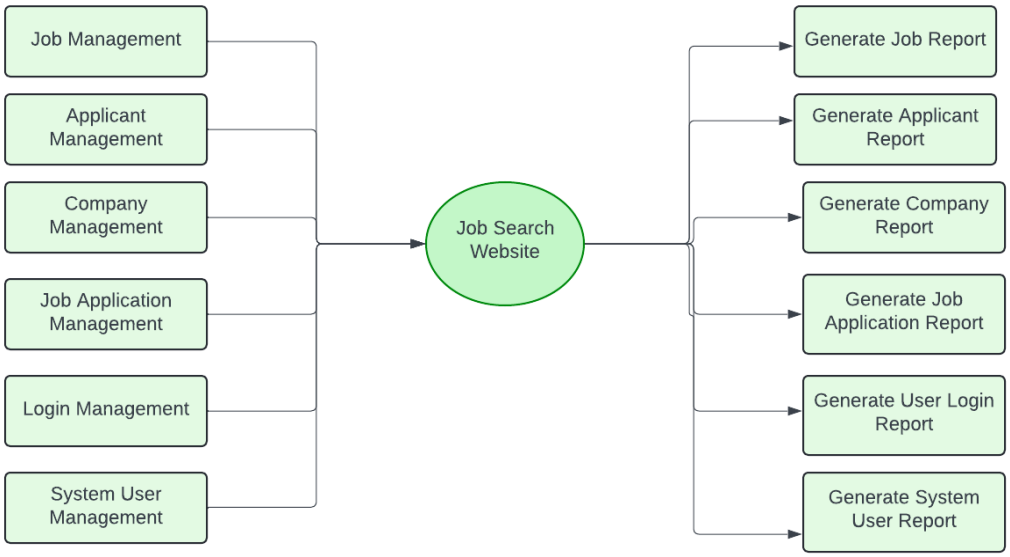

Рисунок 2.6 – DFD першого рівня

DFD другого рівня (рис. 2.7) для вебсайту пошуку роботи:

1) **обробка резюме**:

– зовнішній агент: користувач;

– процес: перегляд, редагування та видалення резюме;

– потік даних: інформація про резюме користувача;

– внутрішнє сховище даних: база даних резюме.

2) **обробка заробітної плати**:

– зовнішній агент: адміністратор;

– процес: додавання, редагування та видалення записів заробітної плати;

– потік даних: інформація про заробітну плату;

– внутрішнє сховище даних: база даних заробітної плати.

3) **обробка кваліфікацій**:

– зовнішній агент: користувач;

– процес: додавання, редагування та видалення кваліфікаційних даних;

– потік даних: інформація про кваліфікацію;

– внутрішнє сховище даних: база даних кваліфікацій.

4) **обробка кандидатів на роботу**:

– зовнішній агент: компанія;

– процес: перегляд, редагування та видалення даних кандидатів на роботу;

– потік даних: інформація про кандидата на роботу;

– внутрішнє сховище даних: база даних кандидатів на роботу.

5) **обробка компаній**:

– зовнішній агент: адміністратор;

– процес: додавання, редагування та видалення даних компаній;

– потік даних: інформація про компанії;

– внутрішнє сховище даних: база даних компаній.

6) **обробка претендентів**:

– зовнішній агент: користувач;

– процес: реєстрація, авторизація та управління обліковим записом претендента;

– потік даних: дані про претендента;

– внутрішнє сховище даних: база даних претендентів.

Ці підсистеми розширюють функціональність, дозволяючи керувати різними аспектами системи, такими як резюме, заробітна плата, кваліфікація та інші [12].

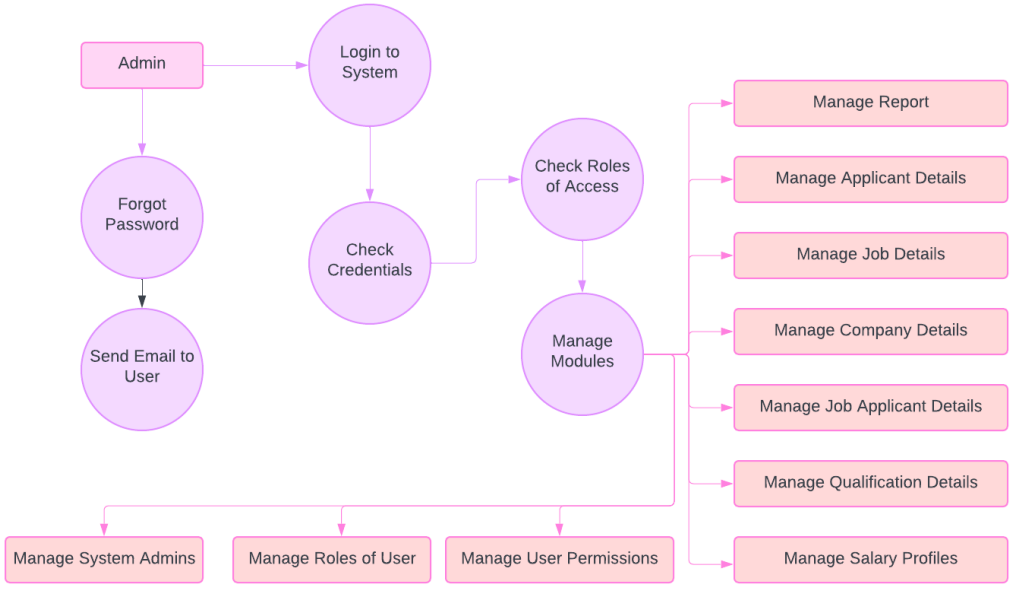

Рисунок 2.6 – DFD другого рівня

Об’кти та потік процесів системи пошуку роботи:

1) адміністратор увіймає до системи та керує всіма функціональностями порталу вакансій;

2) адміністратор може додавати, редагувати, видаляти та переглядати записи про вакансії, компанії, кваліфікації та резюме;

3) адміністратор може керувати всіма деталями щодо аплікантів, кандидатів на роботу та заробітної плати;

4) адміністратор також може генерувати звіти про вакансії, аплікантів, компанії, кандидатів на роботу, кваліфікації та заробітної плати;

5) адміністратор може шукати деталі щодо аплікантів, кваліфікацій та заробітної плати;

6) адміністратор може застосовувати різні рівні фільтрів до звітів про вакансії, кандидатів на роботу та кваліфікації;

7) адміністратор може відстежувати детальну інформацію про аплікантів, компанії, кандидатів на роботу та кваліфікації.

Висновки до розділу 2

У другому розділі було описано моделювання та планування програмного забезпечення. Були виконані такі завдання:

– були побудовані три форми usecase: коротка, поверхнева та повна;

– була створена діаграма використання системи;

– були побудовані три діаграми діяльності: «Пошук роботи», «Пошук збереженої вакансії», «Редагування профілю»;

– були створені діаграми DFD нульового, першого та другого рівнів.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ТЕХНОЛОГІЙ

3.1 Огляд технологій

Створення вебзастосунку для пошуку роботи, особливо в сфері IT, може бути досить складним завданням. Проте, з використанням правильного стеку технологій, цей процес може бути значно спрощений. Нижче наведено огляд основних технологій, які можна використати для побудови такого застосунку, а саме: Angular, Express.js та MongoDB.

Frontend: Angular

Angular – це популярний фреймворк для розробки односторінкових вебзастосунків, створений і підтримуваний Google [13]. Ось деякі з його ключових особливостей:

1) **компонентна архітектура**:

– Angular використовує компонентний підхід, що дозволяє створювати повторно використовувані та легко підтримувані частини застосунку [14].

2) **реактивне програмування**:

– завдяки використанню RxJS, Angular дозволяє ефективно працювати з асинхронними даними та подіями.

3) **вбудовані інструменти**:

– Angular включає в себе різноманітні інструменти для форм, роутингу, валідації, анімації та багато іншого.

Backend: Express.js

Express.js – це легкий та гнучкий фреймворк для Node.js, який надає широкий набір інструментів для створення вебзастосунків та API [15]. Основні переваги Express.js:

1) **простота та гнучкість**:

– Express.js забезпечує мінімалістичну структуру, яка дозволяє розробникам мати повний контроль над архітектурою застосунку.

2) **розширюваність**:

– багато плагінів та middleware, які можна легко інтегрувати для розширення функціоналу.

3) **підтримка RESTful API**:

– Express.js ідеально підходить для створення RESTful сервісів, що є важливим для сучасних вебзастосунків.

Database: MongoDB

MongoDB – це документно-орієнтована база даних NoSQL, яка зберігає дані у вигляді JSON-подібних документів [16]. Ключові особливості MongoDB:

1) **гнучкість моделі даних**:

– документна модель дозволяє зберігати складні дані у вигляді вкладених структур без необхідності нормалізації, як у реляційних базах даних.

2) **масштабованість**:

– MongoDB підтримує горизонтальне масштабування, що дозволяє легко обробляти великі обсяги даних.

3) **швидкість**:

– MongoDB оптимізована для швидких операцій з читанням та записом, що є важливим для застосунків з високою продуктивністю.

3.2 Діаграма класів

Діаграма класів – це один з основних видів діаграм, використовуваних в об'єктно-орієнтованому моделюванні (ООМ) і є частиною мови UML (Unified Modeling Language). Вона дозволяє візуалізувати структуру системи, показуючи класи, їх атрибути, методи, а також відношення між ними.

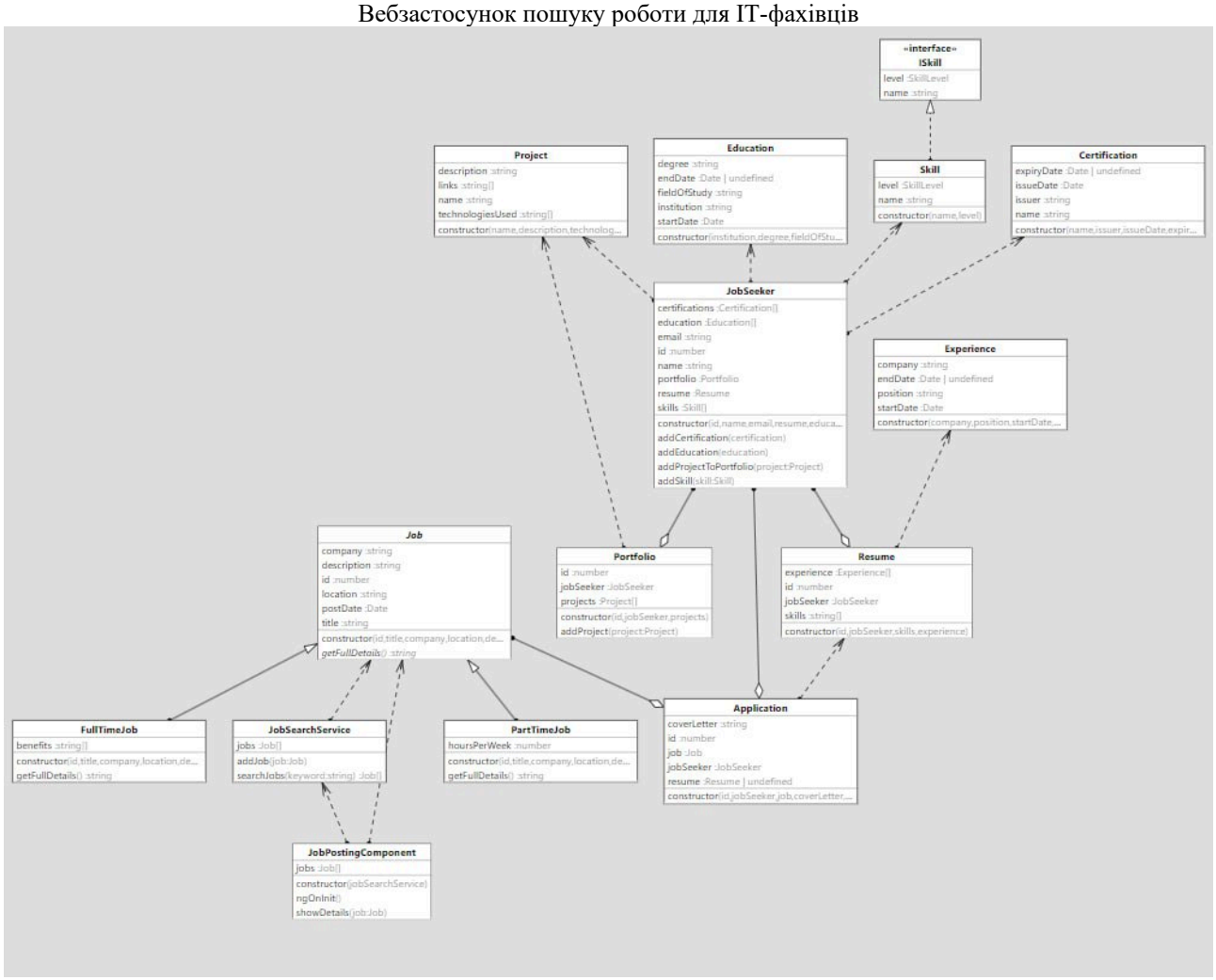

Рисунок 3.1 – Діаграма класів вебзастосунку

Діаграма класів для вебзастосунку пошуку роботи для IT-фахівців (рис. 3.1) представляє складну структуру взаємодії між різними сутностями. Вона включає кілька основних класів, кожен з яких має свої атрибути, методи та взаємозв'язки з іншими класами.

Клас JobSeeker описує шукача роботи, включаючи інформацію, як ім'я, email, резюме, освіта, сертифікації, портфоліо та навички. Він містить методи для додавання інформації про освіту, сертифікації, навички та проєкти до портфоліо. JobSeeker має агрегації з класами Resume, Education, Certification, Portfolio та Skill, що означає, що один шукач роботи може мати одне резюме, декілька освіт, сертифікацій, навичок та одне портфоліо.

Клас Resume є композицією з класом JobSeeker, що вказує на тісний зв'язок між резюме та шукачем роботи. Resume містить інформацію про навички та досвід шукача роботи. Portfolio включає декілька проєктів, що

представлено агрегацією з класом Project. Кожен проєкт описується через назву, опис, технології та посилання на вебсайт або код.

Клас Job є батьківським для двох підкласів: FullTimeJob та PartTimeJob. Job містить загальну інформацію про вакансію, як назва, компанія, локація, опис та дата публікації. Він має абстрактний метод getFullDetails для отримання повних деталей вакансії у певному форматі. FullTimeJob та PartTimeJob успадковують від Job і містять додаткову інформацію відповідно про повну зайнятість (перелік пільг) та неповну зайнятість (кількість годин на тиждень), з реалізацією методу getFullDetails для відображення цієї додаткової інформації.

Клас Application описує заявку на вакансію. Він включає інформацію про шукача роботи, вакансію, супровідний лист та резюме (необов'язково). Application асоціюється з класами JobSeeker і Job, вказуючи, що одна вакансія може мати декілька заявок, і один шукач роботи може мати декілька заявок на різні вакансії.

Класи Education, Certification, Skill та Project описують відповідно освіту, сертифікації, навички та проєкти шукача роботи. Education та Certification містять інформацію про освіту та сертифікації, а Skill описує навичку шукача роботи з рівнем володіння, використовуючи клас SkillLevel, який перераховує можливі рівні володіння навичкою.

Інстанціювання кожного з цих класів передбачає створення об'єктів з певними атрибутами: JobSeeker створюється з ім'ям, email, резюме, освітою, сертифікаціями, портфоліо та навичками; Job з назвою, компанією, локацією, описом та датою публікації; Application з шукачем роботи, вакансією, супровідним листом та резюме (необов'язково); Resume з навичками та досвідом; Education та Certification з відповідною інформацією; Skill з навичкою та рівнем володіння; Project з назвою, описом, технологіями та посиланнями; Portfolio з шукачем роботи та його проєктами.

3.3 Діаграма впровадження

Діаграма впровадження (Deployment Diagram) є одним із видів діаграм, що використовується в мові моделювання UML (Unified Modeling Language). Вона призначена для візуалізації фізичного розгортання програмного забезпечення на апаратних компонентах системи. Діаграма впровадження показує, як програмні компоненти розміщуються на апаратних вузлах (ноди), та їх взаємозв'язки.

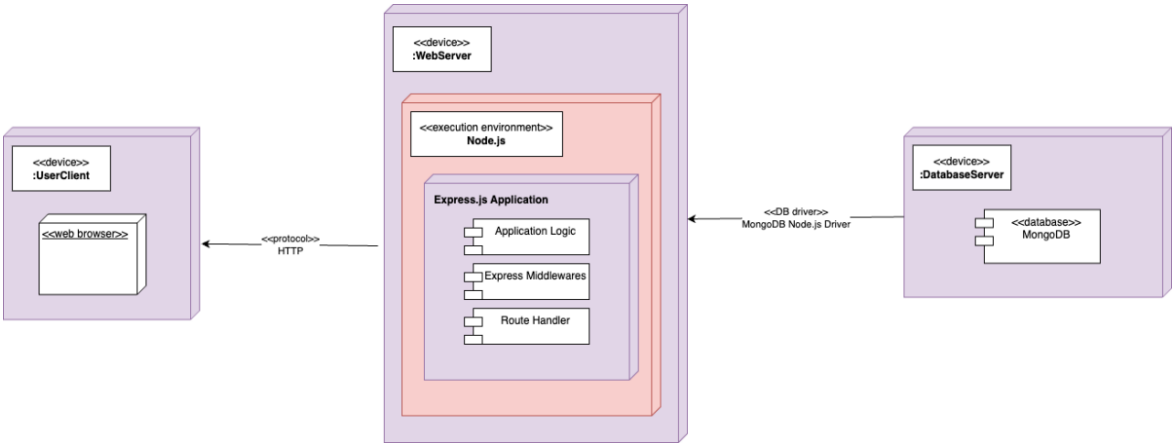

Рисунок 3.2 – Діаграма впровадження вебзастосунку

Діаграма впровадження вебзастосунку пошуку роботи для IT фахівців (рис. 3.2), побудованого з використанням Angular, Express.js та MongoDB, відображає архітектуру, що забезпечує взаємодію між різними компонентами системи.

Користувач починає взаємодію з вебзастосунком через клієнтську частину, реалізовану на Angular. Ця частина відповідає за відображення інтерфейсу користувача та забезпечення динамічної взаємодії. Користувач може реєструватися, входити в систему, шукати вакансії та переглядати свій профіль.

Коли користувач виконує якусь дію, наприклад, натискає кнопку пошуку вакансій, Angular відправляє відповідний HTTP-запит до серверної частини, побудованої на Express.js. Express.js є бекенд-фреймворком, який обробляє ці запити, маршрутизує їх до відповідних контролерів і виконує необхідні бізнес-операції. Контролери в Express.js відповідають за виконання конкретних

завдань, таких як отримання даних від бази даних, обробка цих даних та повернення результатів клієнтській частині.

Серверна частина використовує MongoDB як базу даних для зберігання інформації про користувачів, вакансії та інші важливі дані. Коли Express.js отримує запит, який вимагає взаємодії з базою даних, наприклад, для пошуку вакансій, він надсилає відповідний запит до MongoDB. MongoDB обробляє цей запит і повертає результати до серверної частини.

Отримавши відповідь від бази даних, Express.js обробляє дані, форматує їх і відправляє назад до клієнтської частини Angular. Angular, у свою чергу, оновлює інтерфейс користувача відповідно до отриманих даних, наприклад, відображає список вакансій, що відповідають критеріям пошуку користувача.

Таким чином, вебзастосунок забезпечує інтерактивний та зручний інтерфейс для користувачів, які шукають роботу в IT-сфері, забезпечуючи надійну взаємодію між клієнтською частиною, серверною частиною та базою даних.

3.4 Діаграми компонентів та пакетів

Діаграма компонентів (Component Diagram) є одним з типів діаграм у мові моделювання UML (Unified Modeling Language). Вона використовується для відображення організації та залежностей між різними компонентами системи. Основна мета діаграми компонентів – показати, як різні частини програмного забезпечення взаємодіють одна з одною.

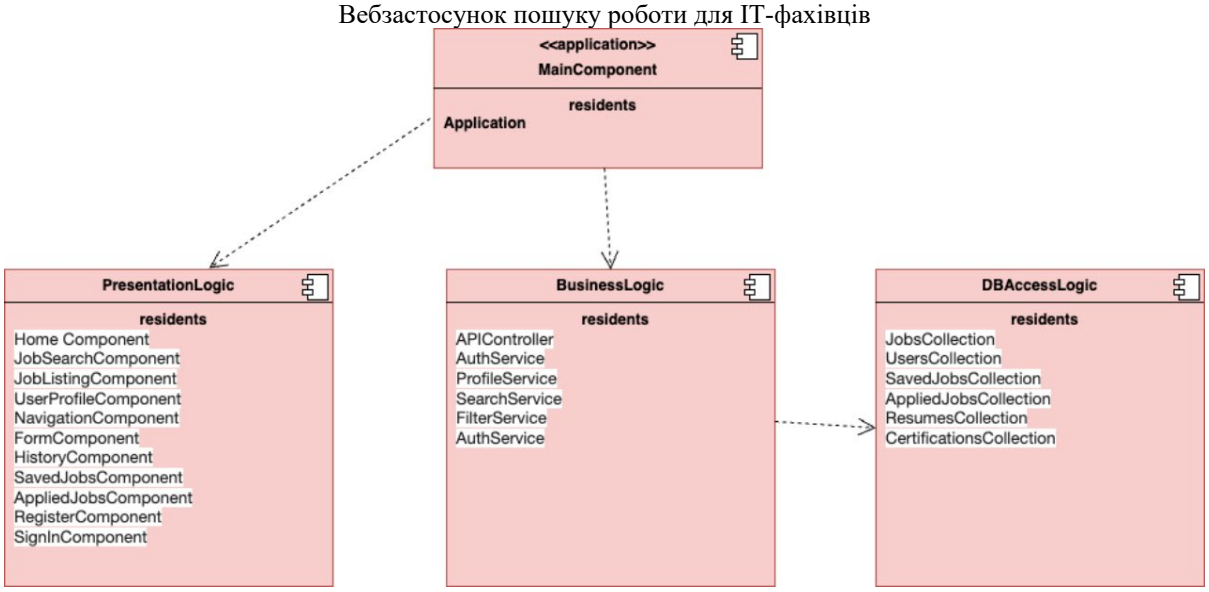

Рисунок 3.3 – Діаграма компонентів

Діаграма компонентів, представлена на рисунку 3.3, описує архітектуру програмного забезпечення, що складається з трьох основних компонентів:

– **MainComponent (Головний компонент)**: Цей компонент відповідає за загальну структуру та функціональність системи. Він взаємодіє з іншими компонентами для отримання та відображення даних, а також для забезпечення навігації та інших системних функцій.

– **PresentationLogic (Логіка презентації)**: Цей компонент відповідає за те, як дані системи візуалізуються користувачеві. Він отримує дані від BusinessLogic, форматує їх та відображає у відповідних інтерфейсах користувача.

– **BusinessLogic (Бізнес-логіка)**: Цей компонент відповідає за основну функціональність системи. Він обробляє запити користувачів, виконує бізнес-правила та взаємодіє з базою даних для отримання та оновлення даних.

Діаграма пакетів – це вид діаграми в мові моделювання UML (Unified Modeling Language), який використовується для візуалізації структури системи з точки зору її компонентів та залежностей між ними. У UML діаграма пакетів дозволяє організовувати елементи програмного забезпечення в логічні групи (пакети), що спрощує аналіз та розробку програм.

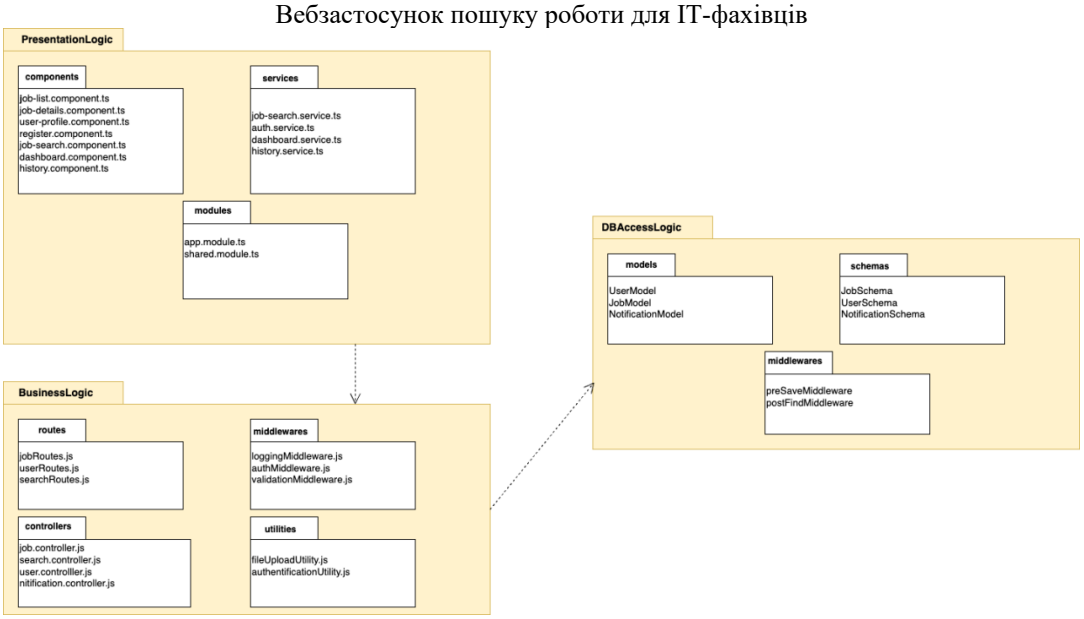

Рисунок 3.4 – Діаграма пакетів

Основні пакети:

1) **PresentationLogic:** цей пакет містить компоненти, відповідальні за візуалізацію даних користувачеві. Він поділяється на підпакети:

2) **BusinessLogic:** цей пакет містить компоненти, що реалізують бізнес-логіку системи.

3) **DBAccessLogic:** цей пакет містить компоненти, що відповідають за взаємодію з базою даних. Він поділяється на підпакети:

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було проведено спроєктовано програмне забезпечення, що є останнім та важливим етапом перед реалізацією ПЗ, завдяки ньому при розробці час буде витрачено лише на реалізацію. Під час проєктування були виконані наступні завдання:

– огляд стеку технологій;
– побудова діаграми класів;
– побудова діаграми компонентів;
– побудова діаграми пакетів;
– побудова діаграми розгортання.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Опис структури вебзастосунку

Правильна структура вебзастосунку є надзвичайно важливою з кількох причин. Вона сприяє модульності та масштабованості проєкту. Коли застосунок розбито на окремі модулі та файли, розробники можуть працювати над різними частинами проєкту незалежно один від одного, що полегшує додавання нових функціональних можливостей. Добре структурований проєкт також легше масштабувати. В майбутньому, коли виникне потреба в розширенні функціоналу або збільшенні обсягу даних, буде простіше додати нові компоненти чи модулі.

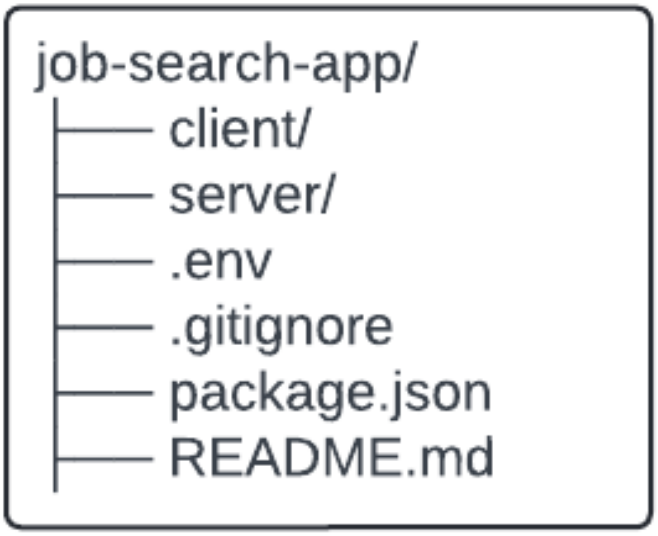

Рисунок 4.1 – Структура вебзастосунку

1. **client/:** ця директорія містить код фронтенд частини застосунку, побудованої на Angular;

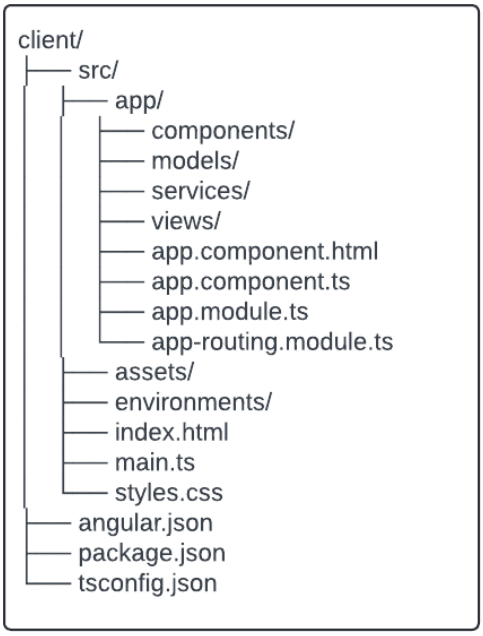

Рисунок 4.2 – Структура папки client

– **components/**: компоненти Angular (наприклад, заголовок, форма пошуку, список вакансій);

– **models/**: моделі даних (інтерфейси для вакансій, користувачів тощо);

– **services/**: сервіси для взаємодії з бекендом через HTTP-запити;

– **views/**: шаблони представлень (сторінки) застосунку;

– **environments/**: конфігурації середовищ (наприклад, dev, prod).

2. server/: ця директорія містить код бекенд частини застосунку, побудованої на Express.js.

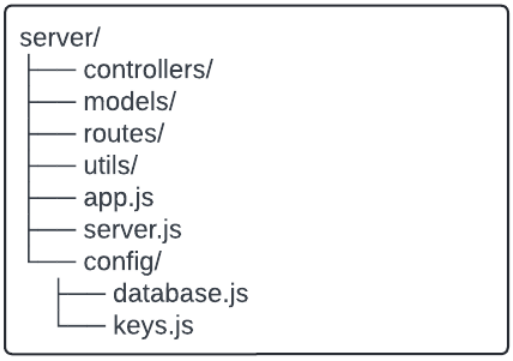

Рисунок 4.3 – Структура папки server

– **controllers/**: контролери для обробки запитів (наприклад, контролер для вакансій, користувачів);

– **models/**: моделі Mongoose для MongoDB (схеми бази даних);

– **routes/**: маршрути (роути) для обробки HTTP-запитів;

– **utils/**: утиліти та допоміжні функції;

– **app.js**: головний файл застосунку Express;

– **server.js**: файл для запуску сервера;

– **config/**: конфігураційні файли (наприклад, конфігурація бази даних, ключі для JWT).

4.2 Налаштування бази даних

MongoDB є документно-орієнтованою базою даних, що використовує модель даних, яка суттєво відрізняється від традиційних реляційних баз даних. Основною одиницею даних у MongoDB є документ, який зберігається у форматі BSON (Binary JSON). Документи можуть містити вкладені структури, що дозволяє зберігати складні ієрархічні дані. Вони схожі на JSON-об'єкти і можуть мати різні схеми. Документи групуються в колекції, яка є аналогом таблиці в реляційних базах даних, але не має фіксованої схеми, тобто документи в одній колекції можуть мати різну структуру [16].

Основні колекції для бази даних:

1) колекція «jobs»:

– **title**: назва вакансії;

– **description**: опис вакансії;

– **skills**: масив навичок, які потрібні для вакансії;

– **experience**: рівень досвіду, необхідний для вакансії;

– **location**: місце розташування вакансії;

– **salary**: заробітна плата.

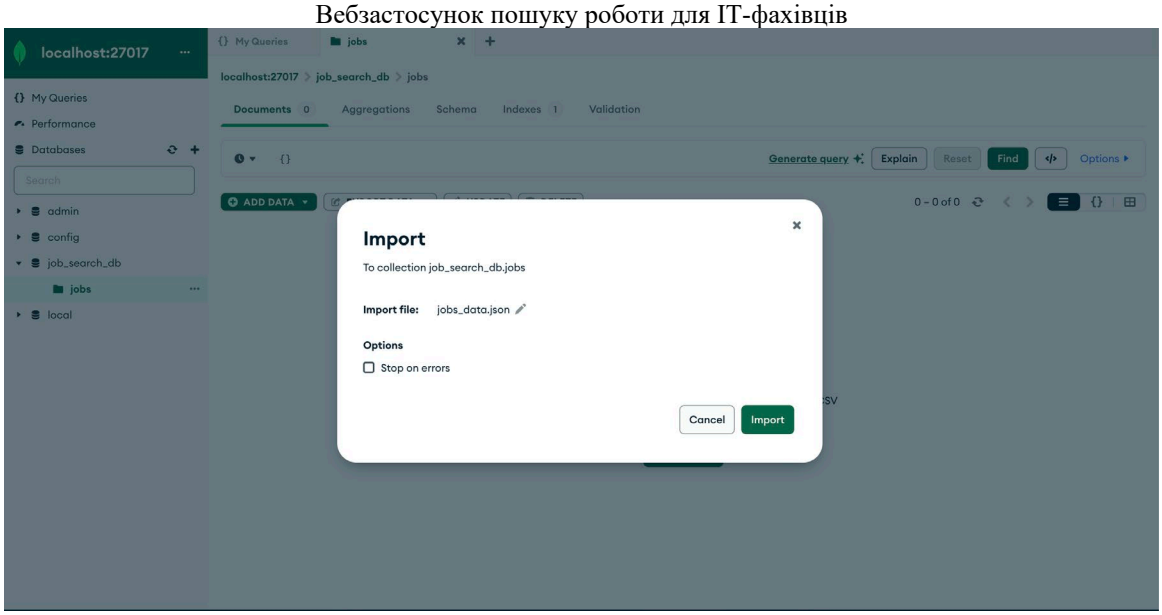

Рисунок 4.4 – Імпорт даних у колекцію **«jobs»**

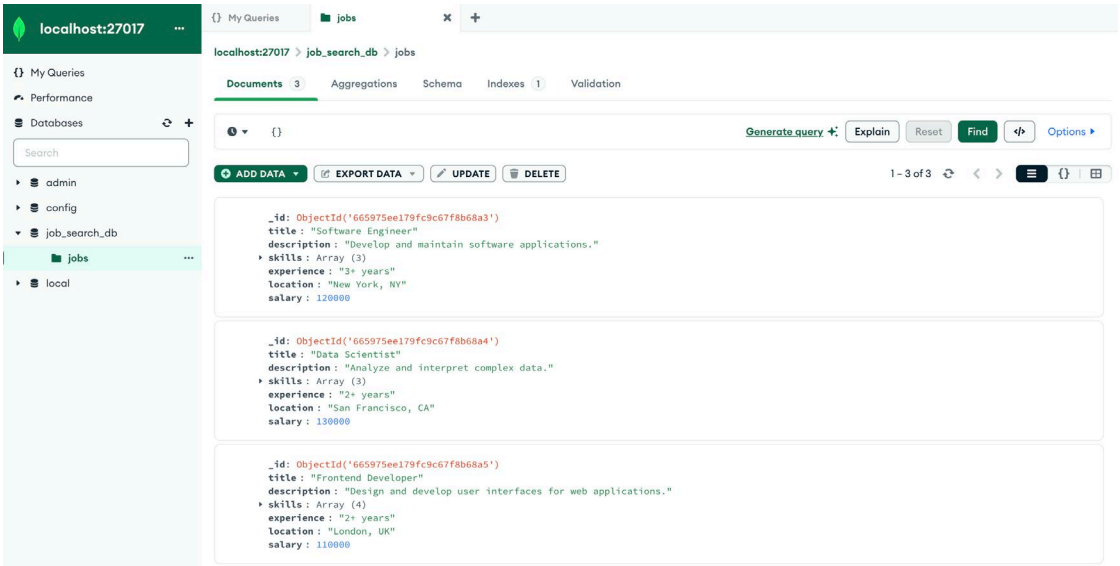

Рисунок 4.5 – Створена колекція **«jobs»**

2) колекція «candidates»:

– **name**: ім'я кандидата;

– **skills**: масив навичок кандидата;

– **experience**: рівень досвіду кандидата;

– **location**: місце розташування кандидата;

– **desiredSalary**: бажана заробітна плата кандидата.

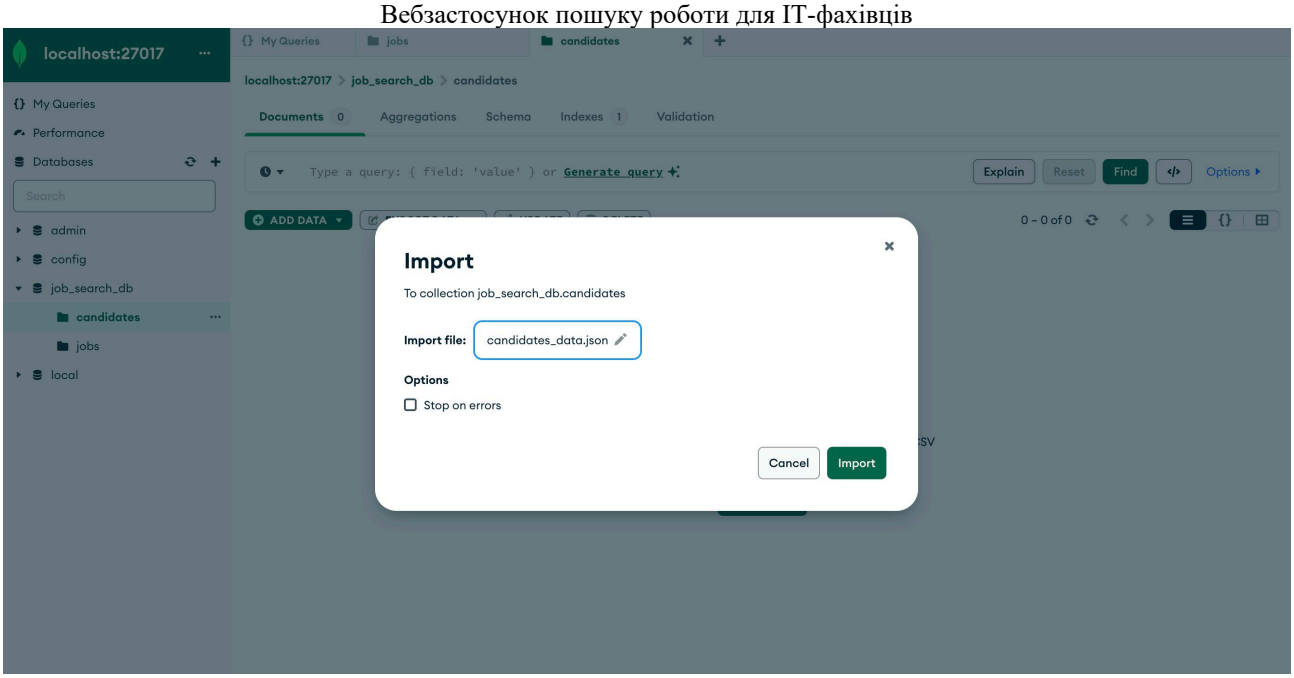

Рисунок 4.6 – Імпорт даних у колекцію **«candidates»**

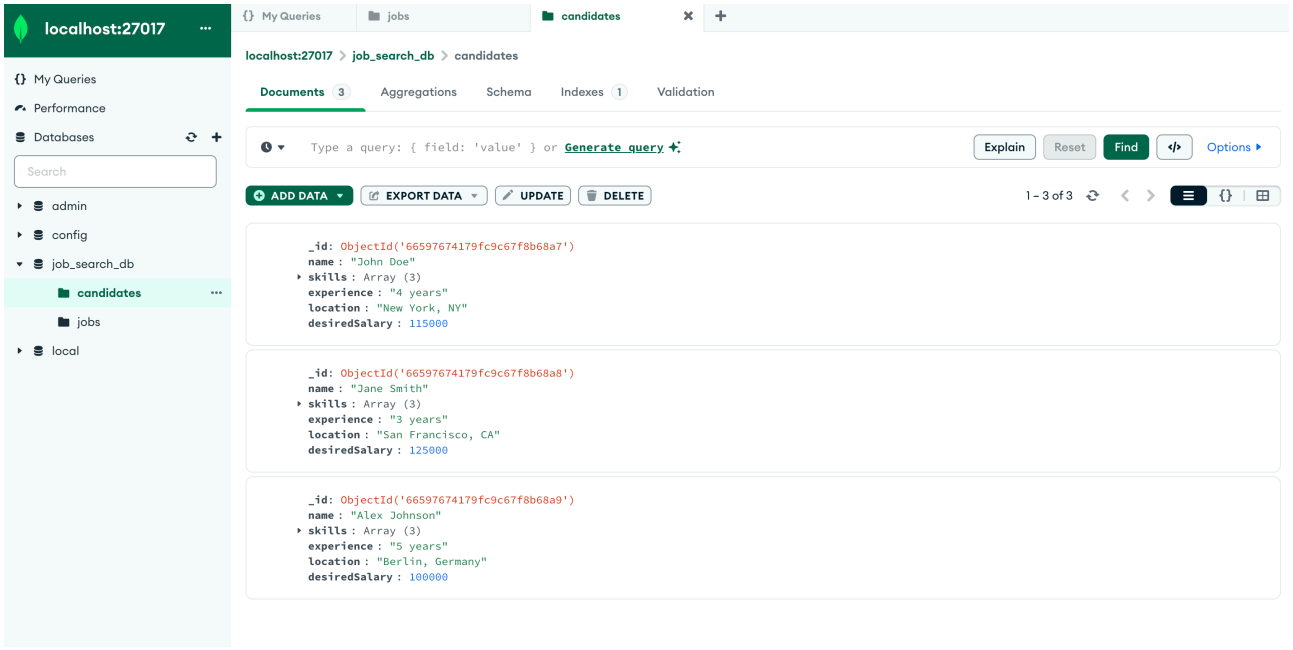

Рисунок 4.7 – Створена колекція **«candidates»**

Лістинг підключення до бази даних:

```
const mongoURI = 'mongodb://localhost:27017/job_search_db;
mongoose.connect(mongoURI, { useNewUrlParser: true, useUnifiedTopology:
true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.log('Error connecting to MongoDB:', err));
```

4.3 Опис інтерфейсу користувача

Інтерфейс користувача (UI) відіграє надзвичайно важливу роль у створенні вебзастосунків. Його значення можна розглядати з кількох основних аспектів.

По-перше, UI визначає перше враження користувача про застосунок. Інтуїтивно зрозумілий і привабливий інтерфейс здатний захопити увагу користувача з перших секунд взаємодії. Якщо застосунок виглядає професійно і має логічно організовані елементи управління, це підвищує довіру користувача до продукту і компанії, яка його розробила.

По-друге, UI впливає на зручність використання. Хороший інтерфейс дозволяє користувачам легко і швидко виконувати необхідні дії, що зменшує їх розчарування та підвищує задоволеність. Наприклад, чіткі та зрозумілі кнопки, логічні меню, інтуїтивно зрозумілі іконки та добре організовані форми допомагають користувачам легко знаходити потрібну інформацію і виконувати завдання.

Інтерфейс вебзастосунку пошуку роботи був побудований на компонентному підході (рис. 4.8).

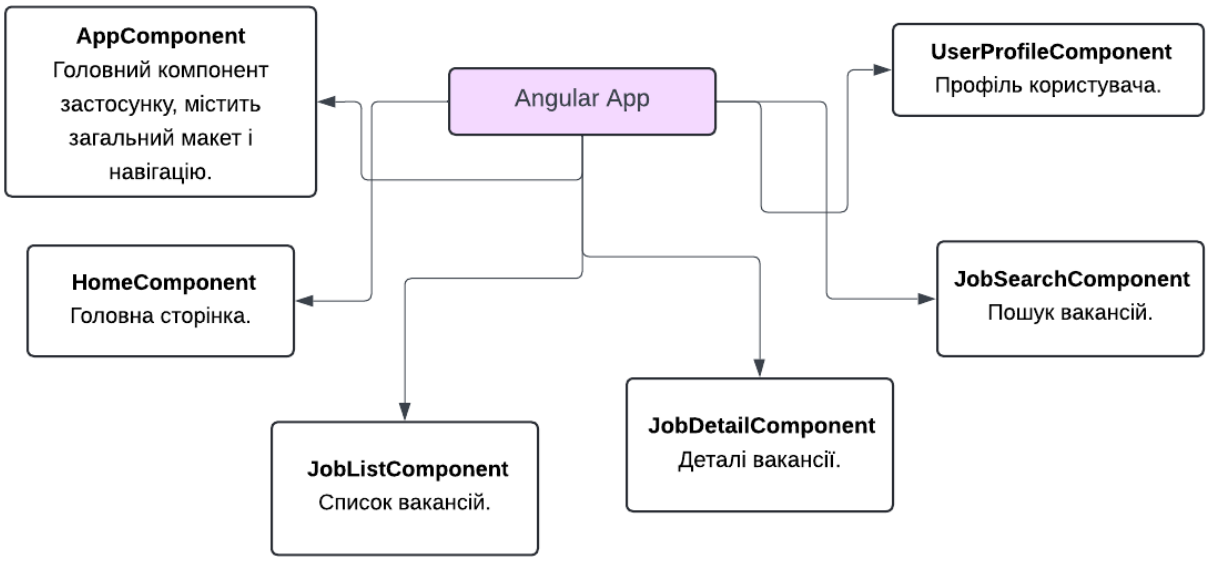

Рисунок 4.8 – Основні компоненти вебзастосунку

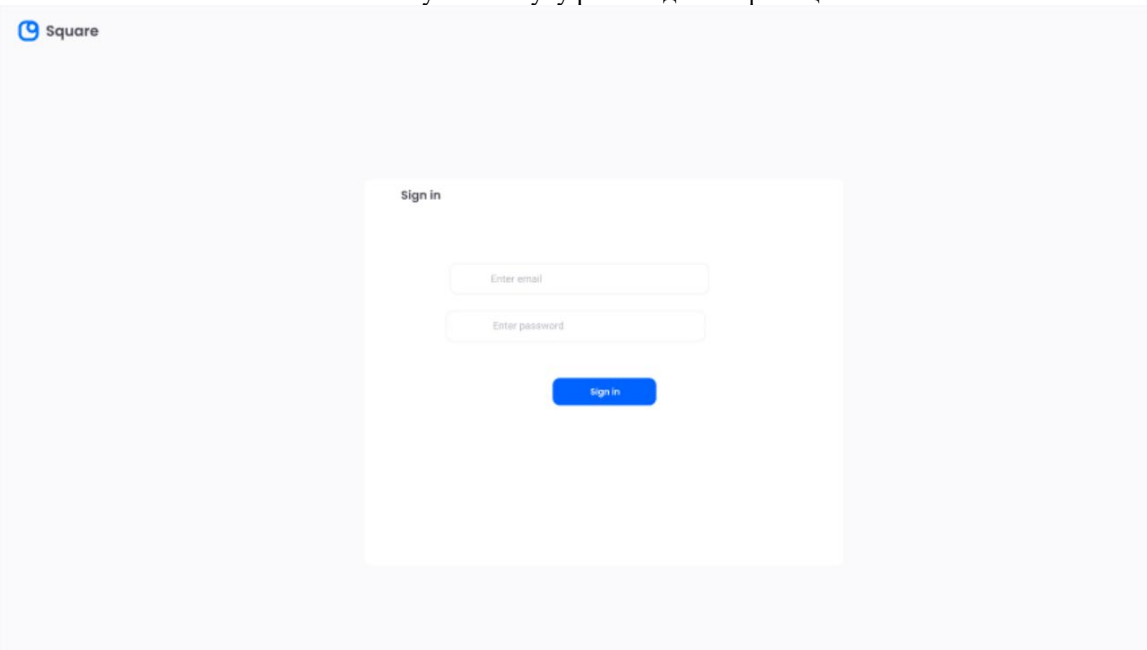

Рисунок 4.9 – Сторінка авторизації

На рисунку 4.9 зображено сторінку авторизації. На цій сторінці користувач повинен ввести свою електронну пошту та пароль. При настисканні кнопки «Sign in» користувач буде авторизований у систему.

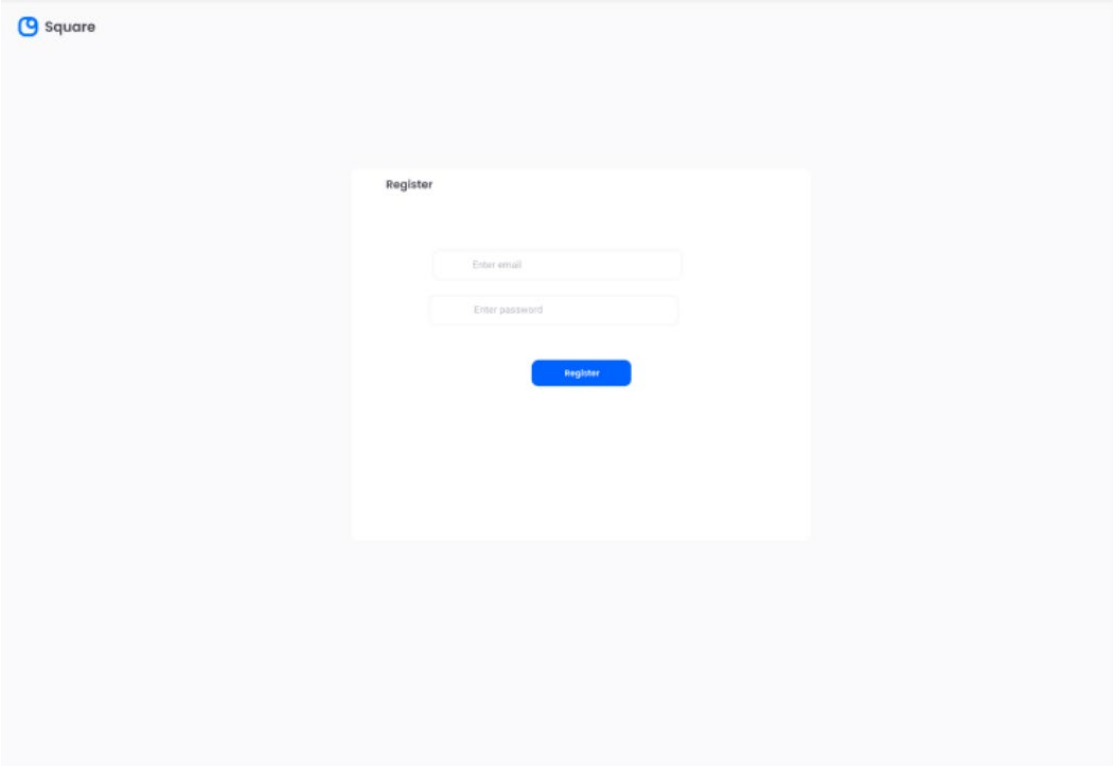

Рисунок 4.10 – Сторінка реєстрації

На рисунку 4.10 зображено сторінку реєстрації. На цій сторінці користувач повинен ввести свою електронну пошту та пароль. При настисканні кнопки «Register» користувач буде зареєстрований та перейде до екрану авторизації.

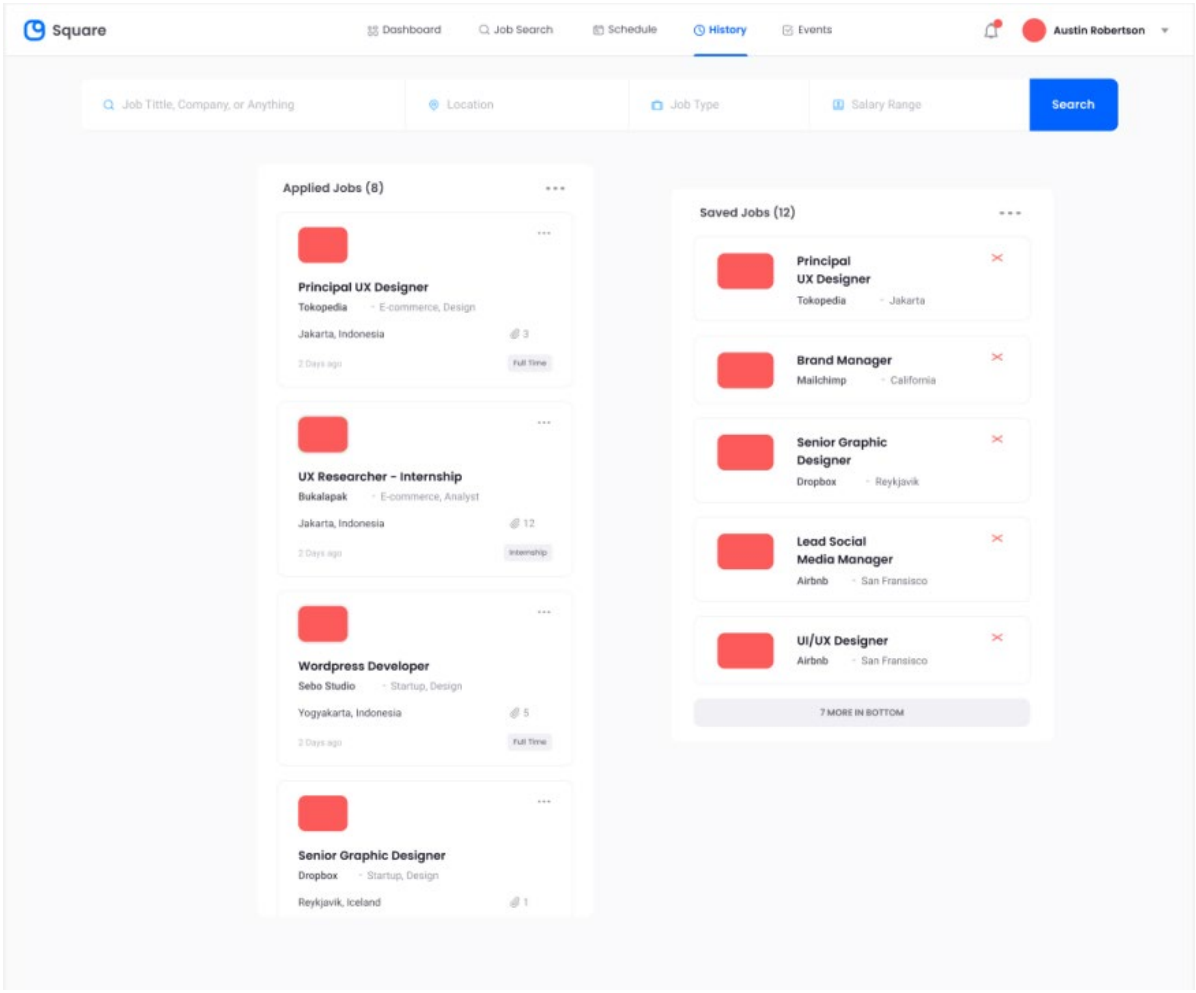

Рисунок 4.11 – Сторінка історії користувача

На даному екрані (рис. 4.11) користувач може переглянути історію вакансій, на які він подав заявку, а також список вакансій, які він зберіг. Також зазначені кількість збережених позицій та вакансій, на які кандидат подав заявку. У списку вакансій зазначено назву вакансії та коротку інформацію про неї, також при кліку можна перейти на сторінку цієї вакансії. Зверху знаходиться тулбар, за яким можна відфільтрувати вакансії за типом, локацією та заробітньою платою. Зверху знаходиться основний хедер сайту.

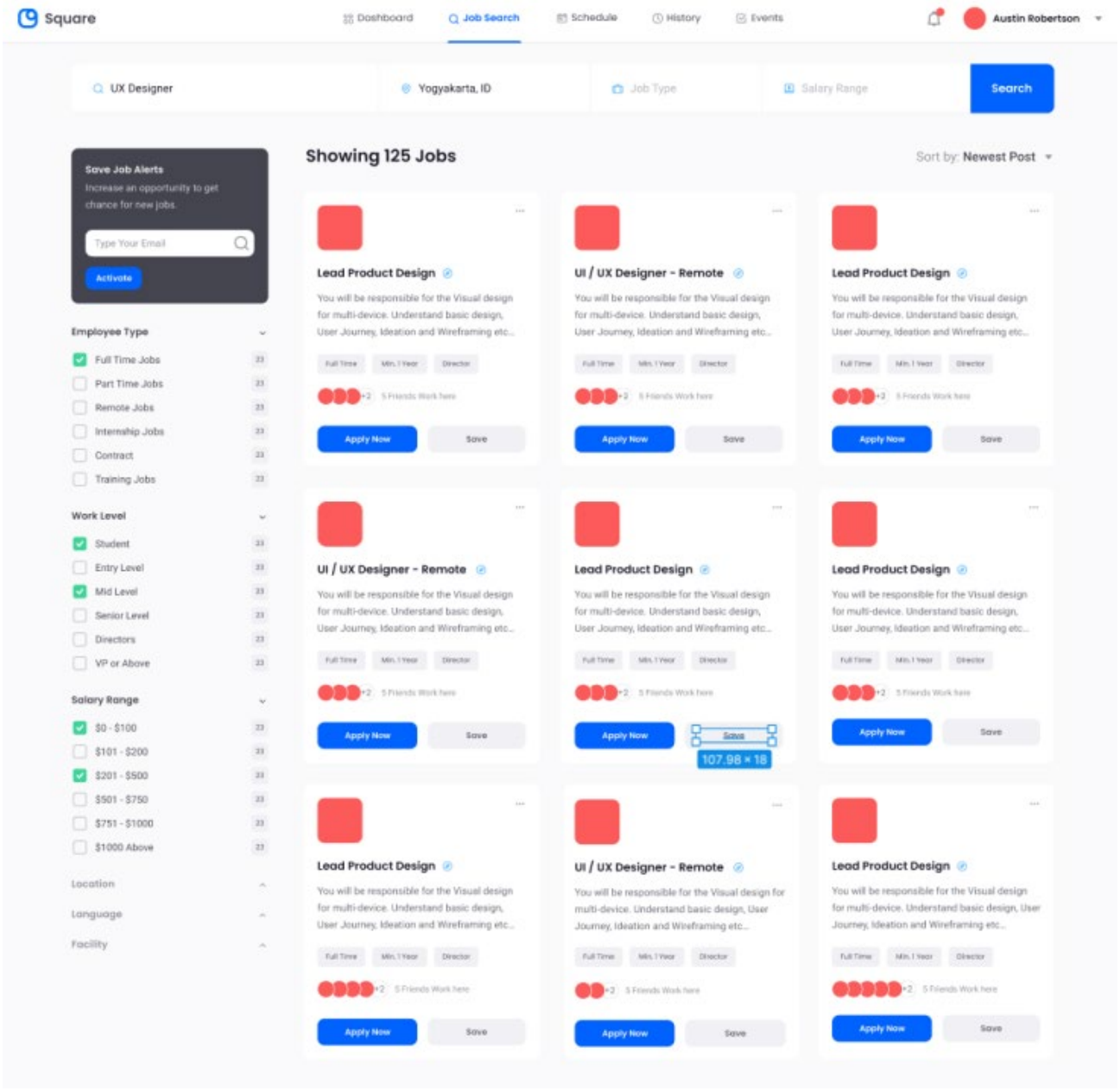

Рисунок 4.12 – Сторінка пошуку вакансій

На рисунку 4.12 зображено список доступних вакансій з їх короткою інформацією. У списку зазначені назви вакансій, їх короткий опис, категорії до яких вони відносяться, а також зображені кнопки для зберігання вакансії та швидкого подання заявки. На кожній картці вакансії можна подати на неї заявку або зберегти. З лівої сторони зображений сайдбар з фільтрами для вакансій, а також карточка з можливістю підписки на вакансії за електронною поштою. Зверху зображений тулбар з пошуком та додатковими фільтрами, а також основний хедер сайту.

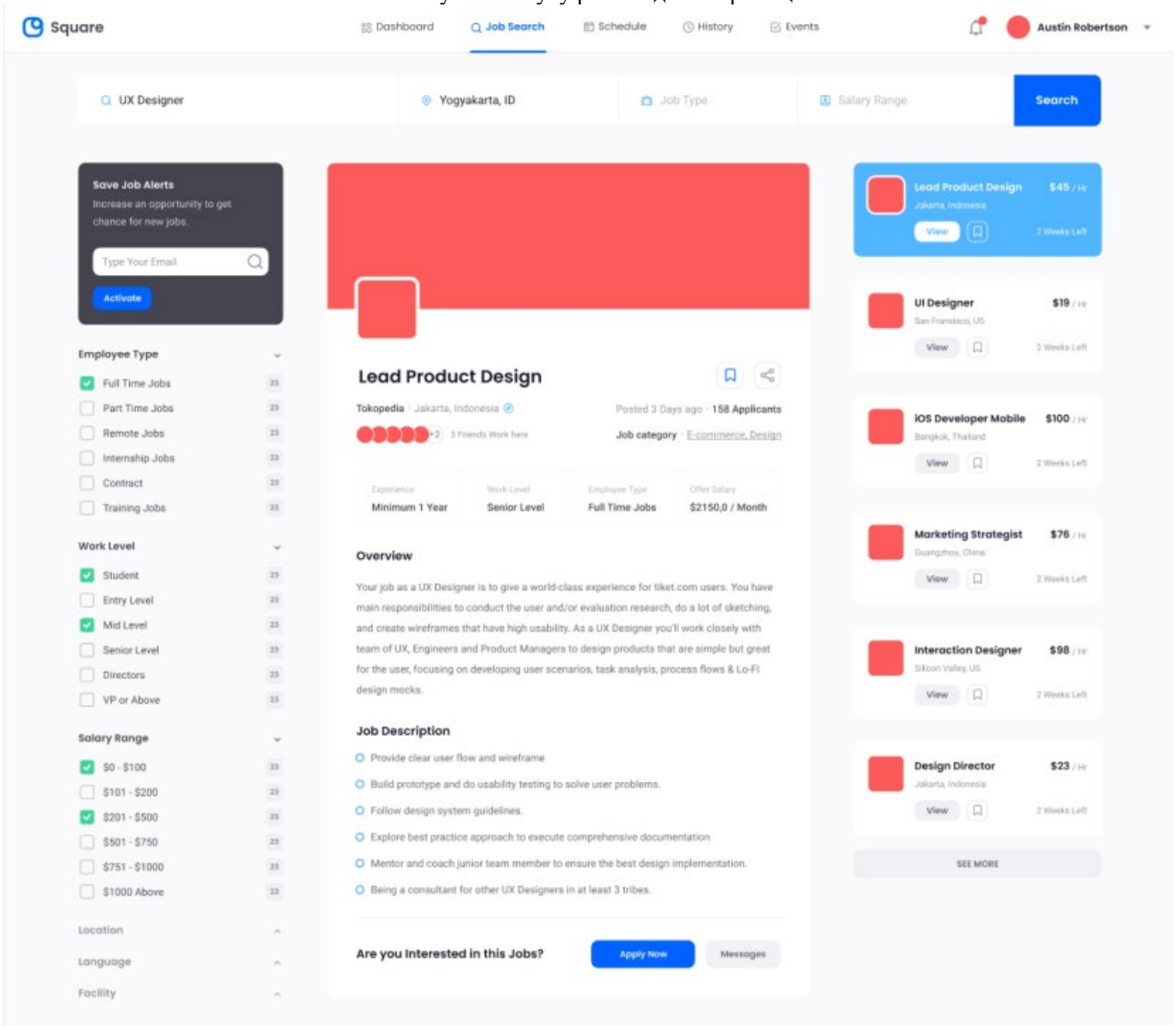

Рисунок 4.13 – Сторінка перегляду вакансії

На рисунку 4.13 зображено детальну інформацію про вакансію. У детальній інформаціїї про вакансію зазначено назву вакансії, її опис, категорії, до яких вона відноситься, скільки кандидатів подало заяку на цю вакансію. Також можна побачити кнопку для подання заявки на вакансію. Зліва зображений сайдбар з обраними фільтрами (за типом вакансії, рівнем заробітньої плати, рівнем кандидата), а також карточка з можливістю підписки на вакансії за електронною поштою. Справа відображені вакансії, які є схожими на поточну. Зверху зображений тулбар з пошуком та додатковими фільтрами, а також основний хедер сайту.

4.4 Алгоритми фільтрації та пошуку

Фільтрація та пошук вакансій є одним з найважливіших аспектів вебзастосунку. Алгоритми пошуку та фільтрації вакансій (рис. 4.14) у вебзастосунку для пошуку роботи інженерів з IT включають у себе аналіз запитів користувачів, порівняння критеріїв пошуку та профілю кандидата з даними про вакансії у базі даних, а також використання різних метрик для відбору найбільш відповідних вакансій.

```
const express = require( id: 'express');
const { Job } = require( id: './models');

const app = express();

// Маршрут для отримання вакансій за певними параметрами
new *
app.get('/jobs', async (req, res) => {
  try {
    const { location, skills } = req.query;
    const jobs = await Job.find({ location, skills: { $in: skills } });
    res.json(jobs);
  } catch (err) {
    console.error(err);
    res.status( code: 500).json( body: { message: "Internal Server Error" });
  }
});

new *
app.listen( port: 3000,  callback: () => {
  console.log("Server is running on port 3000");
});
```

Рисунок 4.14 – Лістинг фільтрації та пошуку вакансій

Наступним кроком є логіка алгоритму для підбору вакансій для кандидата (рис. 4.15). Алгоритм підбору вакансій для кандидата ґрунтується на порівнянні навичок, досвіду, освіти та розташування кандидата з вимогами вакансій.

```
// Маршрут для підбору вакансій
new *
app.get('/match-vacancies/:candidateId', async (req, res) => {
  const candidateId = req.params.candidateId;
  const candidate = await Candidate.findById(candidateId);

  const vacancies = await Vacancy.find();

  const matchedVacancies : any[]  = [];
  for (const vacancy of vacancies) {
    let matchScore : number  = 0;

    if (vacancy.skills.some(skill => candidate.skills.includes(skill))) {
      matchScore++;
    }

    if (vacancy.experience <= candidate.experience) {
      matchScore++;
    }

    if (vacancy.location === candidate.location) {
      matchScore++;
    }

    if (matchScore >= 2) {
      matchedVacancies.push({
        vacancyId: vacancy._id,
        matchScore: matchScore
      });
    }
  }
  matchedVacancies.sort( compareFn: (a, b) => b.matchScore - a.matchScore);
  res.json(matchedVacancies);
});
```

Рисунок 4.15 – Лістинг логіки для фільтрації вакансій для кандидата

Спершу ідентифікатор кандидата витягується з параметрів URL. На основі цього ідентифікатора з бази даних витягується інформація про кандидата за допомогою методу **Candidate.findById(candidateId)**. Потім отримуються всі доступні вакансії за допомогою методу **Vacancy.find()**.

Створюється порожній масив **matchedVacancies**, який буде використовуватися для зберігання вакансій, що відповідають профілю кандидата.

Далі алгоритм проходить по кожній вакансії в масиві **vacancies**. Для кожної вакансії ініціалізується змінна **matchScore**, яка використовується для оцінки відповідності вакансії профілю кандидата.

Перевірка відповідності проводиться за трьома критеріями:

1) **відповідність навичок**: алгоритм перевіряє, чи відповідає якась з навичок, що вимагаються вакансією, навичкам, які має кандидат. Якщо є відповідність, **matchScore** збільшується на 1;

2) **відповідність досвіду**: алгоритм перевіряє, чи досвід кандидата не менший за досвід, необхідний для вакансії. Якщо це так, **matchScore** збільшується на 1;

3) **відповідність місця розташування**: алгоритм перевіряє, чи відповідає місце розташування вакансії місцю розташування кандидата. Якщо це так, **matchScore** збільшується на 1.

Якщо **matchScore** для вакансії дорівнює або перевищує 2 (що вказує на добру відповідність), ця вакансія додається до масиву **matchedVacancies**. Кожен елемент цього масиву включає в себе ідентифікатор вакансії та її оцінку відповідності (**matchScore**).

Наприкінці, масив **matchedVacancies** сортується у спадному порядку за **matchScore**, щоб на першому місці були найкращі відповідності. Після цього відсортований масив відправляється як JSON-відповідь.

Цей алгоритм призначений для знаходження вакансій, які найбільше відповідають профілю кандидата, використовуючи три основні критерії відповідності. Він оцінює кожну вакансію, фільтрує невідповідні варіанти та повертає найбільш відповідні вакансії кандидатам.

4.5 Тестування застосунку

Написання юніт-тестів для Angular-застосунків має величезне значення з кількох причин:

1) **впевненість у працездатності коду**: юніт-тести дозволяють перевірити, чи працюють окремі компоненти програми так, як очікується. Це забезпечує впевненість у функціональності кожного елемента вашого застосунку.

2) **зменшення кількості помилок**: написання тестів допомагає виявляти потенційні проблеми та помилки в коді на ранніх етапах розробки, що дозволяє уникнути проблем у майбутньому та зменшити кількість витрат на виправлення помилок.

3) **підтримка масштабованості**: юніт-тести роблять код більш масштабованим. Під час рефакторингу або додавання нових функцій ви можете бути впевнені, що існуючий функціонал залишиться незмінним, оскільки він покритий тестами.

4) **покращення архітектури застосунку**: під час написання тестів ви зазвичай дотримуєтеся кращих практик програмування та розробки, що сприяє покращенню архітектури вашого застосунку.

У вебзастосунку пошуку роботи на Angular юніт-тестами повинні бути покриті різні компоненти, сервіси, директиви, пайпи.

Для компонентів важливо перевірити створення компонента без помилок, коректність роботи методів, які виконують логіку компонента, правильність обробки подій (наприклад, кліків кнопок, змін інпутів), відображення даних у шаблоні (HTML) та коректність роботи двобічного зв'язування даних (data binding).

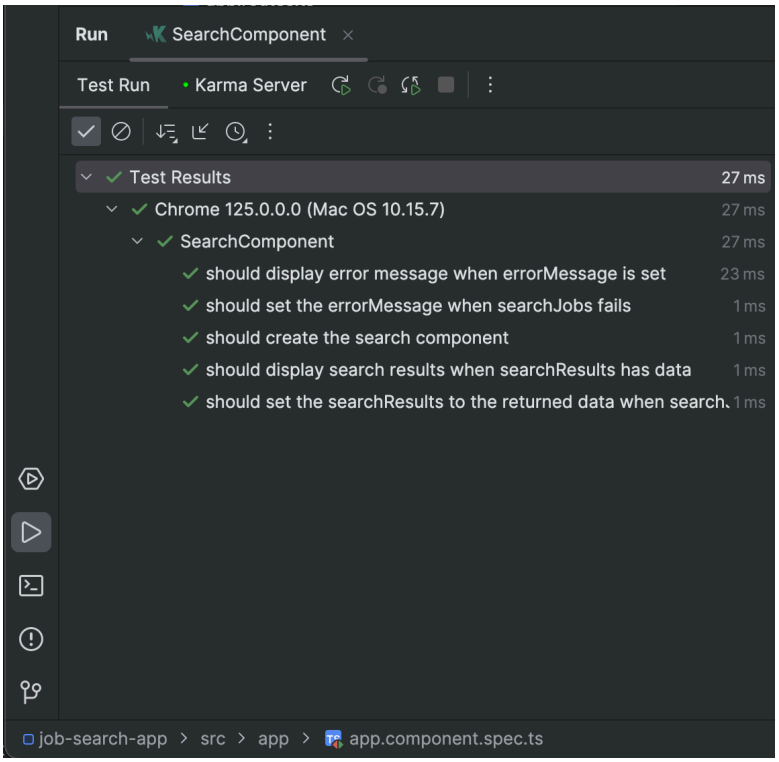

Рисунок 4.16 – Результат виконання юніт-тестів для SearchComponent

Сервіси слід тестувати на коректність роботи методів, зокрема логіки, пов'язаної з API-запитами, та використовувати **HttpTestingController** для перевірки коректності HTTP-запитів і обробки відповідей.

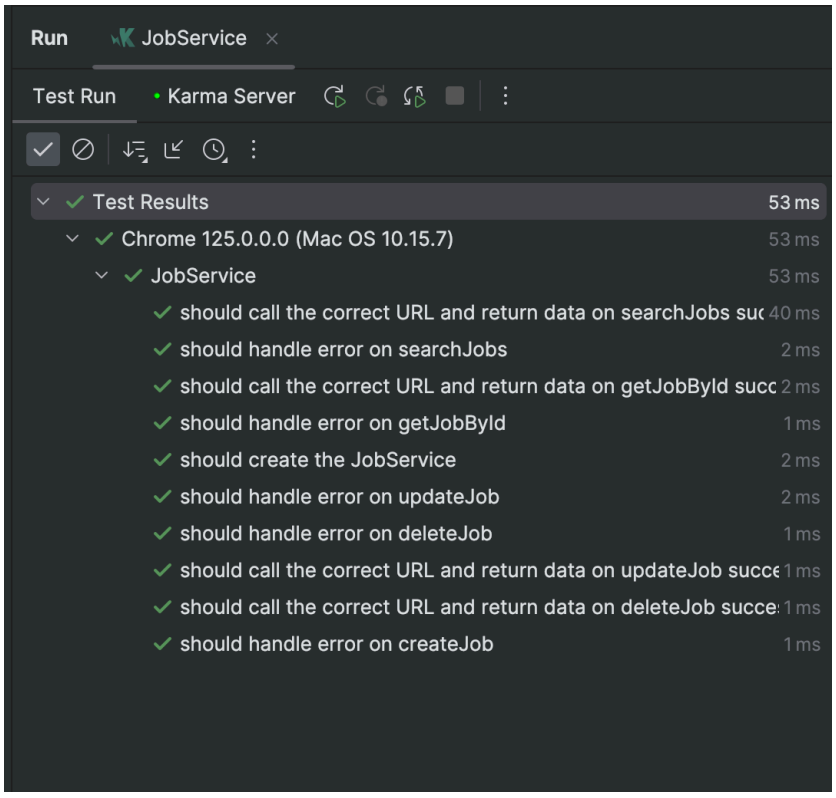

Рисунок 4.17 – Результат виконання юніт-тестів для JobService

Директиви повинні бути перевірені на предмет їхньої функціональності, щоб вони працювали належним чином, змінюючи DOM або поведінку компонентів відповідно до логіки.

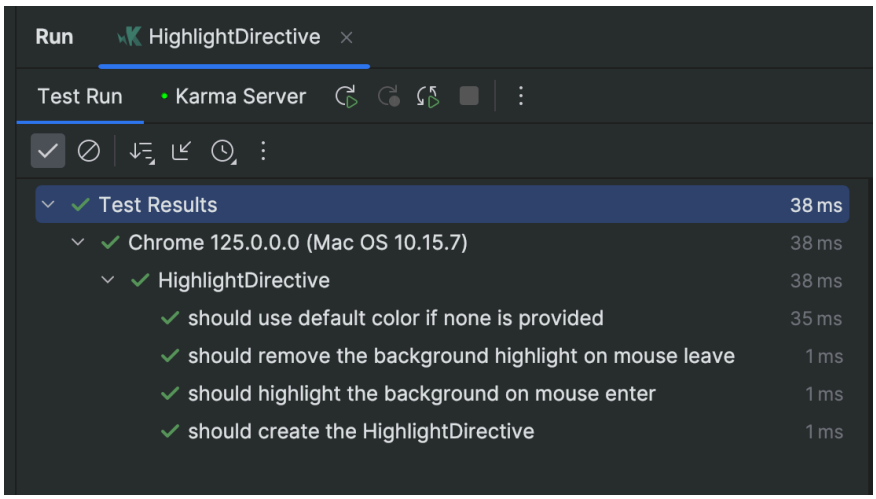

Рисунок 4.18 – Результат виконання юніт-тестів для HighlightDirective

Для пайпів необхідно перевірити, чи правильно вони трансформують вхідні дані до очікуваного виходу.

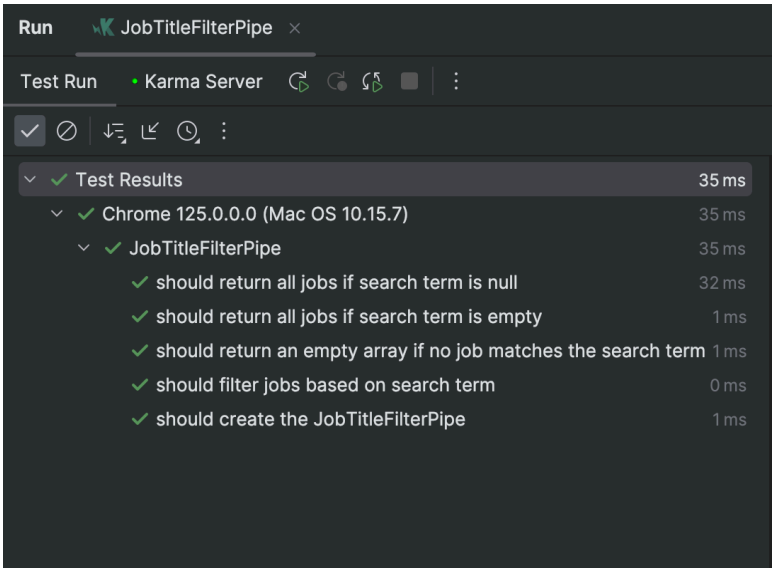

Рисунок 4.19 – Результат виконання юніт-тестів для JobTitleFilterPipe

Ретельне покриття тестами цих аспектів допоможе забезпечити стабільність і надійність вашого вебзастосунку для пошуку роботи.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи було детально описано структуру вебзастосунку, було описано призначення основних папок та визначено їх вміст. Було описано налаштування та наповнення колекцій бази даних, зазначено лістинг підключення до бази даних з клієнтського застосунку. Було зазначено схему основних компонентів UI та детально описано кожну сторінку вебзастосунку. Були зазначені основні критерії та результати тестування.

ВИСНОВКИ

В рамках кваліфікаційної роботи було проведено дослідження предметної галузі, зокрема ринку ІТ-вакансій в Україні. Основна увага була звернена до аналізу поточної ситуації, тенденцій та викликів, а також вивченню основних платформ для пошуку роботи, таких як DOU, Djinni та Linkedin.

Проєктування інтерфейсу користувача здійснювалося з використанням фреймворку Angular, що забезпечило створення зручного та інтуїтивно зрозумілого інтерфейсу з адаптивним дизайном для різних пристроїв.

Для зберігання інформації про вакансії та користувачів була використана база даних MongoDB. Це рішення забезпечило гнучкість та масштабованість системи, що є важливим при роботі з великими обсягами даних.

Імплементація алгоритмів пошуку та фільтрації вакансій здійснювалася за допомогою Express.js. Це дозволило створити ефективні алгоритми, що забезпечують швидкий та точний пошук вакансій за різними критеріями, а також оптимізувати процеси обробки запитів.

Для забезпечення надійності та стабільності вебзастосунку було проведено юніт-тестування. Це дозволило виявити та виправити помилки на ранніх етапах розробки, що значно підвищило якість кінцевого продукту.

Загалом, результати кваліфікаційної роботи свідчать про успішне виконання поставлених завдань та досягнення цілей, що дозволяє розглядати створений вебзастосунок як ефективний інструмент для пошуку ІТ-вакансій в Україні. Результатом реалізації проєкту «Вебзастосунок пошуку роботи для ІТ-фахівців» є інноваційний та корисний інструмент, що сприятиме розвитку ІТ-галузі, сприяючи зростанню якості міжособистих взаємодій та підвищенню продуктивності процесу працевлаштування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Огляд IT-ринку праці, квітень 2023: найнижча за три роки кількість вакансій на DOU і рекордна конкуренція серед кандидатів. URL: https://dou.ua/lenta/articles/job-survey-results-2023/ (Last accessed: 30.05.2023).

2. Як почати працювати в IT без досвіду: покроковий план від айті-компанії. URL: https://lemon.school/blog/yak-pochaty-praczyuvaty-v-it-bez-dosvidu-pokrokovyj-plan-vid-ajti-kompaniyi (Last accessed: 30.05.2023).

3. Як IT-спеціалісти шукають роботу у 2023-му. Для 66% це важко – результати опитування. URL: https://dou.ua/lenta/articles/job-survey-results-2023/ (Last accessed: 30.05.2023).

4. Скільки потрібно шукати роботу в IT. URL: https://forbes.ua/innovations/ne-krashchiy-chas-shchob-uviyti-v-it-za-rik-viyni-tisyachi-ukraintsiv-perevchilisya-na-aytivtsiv-chomu-voni-opinilisya-posered-idealnogo-shtormu-na-rinku-10042023-12961 (дата звернення: 27.04.2023).

5. У пошуках роботи: добірка ресурсів з IT-вакансіями. URL: https://dan-it.com.ua/uk/blog/u-poshukah-roboti-dobirka-resursiv-z-it-vakansijami-dlja-ukrainciv (дата звернення: 27.04.2023).

6. DOU. URL: https://jobs.dou.ua/ (дата звернення: 25.04.2024).

7. Djinny. URL: https://djinni.co/jobs/ (дата звернення: 25.04.2024).

8. Linkedin. URL: https://www.linkedin.com (дата звернення: 25.04.2024).

9. B. Unhelkar, Software Engineering with UML. Auerbach Publications, 2017. 426 с.

10. UML Use Case Diagram Tutorial | Lucidchart. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (Last accessed: 26.04.2023).

11. Use Case Diagrams | Unified Modeling Language (UML). URL: https://www.geeksforgeeks.org/use-case-diagram (Last accessed: 28.04.2023).

12. Online Job Portal Dataflow Diagram. URL: https://www.freeprojectz.com/dfd/online-job-portal-dataflow-diagram (Last accessed: 03.05.2023).

13. Angular. URL: https://angular.dev/ (Last accessed: 29.05.2023).

14. Angular CLI. URL: https://www.npmjs.com/package/@angular/cli/v/13.0.0 (Last accessed: 29.05.2023).

15. Express. URL: https://expressjs.com/ (Last accessed: 29.05.2023).

16. MongoDB. URL: https://www.mongodb.com/ (Last accessed: 29.05.2023).

ДОДАТОК А

Лістинг юніт-тестів

```
// highlight.directive.spec.ts
import { HighlightDirective } from './highlight.directive';
import { Component } from '@angular/core';
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { By } from '@angular/platform-browser';

@Component({
  template: `<p appHighlight="cyan">Highlight me!</p>`
})
class TestComponent {}

describe('HighlightDirective', () => {
  let fixture: ComponentFixture<TestComponent>;
  let p: HTMLElement;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [HighlightDirective, TestComponent]
    });
    fixture = TestBed.createComponent(TestComponent);
    fixture.detectChanges();
    p = fixture.debugElement.query(By.css('p')).nativeElement;
  });

  it('should create an instance', () => {
    const directive = new HighlightDirective(fixture.debugElement.query(By.css('p')));
    expect(directive).toBeTruthy();
  });

  it('should highlight the background on mouse enter', () => {
    const event = new Event('mouseenter');
    p.dispatchEvent(event);
    fixture.detectChanges();
    expect(p.style.backgroundColor).toBe('cyan');
  });

  it('should remove the background highlight on mouse leave', () => {
```

```
    const enterEvent = new Event('mouseenter');
    const leaveEvent = new Event('mouseleave');

    p.dispatchEvent(enterEvent);
    fixture.detectChanges();
    expect(p.style.backgroundColor).toBe('cyan');

    p.dispatchEvent(leaveEvent);
    fixture.detectChanges();
    expect(p.style.backgroundColor).toBe('');
  });

  it('should use default color if none is provided', () => {
    fixture.debugElement.query(By.css('p')).nativeElement.setAttribute('appHighlight', '');
    const event = new Event('mouseenter');
    p.dispatchEvent(event);
    fixture.detectChanges();
    expect(p.style.backgroundColor).toBe('yellow');
  });
});

// job-title-filter.pipe.spec.ts
import { JobTitleFilterPipe } from './job-title-filter.pipe';

describe('JobTitleFilterPipe', () => {
  let pipe: JobTitleFilterPipe;

  beforeEach(() => {
    pipe = new JobTitleFilterPipe();
  });

  it('should create an instance', () => {
    expect(pipe).toBeTruthy();
  });

  it('should filter jobs based on search term', () => {
    const jobs = [
      { id: 1, title: 'Software Developer' },
      { id: 2, title: 'Web Designer' },
      { id: 3, title: 'Project Manager' }
    ];
```

```
  const searchTerm = 'developer';
  const filteredJobs = pipe.transform(jobs, searchTerm);

  expect(filteredJobs.length).toBe(1);
  expect(filteredJobs[0].title).toBe('Software Developer');
});

it('should return all jobs if search term is empty', () => {
  const jobs = [
    { id: 1, title: 'Software Developer' },
    { id: 2, title: 'Web Designer' },
    { id: 3, title: 'Project Manager' }
  ];
  const searchTerm = '';
  const filteredJobs = pipe.transform(jobs, searchTerm);

  expect(filteredJobs.length).toBe(3);
});

it('should return all jobs if search term is null', () => {
  const jobs = [
    { id: 1, title: 'Software Developer' },
    { id: 2, title: 'Web Designer' },
    { id: 3, title: 'Project Manager' }
  ];
  const filteredJobs = pipe.transform(jobs, null);

  expect(filteredJobs.length).toBe(3);
});

it('should be case insensitive', () => {
  const jobs = [
    { id: 1, title: 'Software Developer' },
    { id: 2, title: 'Web Designer' },
    { id: 3, title: 'Project Manager' }
  ];
  const searchTerm = 'DEVELOPER';
  const filteredJobs = pipe.transform(jobs, searchTerm);

  expect(filteredJobs.length).toBe(1);
  expect(filteredJobs[0].title).toBe('Software Developer');
```

```
  });

  it('should return an empty array if no job matches the search term', () => {
    const jobs = [
      { id: 1, title: 'Software Developer' },
      { id: 2, title: 'Web Designer' },
      { id: 3, title: 'Project Manager' }
    ];
    const searchTerm = 'doctor';
    const filteredJobs = pipe.transform(jobs, searchTerm);

    expect(filteredJobs.length).toBe(0);
  });
});

// job.service.spec.ts
import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
import { JobService } from './job.service';

describe('JobService', () => {
  let service: JobService;
  let httpMock: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [JobService]
    });
    service = TestBed.inject(JobService);
    httpMock = TestBed.inject(HttpTestingController);
  });

  afterEach(() => {
    httpMock.verify();
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
```

```
it('should call the correct URL and return data on searchJobs success', () => {
  const query = 'developer';
  const mockJobs = [{ id: 1, title: 'Software Developer' }, { id: 2, title: 'Web Designer' }];

  service.searchJobs(query).subscribe((jobs) => {
    expect(jobs).toEqual(mockJobs);
  });

  const req = httpMock.expectOne(`${service['apiUrl']}?q=${query}`);
  expect(req.request.method).toBe('GET');
  req.flush(mockJobs);
});

it('should handle error on searchJobs', () => {
  const query = 'developer';
  const errorMessage = 'Something went wrong';

  service.searchJobs(query).subscribe(
    () => fail('should have failed with the error'),
    (error) => {
      expect(error).toBe(errorMessage);
    }
  );

  const req = httpMock.expectOne(`${service['apiUrl']}?q=${query}`);
  req.flush(errorMessage, { status: 500, statusText: 'Server Error' });
});

it('should call the correct URL and return data on getJobById success', () => {
  const jobId = '123';
  const mockJob = { id: jobId, title: 'Software Developer' };

  service.getJobById(jobId).subscribe((job) => {
    expect(job).toEqual(mockJob);
  });

  const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
  expect(req.request.method).toBe('GET');
  req.flush(mockJob);
});
```

```
  it('should handle error on getJobById', () => {
    const jobId = '123';
    const errorMessage = 'Something went wrong';

    service.getJobById(jobId).subscribe(
      () => fail('should have failed with the error'),
      (error) => {
        expect(error).toBe(errorMessage);
      }
    );

    const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
    req.flush(errorMessage, { status: 500, statusText: 'Server Error' });
  });

  it('should call the correct URL and return data on createJob success', () => {
    const newJob = { title: 'Software Developer' };
    const createdJob = { id: '123', ...newJob };

    service.createJob(newJob).subscribe((job) => {
      expect(job).toEqual(createdJob);
    });

    const req = httpMock.expectOne(service['apiUrl']);
    expect(req.request.method).toBe('POST');
    expect(req.request.body).toEqual(newJob);
    req.flush(createdJob);
  });

  it('should handle error on createJob', () => {
    const newJob = { title: 'Software Developer' };
    const errorMessage = 'Something went wrong';

    service.createJob(newJob).subscribe(
      () => fail('should have failed with the error'),
      (error) => {
        expect(error).toBe(errorMessage);
      }
    );

    const req = httpMock.expectOne(service['apiUrl']);
```

```
  req.flush(errorMessage, { status: 500, statusText: 'Server Error' });
});

it('should call the correct URL and return data on updateJob success', () => {
  const jobId = '123';
  const updatedJob = { title: 'Senior Software Developer' };

  service.updateJob(jobId, updatedJob).subscribe((job) => {
    expect(job).toEqual(updatedJob);
  });

  const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
  expect(req.request.method).toBe('PUT');
  expect(req.request.body).toEqual(updatedJob);
  req.flush(updatedJob);
});

it('should handle error on updateJob', () => {
  const jobId = '123';
  const updatedJob = { title: 'Senior Software Developer' };
  const errorMessage = 'Something went wrong';

  service.updateJob(jobId, updatedJob).subscribe(
    () => fail('should have failed with the error'),
    (error) => {
      expect(error).toBe(errorMessage);
    }
  );

  const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
  req.flush(errorMessage, { status: 500, statusText: 'Server Error' });
});

it('should call the correct URL and return data on deleteJob success', () => {
  const jobId = '123';

  service.deleteJob(jobId).subscribe((response) => {
    expect(response).toEqual({});
  });

  const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
```

```
    expect(req.request.method).toBe('DELETE');
    req.flush({});
  });

  it('should handle error on deleteJob', () => {
    const jobId = '123';
    const errorMessage = 'Something went wrong';

    service.deleteJob(jobId).subscribe(
      () => fail('should have failed with the error'),
      (error) => {
        expect(error).toBe(errorMessage);
      }
    );

    const req = httpMock.expectOne(`${service['apiUrl']}/${jobId}`);
    req.flush(errorMessage, { status: 500, statusText: 'Server Error' });
  });
});
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { SearchComponent } from './search.component';
import { JobService } from './job.service';
import { of, throwError } from 'rxjs';

describe('SearchComponent', () => {
  let component: SearchComponent;
  let fixture: ComponentFixture<SearchComponent>;
  let jobService: jasmine.SpyObj<JobService>;

  beforeEach(async () => {
    const jobServiceSpy = jasmine.createSpyObj('JobService', ['searchJobs']);

    await TestBed.configureTestingModule({
      declarations: [SearchComponent],
      providers: [{ provide: JobService, useValue: jobServiceSpy }]
    }).compileComponents();

    jobService = TestBed.inject(JobService) as jasmine.SpyObj<JobService>;
    fixture = TestBed.createComponent(SearchComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
```

```
});

it('should create the search component', () => {
  expect(component).toBeTruthy();
});

it('should call jobService.searchJobs() with the provided query when search() is called', () => {
  const query = 'developer';
  component.query = query;

  jobService.searchJobs.and.returnValue(of([]));
  component.search();

  expect(jobService.searchJobs).toHaveBeenCalledWith(query);
});

it('should set the searchResults to the returned data when searchJobs is successful', () => {
  const jobs = [{ id: 1, title: 'Software Developer' }, { id: 2, title: 'Web Designer' }];
  jobService.searchJobs.and.returnValue(of(jobs));

  component.query = 'developer';
  component.search();

  expect(component.searchResults).toEqual(jobs);
  expect(component.errorMessage).toBeUndefined();
});

it('should set the errorMessage when searchJobs fails', () => {
  const error = 'Internal Server Error';
  jobService.searchJobs.and.returnValue(throwError(error));

  component.query = 'developer';
  component.search();

  expect(component.errorMessage).toEqual('An error occurred while searching for jobs.');
  expect(component.searchResults).toBeUndefined();
});

// Template tests
it('should display search results when searchResults has data', () => {
  const jobs = [{ id: 1, title: 'Software Developer' }, { id: 2, title: 'Web Designer' }];
```

```
    component.searchResults = jobs;
    fixture.detectChanges();

    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.job-list').textContent).toContain('Software Developer');
    expect(compiled.querySelector('.job-list').textContent).toContain('Web Designer');
  });

  it('should display error message when errorMessage is set', () => {
    component.errorMessage = 'An error occurred while searching for jobs.';
    fixture.detectChanges();

    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.error-message').textContent).toContain('An error occurred while searching
for jobs.');
  });
});
```

ДОДАТОК Б

Лістинг файлів для імпорту даних

```
[
  {
    "title": "Software Engineer",
    "description": "Develop and maintain software applications.",
    "skills": ["JavaScript", "Node.js", "MongoDB"],
    "experience": "3+ years",
    "location": "New York, NY",
    "salary": 120000
  },
  {
    "title": "Data Scientist",
    "description": "Analyze and interpret complex data.",
    "skills": ["Python", "Machine Learning", "Data Analysis"],
    "experience": "2+ years",
    "location": "San Francisco, CA",
    "salary": 130000
  },
  {
    "title": "Frontend Developer",
    "description": "Design and develop user interfaces for web applications.",
    "skills": ["HTML", "CSS", "JavaScript", "React"],
    "experience": "2+ years",
    "location": "London, UK",
    "salary": 110000
  }
]
[
  {
    "name": "John Doe",
    "skills": ["JavaScript", "React", "Node.js"],
    "experience": "4 years",
    "location": "New York, NY",
    "desiredSalary": 115000
  },
  {
    "name": "Jane Smith",
    "skills": ["Python", "Machine Learning", "TensorFlow"],
    "experience": "3 years",
    "location": "San Francisco, CA",
    "desiredSalary": 125000
  },
  {
    "name": "Alex Johnson",
    "skills": ["Java", "Spring Boot", "Hibernate"],
    "experience": "5 years",
    "location": "Berlin, Germany",
    "desiredSalary": 100000
  }]
```