**Пояснювальна записка  
до дипломного проєкту**

на тему: Веб застосунок для автоматичного підбору вакансій на основі резюме та адаптації резюме за допомогою нейромереж для ІТ-галузі (комплексна тема)

КПІ.ІП-1122.ІП-1124.045440.02.81

Київ – 2025

Зміст

[Вступ 5](#_Toc199180540)

[1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 6](#_Toc199180541)

[1.1 Постановка завдання дипломного проєктування 6](#_Toc199180542)

[1.2 Аналіз предметної області 6](#_Toc199180543)

[1.3 Аналіз існуючих рішень 8](#_Toc199180544)

[1.3.1 Аналіз відомих програмних продуктів 8](#_Toc199180545)

[1.3.2 Аналіз відомих алгоритмічних та технічних рішень 19](#_Toc199180546)

[1.4 Аналіз та моделювання бізнес-процесів 24](#_Toc199180547)

[Висновки до розділу 27](#_Toc199180548)

[2 РОЗРОБЛЕННЯ вимог до програмного забезпечення 28](#_Toc199180549)

[2.1 Варіанти використання програмного забезпечення 28](#_Toc199180550)

[2.2 Розроблення нефункціональних вимог 33](#_Toc199180551)

[2.3 Аналіз системних вимог 33](#_Toc199180552)

[2.4 Аналіз економічних показників програмного забезпечення 34](#_Toc199180553)

[Висновки до розділу 35](#_Toc199180554)

[3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 36](#_Toc199180555)

[3.1 Архітектура програмного забезпечення 36](#_Toc199180556)

[3.2 Архітектурні рішення та обґрунтування вибору засобів розробки 37](#_Toc199180557)

[3.3 Конструювання програмного забезпечення 38](#_Toc199180558)

[3.3.1 Опис структури бази даних 38](#_Toc199180559)

[3.4 Аналіз безпеки даних 40](#_Toc199180560)

[Висновки до розділу 41](#_Toc199180561)

[4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 42](#_Toc199180562)

[4.1 Аналіз якості ПЗ 42](#_Toc199180563)

[4.2 Опис процесів тестування 42](#_Toc199180564)

[4.3 Опис контрольного прикладу 43](#_Toc199180565)

[Висновки до розділу 43](#_Toc199180566)

[5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 44](#_Toc199180567)

[5.1 Розгортання програмного забезпечення 44](#_Toc199180568)

[5.2 Супровід програмного забезпечення 48](#_Toc199180569)

[Висновки до розділу 48](#_Toc199180570)

[ВИСНОВКИ 49](#_Toc199180571)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50](#_Toc199180572)

[ДОДАТКИ 52](#_Toc199180573)

Перелік умовних позначень

|  |  |  |
| --- | --- | --- |
| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| API | – | Application programming interface, прикладний програмний Інтерфейс. |
| IT | – | Інформаційні технології. |
| ML | – | Machine Learning |
| БД | – | База даних. |
|  |  |  |
|  |  |  |

Вступ

Пошук роботи є трудомістким процесом, що потребує значних зусиль від пошукачів. Вони змушені самостійно аналізувати численні пропозиції з різних сайтів пошуку роботи, що збільшує час та зусилля для написання ефективного резюме і пошуку роботи загалом.

Дипломний проєкт присвячено розробці веб-застосунку для пошуку вакансій пошукачами роботи та отримання рекомендацій до адаптації резюме. Метою розробки програмного забезпечення є пришвидшення та автоматизація пошуку релевантних вакансій пошукачами роботи та адаптації під них резюме користувача.

Для виконання мети дипломного проєкту буде проаналізовано альтернативи, що частково виконують поставлене завдання та розроблено програмне забезпечення, що матиме переваги над альтернативами завдяки збору вакансій з різних сайтів пошуку роботи та автоматизації їх підбору для користувача за рахунок використання методів машинного навчання, можливості отримати рекомендації для резюме по знайденим або вказаним користувачем вакансіям, а також автоматизованим нотифікаціям про нові релевантні для користувача вакансії.

# ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

* аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;
* аналіз існуючих програмних продуктів за темою ДП та існуючих алгоритмічних рішень машинного навчання, необхідних для виконання завдання розробки у рамках ДП;
* аналіз та моделювання бізнес-процесів;
* розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;
* постановка завдання на розробку програмного забезпечення ДП;
* розроблення архітектури програмного забезпечення;
* розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;
* конструювання та розроблення програмного забезпечення;
* аналіз безпеки даних програмного забезпечення;
* аналіз якості та тестування програмного забезпечення;
* розгортання та супровід програмного забезпечення;
* створення супроводжувальної документації до розробленого програмного забезпечення, такої як керівництво користувача, програма та методика тестування.

## Аналіз предметної області

Сфера працевлаштування в IT є дуже динамічною та актуальною в рамках українського ІТ ринку. За даними популярного сайту пошуку роботи Djinni у 2024 році кількість кандидатів у пошуку роботи варіювалась між 80 та 90 тисячами осіб [1], при чому в найпопулярніших категоріях середня кількість відкликів на вакансію сягає понад 50. Згідно іншої статистики міжнародних компаній лише 3% надісланих резюме доходять до етапу інтерв’ю [2], тому для кандидатів дуже важливо якнайкраще написати резюме та знайти вакансію яка їм найбільше підходить.

В останні роки великим фактором при підготовці резюме до подання на вакансію стало використання ATS [3], що є системами які дозволяють фільтрувати кандидатів ще до перегляду резюме рекрутерами, і що використовуються близько 70% великих компаній. Через ці системи кандидатам важливо мати якнайбільше ключових слів з вакансії у резюме перед тим як його подавати.

В області рекрутингу виділяють 3 ери диджиталізації рекрутингу [4]. До середини 90-х років 20-го століття вакансії публікувались в газетах, і кандидатам треба було ходити до офісів компаній щоб податись на роботу. Перша ера диджиталізації рекрутингу настала, коли компанії почали завантажувати вакансії на свої веб-сторінки замість того щоб розміщувати їх в газетах, що значно знизило витрати на пошук кандидатів. Друга ера диджиталізації наступила з появою незалежних сайтів, що агрегували вакансії багатьох компаній. Третя ж ера диджиталізації визначається все більш активним використанням технологій штучного інтелекту на всіх етапах рекрутингу, що включають в себе:

* Використання ШІ рекрутерами при формуванні описів вакансій;
* Використання ШІ при пошуку роботи кандидатами, базуючись на їх навичках та інших факторах;
* Використання ШІ при написанні резюме для найбільших шансів отримання роботи;
* Використання ШІ для оцінки кандидатів, базуючись на відео інтерв’ю.

Хоч використання ШІ в таких масштабах і значно полегшило роботу для кандидатів та рекрутерів у багатьох аспектах, це означає що щоб бути конкурентоспроможними на рикну праці, кандидатам потрібно користуватись значною кількістю сервісів для написання резюме, а також значною кількістю агрегаторів вакансій щоб знайти найкращу для себе.

В даній роботі було вирішено поєднати сервіс, який надає рекомендації для резюме з агрегатором вакансій, що також надаватиме та сортуватиме вакансії за релевантністю до резюме. Таким чином, це значно полегшить процес пошуку роботи.

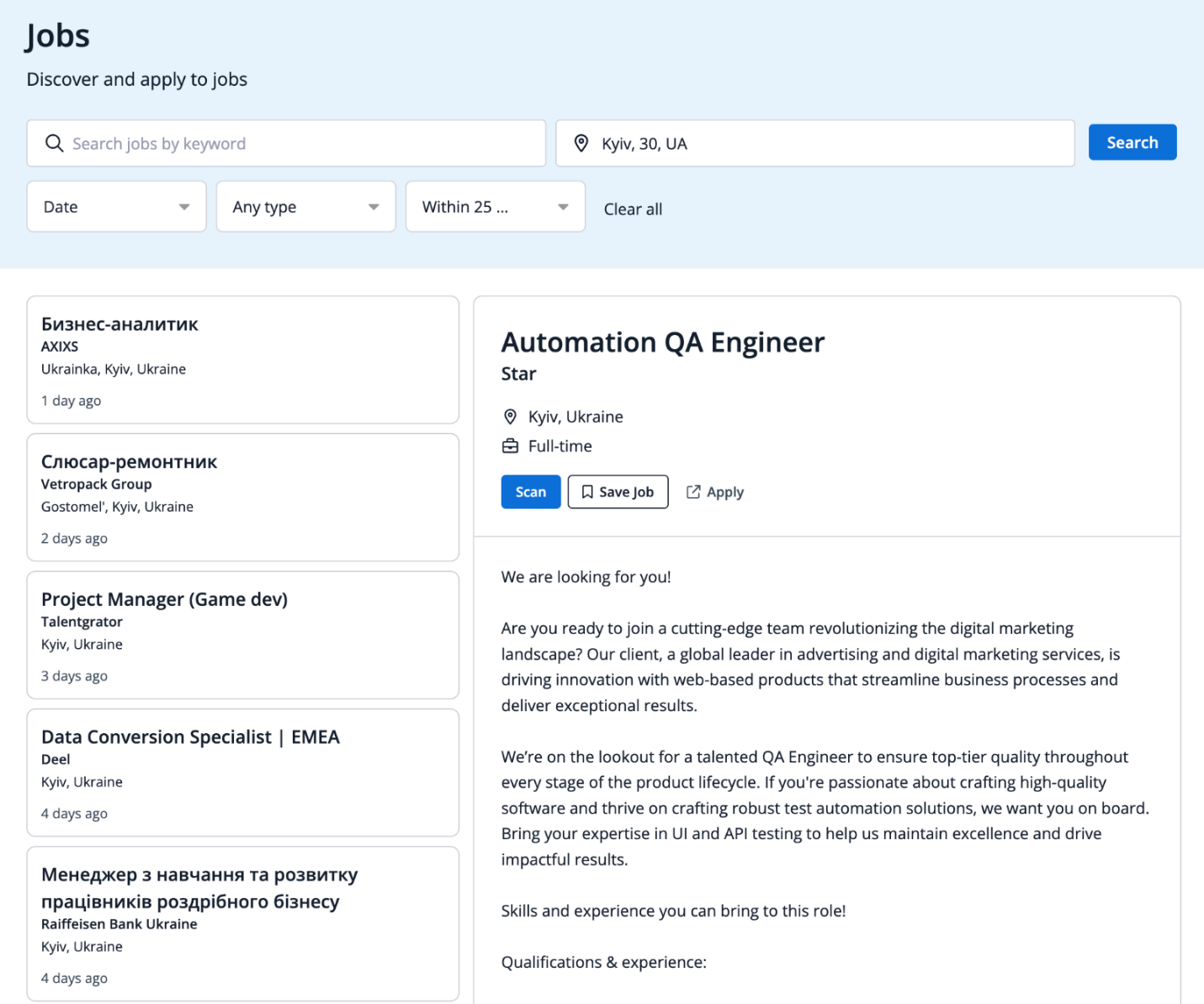
## Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації Kolyba Resume. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

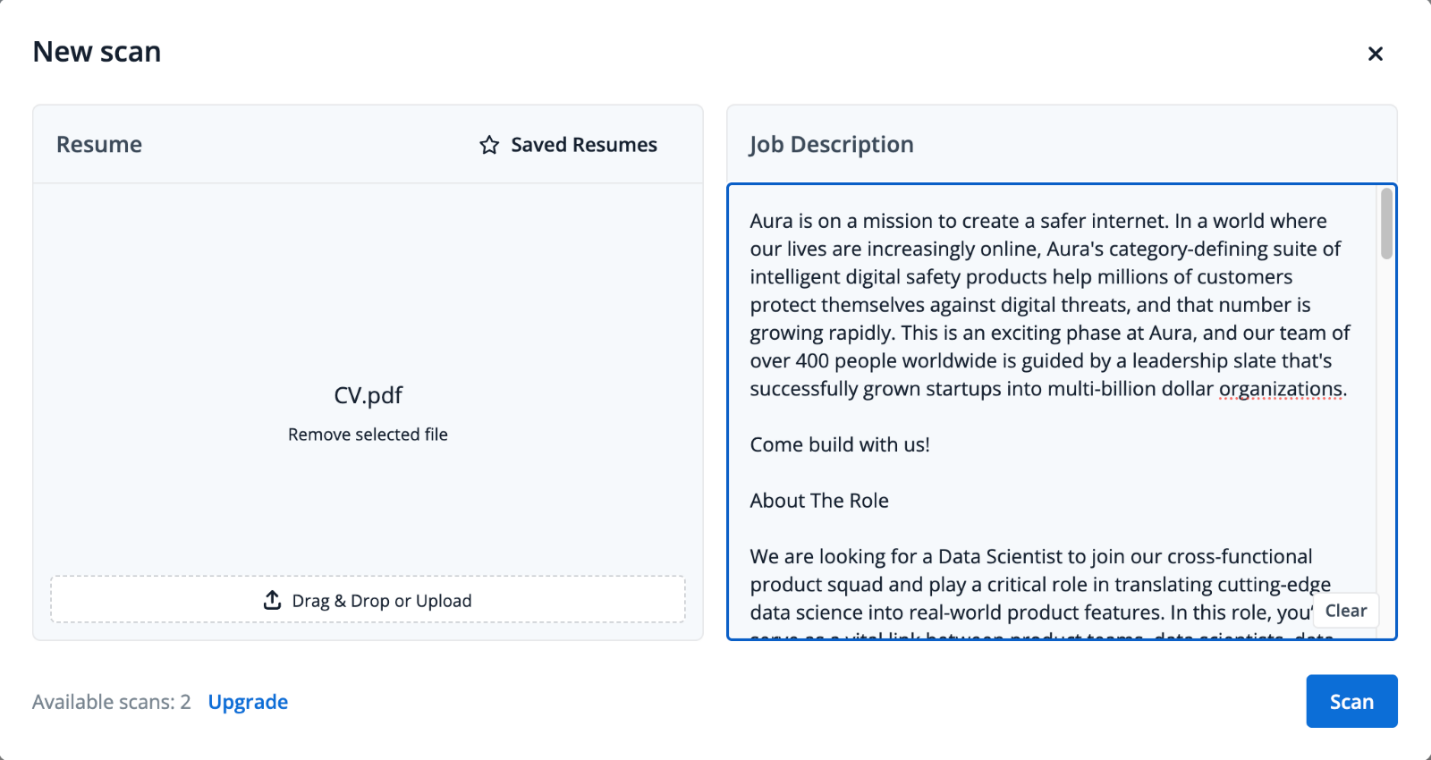
### Аналіз відомих програмних продуктів

Наразі на ринку існує декілька продуктів, які дозволяють підібрати релевантні вакансії по резюме або ж оцінити резюме загалом чи відносно конкретної вакансії та запропонувати шляхи покращення чи автоматично покращити його. Найбільш відомими є Djinni, Jobscan, Enhancv та Resumatch.

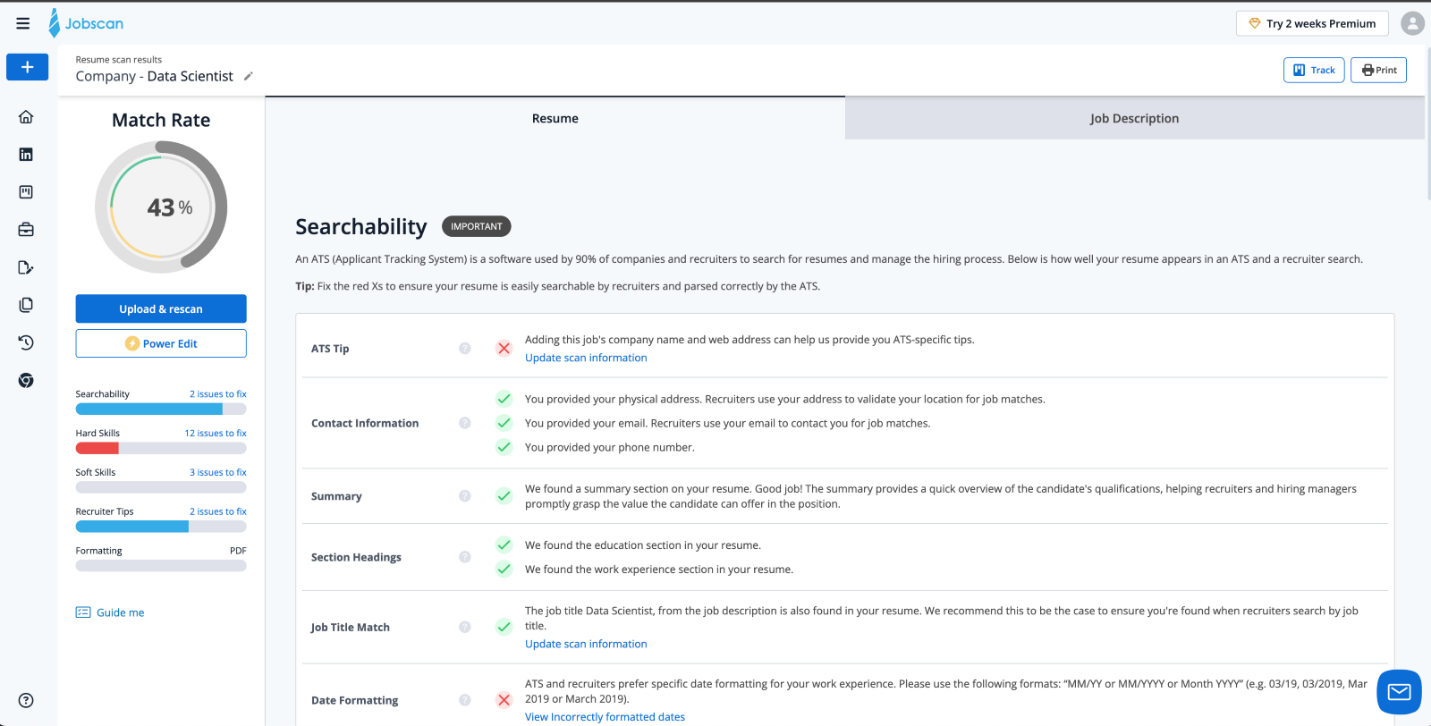
Jobscan [5] – це платформа для пошуку роботи та аналізу резюме, основним функціоналом якого є порівняння резюме з описом вакансії та надання порад по оптимізації резюме під вимоги систем ATS та вакансію. Крім того, він виконує агрегацію вакансій з різних сайтів та дозволяє виконувати пошук як просто по ключовим словам, так і з використанням фільтрів на кшталт локації (рисунок 1.1).

Рисунок 1.1 – Пошук вакансій

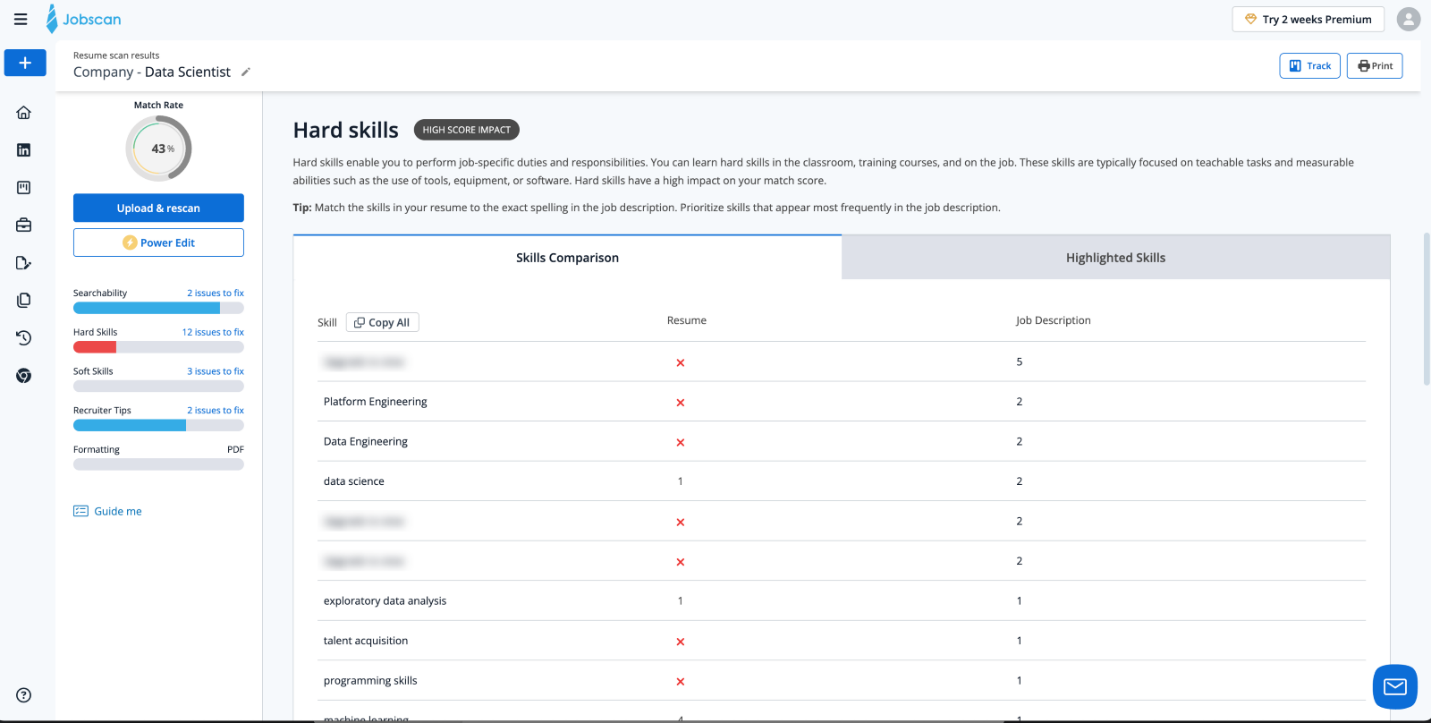
Щодо завантаження резюме та вакансії, то Jobscan може працювати з резюме у текстовому форматі або ж з .pdf чи .docx файлами. Опис вакансії можна надавати лише у текстовому форматі (рисунок 1.2).

Рисунок 1.2 – Завантаження резюме та опису вакансії

 Після виконання аналізу Jobscan надає звіт по відповідності резюме та вакансії, зокрема відсоток відповідності вакансії (match rate) та рекомендації щодо покращення певних частин резюме для оптимізації під ATS та оптимізації під дану вакансію (рисунок 1.3).

Рисунок 1.3 – Звіт по резюме та відповідності вакансії у Jobscsan

Для оцінки відповідності навичок кандидата, зазначених у резюме, вакансії Jobscan перевіряє наявність у резюме відповідних ключових слів, зазначених у вакансії (рисунок 1.4).

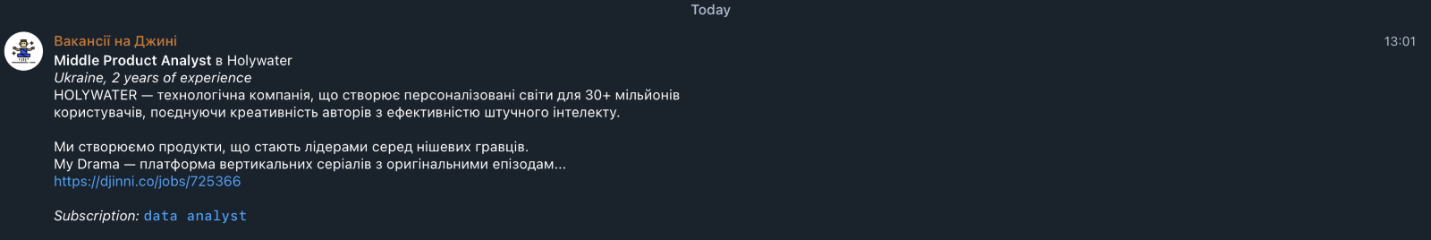
Рисунок 1.4 – Аналіз навичок кандидата на відповідність вакансії

Варто зазначити, що Jobscan не виконує автоматичний підбір релевантних вакансій по резюме та не реалізовує функціонал надсилання повідомлень про нові вакансії відповідно до резюме користувача. Крім того, в рамках безкоштовної версії Jobscan дозволяє робити до 5 аналізів резюме на місяць, а у звіті по резюме доступна не вся інформація без підписки.

Djinni [6] – це українська платформа пошуку роботи, яка орієнтована здебільшого на IT-сектор. На відміну від інших програмних продуктів, розглянутих у цьому розділі, ця платформа надає роботодавцям публікувати вакансії та шукати кандидатів. Таким чином, це спрощує пошук вакансій та комунікацію з компаніями й рекрутерами для кандидатів, проте звужує пошук лише до тих вакансій, що були опубліковані на Djinni. У контексті даної предметної області ця платформа, окрім пошуку вакансій, реалізує автоматичний підбір релевантних вакансій на основі профіля користувача, причому цей список вакансій регулярно оновлюється, якщо з’являються нові вакансії або деякі вакансії стають неактивними (рисунок 1.5).

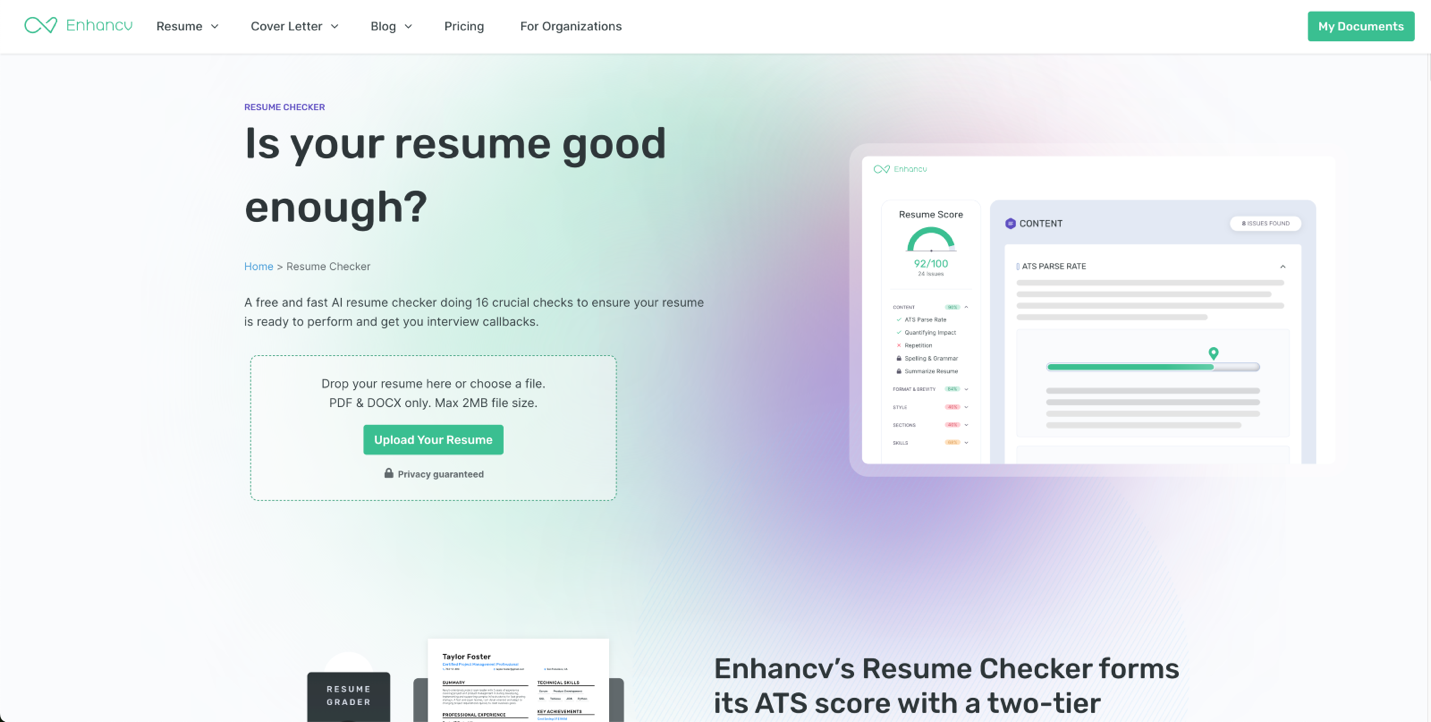
Рисунок 1.5 – Підбір релевантних вакансій за профілем користувача

Крім того, Djinni надає можливість користувачу підписатись на розсилку про появу нових вакансій за обраними напрямками. Розсилка реалізується через бота в telegram (рис. 1.6).

Рисунок 1.6 – Розсилка нових вакансій через бота в telegram

Функціональність для адаптації чи аналізу резюме в Djinni відсутня.

Enhancv [7] – це популярний онлайн-конструктор резюме, який впроваджує штучний інтелект для аналізу резюме та його покращення. На відміну від Jobscan та Djinni, Enhancv не передбачає пошуку чи підбору релевантних вакансій відповідно до резюме. Можливими форматами для завантаження резюме є .pdf або .docx файли (рисунок 1.7), тобто немає можливості користувачу просто вставити текст резюме чи написати його вручну.

Рисунок 1.7 – Інтерфейс Enhancv для завантаження резюме

При звичайному аналізі резюме Enhancv надає звіт по ньому з оцінкою резюме загалом від 0 до 100 та оцінкою кожного критерію, зокрема змісту, формату, наявності усіх необхідних секцій, навичок й стилю. Чим вища загальна оцінка, тим більш резюме є оптимізованим під ATS (рисунок 1.8).

Рисунок 1.8 – Звіт по резюме

Особливістю Enhancv, яка вирізняє його від аналогів, є наявність AI-асистента, якиий може автоматично відредагувати обрані секції резюме (рисунок 1.9).

Рисунок 1.9 – Використання AI-асистента для покращення секції резюме

Також Enhancv дозволяє автоматично адаптувати резюме під конкретну вакансію на основі її опису (рисунки 1.10 та 1.11). Для цього також використовується AI-асистент, який автоматично редагує необхідні секції резюме, щоб підвищити його відповідність даній вакансії.

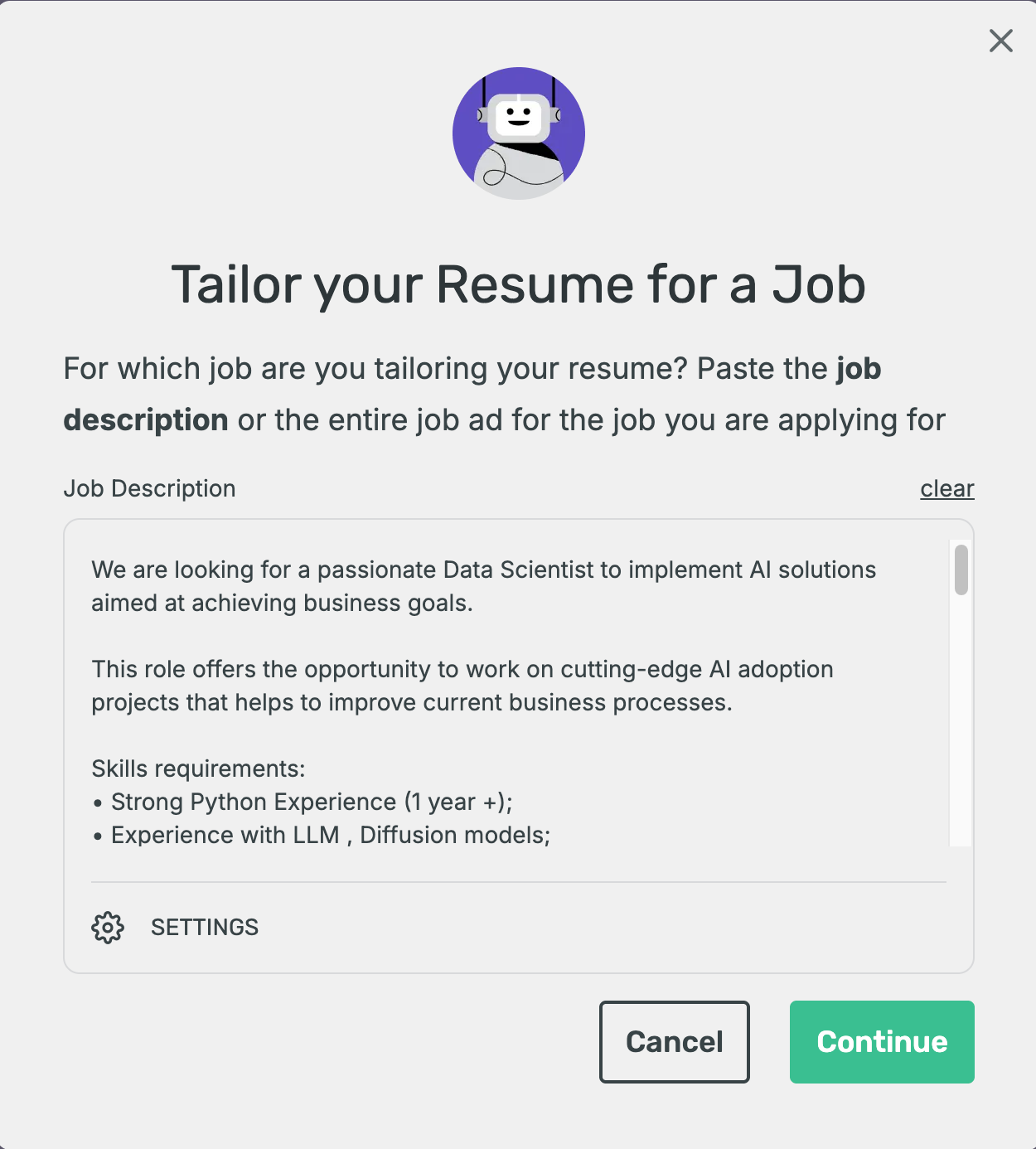
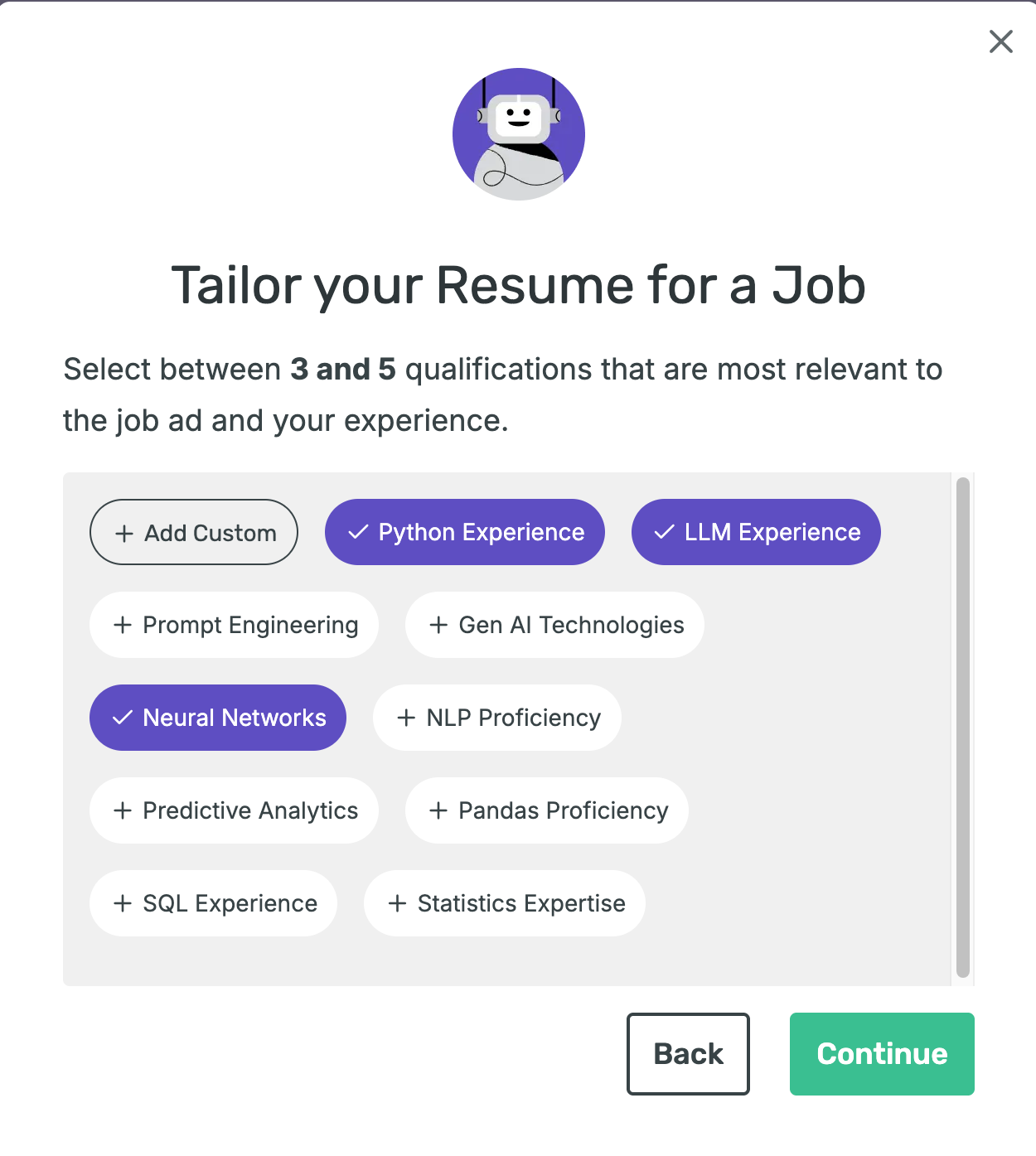


Рисунок 1.10 – Інтерфейс для надання опису вакансії

Рисунок 1.11 – Автоматична адаптація резюме під вакансію

Варто зазначити, що користувачу все одно потрібно після надання опису вакансії обрати від 3 до 5 найбільш релевантних ключових слів відповідно до його досвіду та опису вакансії (рисунок 1.12). Таким чином, адаптація не є повністю автоматизованою.

Рисунок 1.12 – Вибір релевантних ключових слів для адаптації

Для порівняння проєкту з аналогами можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогами

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерій | Kolyba Resume | Jobscan | Djinni | Enhancv | Пояснення |
| Загальні рекомендації резюме | - | - | - | + |  |
| Рекомендації по резюме під конкретну вакансію | + | + | - | + |  |
| Автоматичне редагування резюме | - | + | - | + |  |
| Агрегація вакансій | + | + | - | - |  |
| Підбір релевантних вакансій по резюме | + | - | + | - |  |
| Сповіщення про нові вакансії | + | - | + | - |  |
| Цінова політика | Безкоштовна ліцензія | Часткова платна ліцензія | Безкоштовна ліцензія | Часткова платна ліцензія |  |

### Аналіз відомих алгоритмічних та технічних рішень

Ключовою задачею в рамках даної розробки є підбір релевантних вакансій на основі тексту резюме. Оскільки агреговані вакансії зберігаються в базі даних, то дана задача зводиться до обрахунку певної метрики подібності між резюме та кожною з вакансією, щоб на основі отриманих результатів обрати необхідну кількість найбільш релевантних вакансій. Варто зазначити, що перебір усіх вакансій не є оптимальним як з точки зору часу, так і з точки зору точності отриманих результатів, так і з точки зору точності, так як використання лише метрики подібності не гарантує коректність отриманих результатів. Це пояснюється ефектом хабності [8], коли певна група вакансій можуть мати дуже загальний опис та ймовірно матимуть високе значення метрики подібності із більшістю резюме. Таким чином, для усунення цих проблем буде виконано попередню класифікацію резюме, тобто визначення релевантної сфери діяльності (категорії). Для заданого резюме будуть розглядатись лише вакансії отриманої категорії.

Задача класифікація резюме – це задача багатокласової класифікації текстових документів. Найбільш поширеними алгоритмами для цього є традиційні моделі машинного навчання [9], такі як: наївний баєсівський класифікатор та метод опорних векторів, рекурентні нейромережі, зокрема: LSTM та GRU [10], а також трансформери, а саме BERT [11].

Алгоритм наївного баєсівсього класифікатора спирається на теорему Баєса з припущенням умовної незалежності ознак [12]. Задача класифікації формулюється наступним чином: для кожного класу та вектора ознак необхідно обчислити апостеріорну ймовірність за наступною формулою:

Тут – апріорна ймовірність класу , – умовна ймовірність ознак при заданому класі , а – повна ймовірність спостережуваного вектора ознак. Оскільки є однаковою константою для всіх класів, то для визначення найбільш релевантного класу достатньо максимізувати лише чисельник, що дає наступну формулу:

До переваг даного алгоритму можна віднести простоту реалізації та швидкість навчання, проте він сприрається на припущення незалежності ознак, яке часто порушується в реальних даних, та абсолютно не враховує контекст та порядок слів у тексті, що є важливим для коректної класифікації резюме.

Метод опорних векторів (SVM) – це метод, який шукає гіперплощину, що максимально розділяє об’єкти різних класів із максимальним зазором (margin), мінімізуючи при цьому норму вектора вагових коефіцієнтів. Завдяки цьому даний метод може адекватно виконувати класифікацію навіть при великій кількості ознак [13]. У задачах текстової класифікації текст перетворюється у вектор ознак, де кожне слово відповідає окремій ознаці. Значення ознаки – це кількість появ слів у документі, а саме нормалізована частота, що враховує кількість документів, у яких це слово з’являється. Таким чином, простір ознак має високу розмірність. Крім того, завдяки використанню ядрових функцій SVM може також класифікувати дані, які не є лінійно роздільними.

Недоліками традиційних алгоритмів машинного навчання в контексті текстової класифікації є те, що вони не враховують ні порядок слів у реченні, ні контекст [14].

LSTM та GRU є модифікаціями класичної рекурентної нейронної мережі. На відміну від традиційних моделей машинного навчання, ці нейромережі враховують порядок слів та дозволяють запам’ятовувати довготривалі залежності в послідовності (тексті) завдяки системі ґейтів [10], кожен з яких є вектором від 0 до 1.

GRU (Gated Recurrent Unit) використовує три ґейти: ґейт оновлення, ґейт скидання та вихідний ґейт. Таким чином, обичслення у GRU виконуються за наступними формулами:

Тут – сигмоїда, – вектор ознак для поточного токену (слова), – прихований стан попереднього кроку відповідно, , та – параметри моделі, – поелементне множення. Ґейт оновлення визначає частку нової інформації, яка увійде в стан пам’яті. Ґейт скидання визначає частку інформації, що забути, тобто частку прихованого стану з попереднього кроку.

LSTM (Long Short-Term Memory) додатково зберігає окремий вектор-стан , який керується трьома ґейтами: вхідним ґейтом , забувальним ґейтом та вихідним ґейтом . Формули обчислень у LSTM є наступними:

Вхідний ґейт визначає, визначає частку нової інформації, що потрапить у вектор . Аналогічно до GRU, ґейт забування визначає частку інформації, що забути. Вихідний ґейт визначає частину вектора , яка буде збережена у прихований стан .

Порівнюючи GRU та LSTM, GRU потребує потребує менше обчислень через меншу кільість ґейтів, тоді як LSTM є більш стійкою моделлю від перенавчення та може давати більш точні результати [10].

BERT (Bidirectional Encoder Representation from Transformers) хоч і спирається на класичні ймовірнісні мовні моделі, проте завдяки використанню архітектури багатошарового трансформер-енкодера та двонапрямного контексту усуває їхні обмеження. На відміну від вище описаних моделей, весь текст одночасно подається у трансформер-екнодер, принцип роботи якого спирається на механізм уваги, та формує двонапрямні контекстні представлення, тобто попереднє навчання формує універсальні контекстуальні вектори. [11] В отриманих векторах закодовано лексичні, синтаксичні та семантичні характеристики.

Попереднє навчання оригінальної моделі BERT було виконано на текстовому корпусі, що містить близько 3.3 міліарди слів [15], тому для використання даної моделі під конкретну задачу на кшталт класифікації достатньо використати відносно невеликий розмічений текстовий корпусї та на цьому корпусі вже виконати тонке настроювання (fine-tuning).

Модель BERT-base на вхід приймає послідовність завдовжки до 512 токенів та повертає векторне представлення для цієї послідовності, яка може складатись або з одного, або з двох сегментів [16]. Першим сегментом завжди є спеціальний токен [CLS], який і містить спеціальне класифікаційне представлення. Для задачі класифікації текстів часто кінцевий прихований вектор токена [CLS] трактують як узагальнене представлення даного текстового документа [16]. Таким чином, для fine-tuning моделі для задачі класифікації достатньо додати softmax-шар, що обчислює ймовірність належності даного текстового документу до класу :

Тут – матриця вагових коефіцієнтів (параметрів), яка є специфічною для конкретної задачі класифікації. В процесі fine-tuning оновлюються всі параметри моделі разом з матрицею

Отже, оскільки для задачі класифікації резюме важливо враховувати контекст та порядок слів у реченні, а кількість резюме кандидатів у сфері IT у відритому доступі є сильно обмеженою, то для проєкту було обрано BERT, а саме fine-tuning даної моделі.

Основними двома архітектурними підходами написання веб-застосунків є монолітна та мікросервісна архітектура [17]. У випадку монолітної архітектурі весь функціонал застосунку розробляється в рамках однієї кодової бази і працює як один сервіс, в якому багато модулів можуть бути тісно пов’язані між собою. Мікросервісна ж архітектура виділяється тим, що застосунки що її використовують будуються з багатьох окремих сервісів, що ізолюють певний функціонал всередині себе і комунікують за допомогою мережевих запитів, частіше за все REST API.

Основні переваги монолітної архітектури включають те що ця архітектура є більш простою для початкового налаштування, і те що розгортання застосунків з цією архітектурою є значно простішим за рахунок наявності всього одного сервісу. Найбільшим недоліком монолітної архітектури є дуже невелика гнучкість - через тісну пов’язаність модулів всередині моноліту такі застосунки дуже важко і дорого масштабувати. Також монолітна архітектура обмежує розробників тільки однією технологією для всього застосунку, що може негативно вплинути на якість коду і продуктивність.

Основною ж перевагою мікросервісної архітектури є гнучкість. За рахунок того що окремий фукціонал ізольований в різних сервісах, точкове масштабування є в рази простішим ніж при монолітній архітектурі. Розподілення кодової бази між окремими сервісами також дозволяє використання різних технологій для сервісів в залежності від задач кожного сервісу, що значно покращує якість коду. Недоліком мікросервісної архітектури є потенційні складності при розгортанні великої кількості сервісів, а також менша продуктивність при невеликій кількості користувачів, що обумовлено додатковими затримками на комунікацію між сервісами, хоча при великій інтенсивності запитів продуктивність мікросервісних та монолітних застосунків є достатньо схожою[17].

Оскільки в рамках розробки дипломного проєкту необхідно розробити як частину машинного навчання, так і саму основу веб-застосунку, було обрано використати мікро-сервісну архітектуру, що дозволить більш гнучко обирати технології для машинного навчання і самого веб-застосунку.

## Аналіз та моделювання бізнес-процесів

Для моделювання бізнес-процесів пошуку вакансій та адаптації резюме використовуються BPMN моделі, зображені на рисунках 1.13 та 1.14.

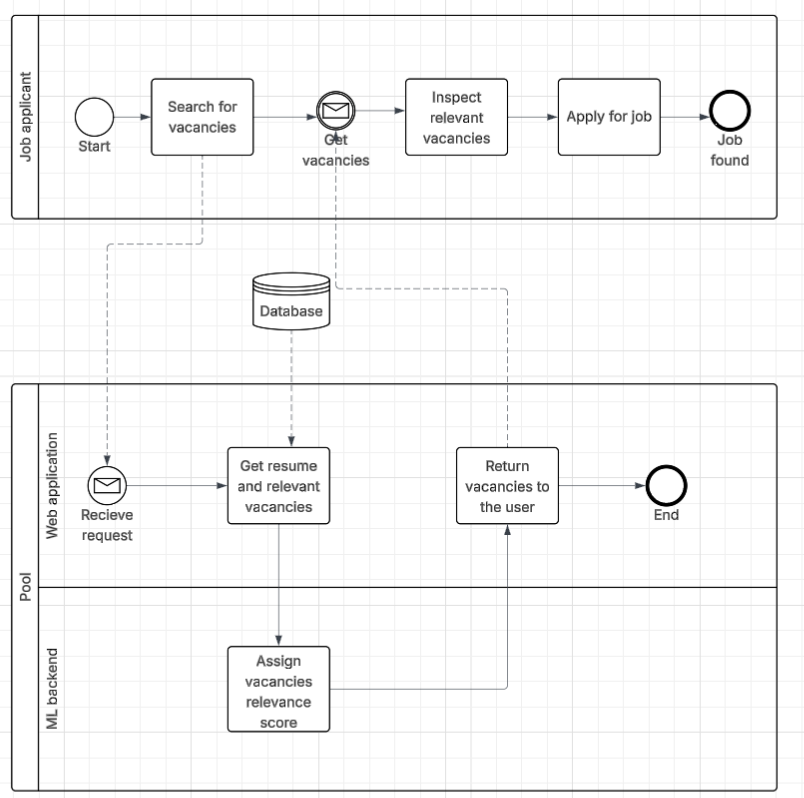
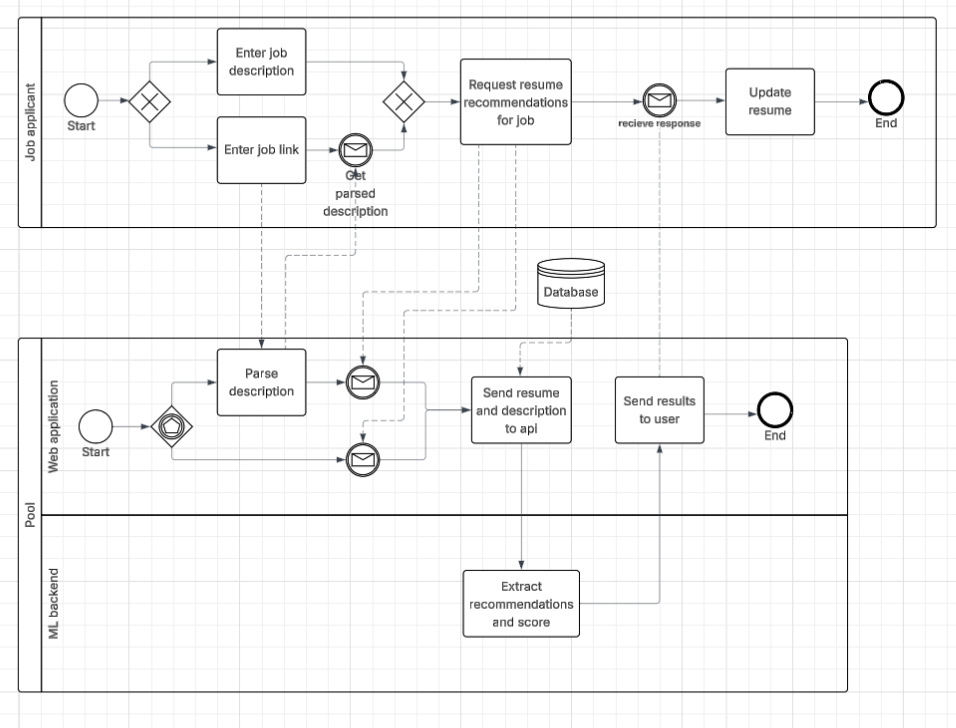


Рисунок 1.13 - BPMN модель процесу пошуку вакансій

Рисунок 1.14 - BPMN модель процесу адаптації резюме

Опис моделі бізнес-процесу пошуку вакансій:

* Користувач ініціює пошук вакансій на веб-сайті;
* Веб-застосунок дістає резюме користувача та релевантні вакансії з бази даних та надсилає їх на апі машинного навчання;
* Апі машинного навчання оцінює релевантність кожної вакансії до резюме користувача та надсилає результати веб-застосунку;
* Веб-застосунок повертає результати користувачу;
* Користувач обирає вакансію та подається на роботу.

Опис бізнес-процесу адаптації резюме:

* Користувач вводить посилання на вакансію, під яку він хоче адаптувати резюме, або текст цієї вакансії;
* Якщо користувач ввів посилання, веб-застосунок зчитує опис вакансії та повертає його користувачу;
* Користувач ініціює оцінку свого резюме до тексту вакансії;
* Веб-застосунок надсилає резюме та текст вакансії до апі машинного навчання;
* Апі машинного навчання надає оцінку релевантності резюме до вакансії та список ключових слів спільних та відмінних між резюме і вакансією;
* Веб-застосунок надає користувачу результати;
* Користувач змінює своє резюме відповідно до рекомендацій.

## Висновки до розділу

В рамках першого розділу дипломного проєкту було поставлено задачі для проєкту та проаналізовано предметну область сфери працевлаштування в ІТ. Було проаналізовано історію диджиталізації цієї сфери та поточний її стан, а саме етапи працевлаштування на яких використовуються технології ШІ. Було проведено аналіз проблем з якими може стикнутись кандидат при пошуку роботи і запропоновано спосіб їх часткового вирішення.

Далі було проведено аналіз існуючих рішень в обраній сфері, таких як Djinni, Jobscan та Enhancv, і порівняльний аналіз між дипломним проєктом та аналогами і виділено переваги та недоліки перед кожним з них. Було проведено аналіз існуючих алгоритмічних рішень машинного навчання для поставлених задач, а також аналіз потенційних варіантів архітектурних рішень застосунку.

Було проаналізовано та промодельовано 2 основних бізнес-процеси, що будуть реалізовані в дипломному проєкті, а саме процес пошуку релевантних вакансій за резюме пошукача і адаптацію резюме під конкретну вакансію.

# РОЗРОБЛЕННЯ вимог до програмного забезпечення

## Варіанти використання програмного забезпечення

Діаграма варіантів використання з ключовим актором та основними функціями показана на рисунку 2.1

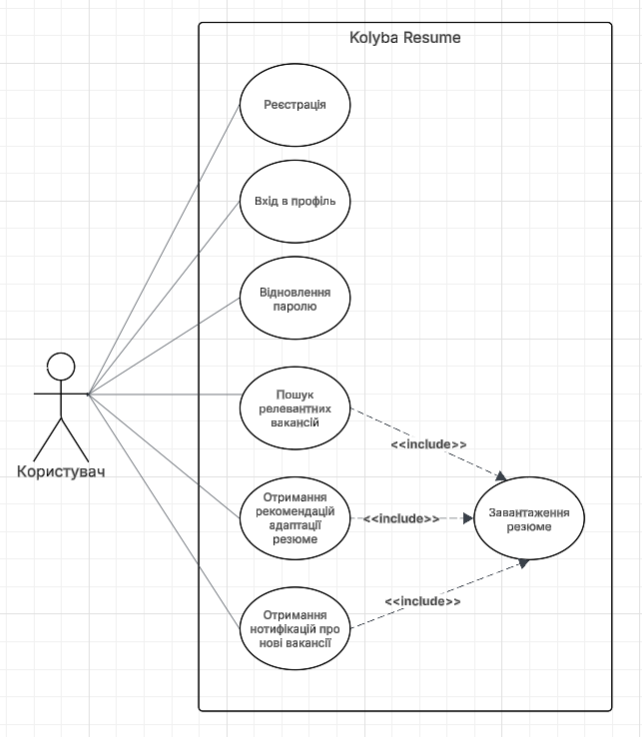


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.7 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

|  |  |
| --- | --- |
| Use case name | Реєстрація |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Користувач |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, його ім’я і пароль в системі. Після заповнення даних користувача натискає кнопку реєстрації. З’являється повідомлення про те що користувачу було надіслано посилання для підтвердження імейлу |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно  підсвічується червоним надписом. |
| Post-Condition | Користувач може увійти в систему |

Tаблиця 2.2 – Варіант використання UC-02

|  |  |
| --- | --- |
| Use case name | Вхід в профіль |
| Use case ID | UC-02 |
| Goals | Вхід користувача до системи |
| Actors | Користувач |
| Trigger | Користувач бажає увійти в систему |
| Pre-conditions | Користувач зареєстувався і підтвердив імейл |
| Flow of Events | Користувач переходить на сторінку входу. В поля для реєстрації вводяться пошта користувача і пароль в системі. Після заповнення даних користувача натискає кнопку входу. Після цього користувач переходить на головну сторінку |
| Extension | В випадку введення не коректних даних, кнопка входу стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно  підсвічується червоним надписом. |
| Post-Condition | Користувач перебуває на головній сторінці застосунку |

Таблиця 2.3 – Варіант використання UC-03

|  |  |
| --- | --- |
| Use case name | Відновлення паролю |
| Use case ID | UC-03 |
| Goals | Відновлення паролю користувача |
| Actors | Користувач |
| Trigger | Користувач бажає увійти в систему |
| Pre-conditions | Користувач забув свій пароль або хоче його змінити |
| Flow of Events | Користувач переходить на сторінку відновлення паролю. У відповідне поле вводиться імейл користувача. Далі користувач натискає кнопку відновлення паролю, після чого йому надсилається імейл з посиланням відновлення паролю і він може перейти по ньому і зробити це |
| Extension | В випадку якщо введений імейл не існує в системі, користувач отримує відповідне повідомлення і кнопка відновлення паролю стає неактивною |
| Post-Condition | Користувач може увійти в систему |

Таблиця 2.4 – Варіант використання UC-04

|  |  |
| --- | --- |
| Use case name | Завантаження резюме |
| Use case ID | UC-04 |
| Goals | Завантаження резюме користувача до системи |
| Actors | Користувач |
| Trigger | Користувач бажає завантажити резюме до системи |
| Pre-conditions | Користувач зареєстрований |
| Flow of Events | Користувач переходить на головну сторінку застосунку. Далі він натискає на кнопку завантаження резюме та завантажує своє резюме у форматі pdf, docx або doc. Після завантаження користувач отримує відповідне повідомлення |

Продовження таблиці 2.4

|  |  |
| --- | --- |
| Extension | В випадку якщо файл неможливо було зчитати користувач отримує відповідне повідомлення про помилку. |
| Post-Condition | Користувач може здійснити пошук вакансій або адаптацію резюме |

Таблиця 2.5 – Варіант використання UC-05

|  |  |
| --- | --- |
| Use case name | Пошук вакансій |
| Use case ID | UC-05 |
| Goals | Показ користувачу релевантних вакансій за його резюме |
| Actors | Користувач |
| Trigger | Користувач бажає знайти релевантні для нього вакансії |
| Pre-conditions | Користувач зареєстрований, має завантажене резюме |
| Flow of Events | Користувач переходить на головну сторінку застосунку. Далі він натискає на кнопку пошуку вакансій. Після цього він переадресовуюється на сторінку результатів пошуку, де відображається список вакансій, відсортованих за тим наскільки вони підходять до резюме користувача |
| Extension | - |
| Post-Condition | - |

Таблиця 2.6 – Варіант використання UC-06

|  |  |
| --- | --- |
| Use case name | Адаптація резюме |
| Use case ID | UC-06 |
| Goals | Дати користувачу рекомендації адаптації резюме під вакансію |
| Actors | Користувач |
| Trigger | Користувач бажає адаптувати своє резюме під вакансію |
| Pre-conditions | Користувач зареєстрований, має завантажене резюме |

Продовження таблиці 2.6

|  |  |
| --- | --- |
| Flow of Events | Користувач переходить на головну сторінку застосунку. Далі він натискає на кнопку адаптації резюме. Після цього відображається вікно в якому користувач може ввести посилання на вакансію і натиснути кнопку “Зчитати вакансію”, або ввести текст вручну. Коли текст вакансії введено користувач натискає кнопку “Отримати рекомендації”, після чого отримує оцінку наскільки резюме підходить до вакансії та які ключові слова варто було б додати до резюме |
| Extension | В випадку, якщо опис вакансії не вдалось зчитати за посиланням, користувачку показується відповідне повідомлення і пропонується ввести текст вручну |
| Post-Condition | - |

Таблиця 2.7 – Варіант використання UC-07

|  |  |
| --- | --- |
| Use case name | Отримання нотифікацій про нові вакансії |
| Use case ID | UC-07 |
| Goals | Сповіщення користувача про нові вакансії, релевантні до його резюме |
| Actors | Користувач |
| Trigger | Користувач бажає отримувати сповіщення про нові вакансії |
| Pre-conditions | Користувач зареєстрований, користувач має завантажене резюме |
| Flow of Events | При завантаженні нових вакансій в систему, вона перевіряє їх релевантність до резюме користувача та надсилає вакансію на імейл користувача у випадку виявлення співпадіння |
| Extension | - |
| Post-Condition | - |

## Розроблення нефункціональних вимог

Веб-застосунок має коректно обробляти  помилки як на серверній, так і на клієнтській стороні, повертаючи відповідні статус коди HTTP запитів та відображаючи відповідні повідомлення про помилки. Помилки на серверній стороні застосунку повинні бути оброблені таким чином, щоб вони не призводили до недоступності веб-застосунку.

Програмне забезпечення має працювати на всіх основних браузерах, зокрема: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та Opera, як на персональних комп’ютерах так і на мобільних пристроях.

Програмне забезпечення має забезпечувати продуктивність, характеризовану затримкою запитів до веб-серверу довжиною не довше за пів хвилини.

Вакансії, що відображаються у веб-застосунку мають бути актуальними як мінімум на момент початку поточного дня.

## Аналіз системних вимог

Мінімальні системні вимоги для персональних комп’ютерів є наступними:

* Процесор Intel Core i3;
* ОЗП 4 Гб;
* Наявність Інтернет-з’єднання зі швидкістю не меншою за 1 Мбіт/с.

Рекомендовані системні вимоги для ПК наступні:

* Процесор Intel Core i7;
* ОЗП 8 Гб;
* Наявність Інтернет-з’єднання зі швидкістю не меншою за 20 Мбіт/с.

Мінімальні системні вимоги для мобільних пристроїв на iOS або Android є наступними:

* Процесор Qualcomm MSM8937 Snapdragon 430;
* ОЗП 2 Гб;
* Наявність Інтернет-з’єднання зі швидкістю не меншою за 1 Мбіт/с.

Рекомендовані системні вимоги для мобільних пристроїв на iOS або Android є наступними:

* Процесор Qualcomm SDM845 Snapdragon 845;
* ОЗП 4 Гб;
* Наявність Інтернет-з’єднання зі швидкістю не меншою за 20 Мбіт/с.

## Аналіз економічних показників програмного забезпечення

Вихідні дані проєкту, на основі яких буде розраховано відповідні економічні показники є наступними: кількість рядків коду становить близько 5000, тобто LOC = 5000, категорія проєкту (COCOMO) – Organic, кількість розробників – два, зарплата кожного з розробників взята на рівні середньої зарплати розробників рівня junior і становить 900$ на місяць. Фактична тривалість розробки становить 2.5 місяці.

Основними метриками, які будуть розраховані для даного проєкту, є оцінка трудомісткості, оцінка календарної тривалості та собівартість розробки. Базова модель Basic COCOMO має наступні коефіцієнти для класу Organic: , . Таким чином, формула для обчислення трудомісткості наступна:

Тут – це розмір проєкту, причому . У результаті підстановки відповідних коефіцієнтів отримано 𝐸. Отже, прогноз моделі COCOMO є у 2.6 разів більшим за фактичне значення трудомісткості (). Дана різниця може бути пояснена відносно невеликим реальним обсягом коду для MVP та використанням бібліотек, зокрема для задач машинного навчання.

Календарна тривалість проєкту обраховується за наступною формулою на основі оцінки трудомісткості:

Підстановка коефіцієнтів та дає наступне значення тривалості: Оскільки дана метрика є прямо пропорційною трудомісткості, то причинами такої оцінки моделі, яка є більшою за фактичне значення , є ті ж причини, що й для оцінки трудомісткості.

Собівартість розробки складається із зарплат та одноразових платежів. Оскільки місячна зарплата становить 900$, розробників двоє, а фактична тривалість розробки 2.5 місяців, то загальні витрати на зарплати становлять . В одноразові платежі входить підписка на сервіс Google Colab, що дозволяє використовувати відносно потужні графічні чіпи для швидкого навчання нейромереж, які є важливою частиною проєкту. Дана підписка коштує 12$ на місяць, але, враховуючи, що даний сервіс потрібен лише на один місяць, то це всі одноразова витрати. Крім того, для даного проєкту, щоб покрити весь цикл MVP-роботи, достатньо близько двох місяців його роботи, тобто це два місяці оплати хостингу. Середня ціна хостингу на місяць мінімальної конфігурації становить близько 50$, тобто сумарна вартість виходить 100$. Таким чином, собівартість розробки .

## Висновки до розділу

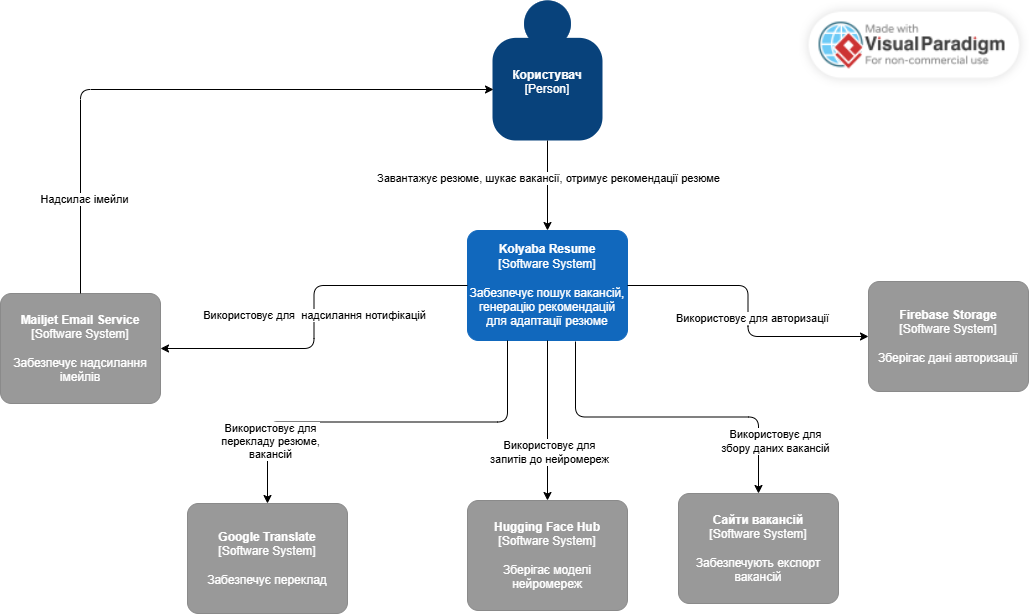
В рамках другого розділу пояснювальної записки до дипломного проєкту було наведено варіанти використання програмного забезпечення. Далі було наведено нефункціональні вимоги до програмного забезпечення та сформульовано системні вимоги до нього. Було проведено аналіз економічних показників ПЗ та наведено його приблизну собівартість.

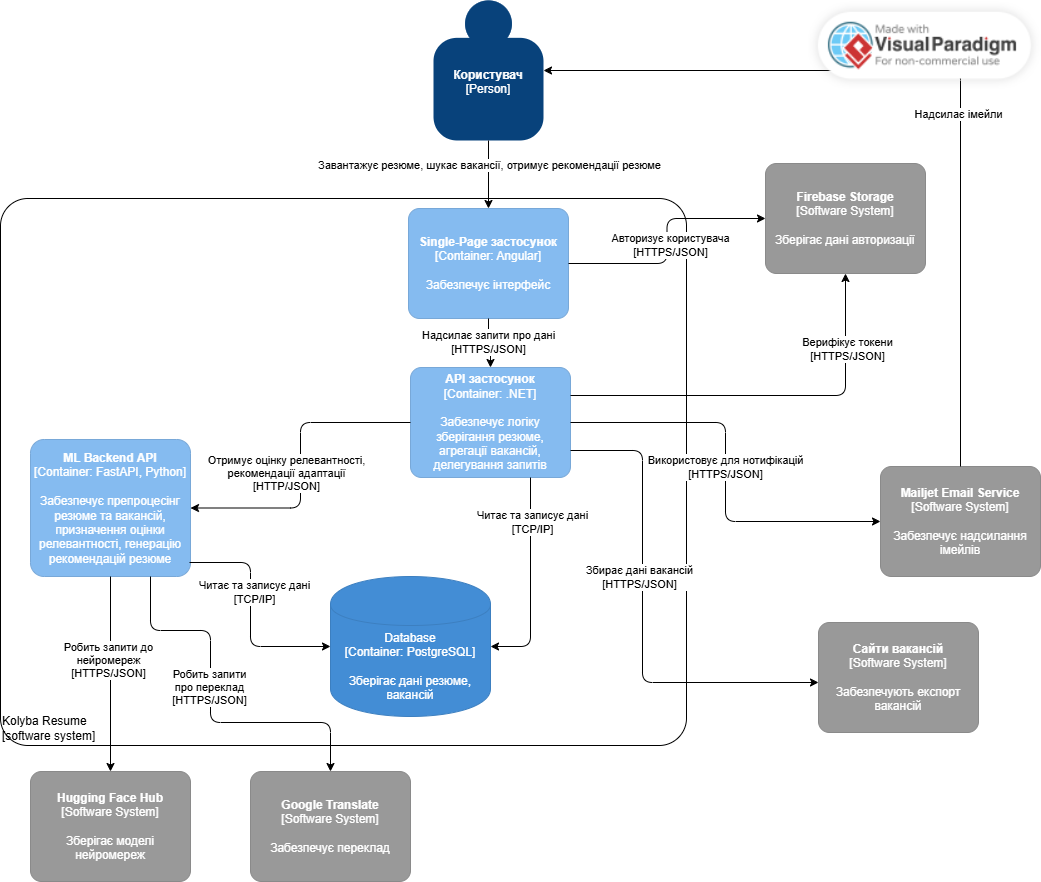
# КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Архітектура програмного забезпечення

Для розробки програмного забезпечення було обрано використання клієнт-серверної архітектури, з використанням мікросервісної архітектури для серверної частини та Single-Page Application для клієнтської частини.Серверна частина застосунку складається з 2 сервісів. Перший з них служить як Gateway, і відповідає за агрегацію вакансій, зчитування та збереження резюме користувача, а також делегацією викликів до сторонніх сервісів та іншого сервісу застосунку. Другий сервіс - це API до частини веб-застосунку пов’язаної з машинним навчанням. Він відповідає за генерацію рекомендацій для адаптації резюме та оцінку релевантності резюме до вакансії.

Перші 2 рівні діаграми C4, що описують архітектуру програмного забезпечення, наведені на рисунках 3.1 та 3.2:

Рисунок 3.1 – Перший рівень діаграми C4 архітектури програмного забезпечення

Рисунок 3.2 – Другий рівень діаграми C4 архітектури програмного забезпечення

## Архітектурні рішення та обґрунтування вибору засобів розробки

Для забезпечення нефункціональної вимоги про продуктивність застосунку було прийнято рішення про синхронізацію бази даних вакансій з експортами сайтів вакансій раз на день та попередню обробку даних вакансій та резюме.

Один раз на день починає роботу компонент програмного забезпечення, що зчитує усі вакансії з сайтів вакансій та порівнює їх з наявними у базі даних. Після цього він видаляє з бази вакансії що не є активними і додає нові, та передає АРІ сервісу машинного навчання дані про нові вакансії, який в свою чергу очищує текст описів вакансій та проводить їх векторизацію.

Аналогічно, при завантаженні резюме користувача до АРІ сервісу машинного навчання надсилається запит, і він очищує та векторизує текст резюме.

Завдяки цьому при запиті на отримання вакансій по резюме немає необхідності на обробку резюме та вакансій, а тільки безпосередньо на оцінку релевантності вакансій до резюме, що значно пришвидшує роботу застосунку.

Щодо типу баз даних, було розглянуто реляційну та векторну базу даних. Перевагою векторної бази даних є те, що вона дозволяє виконувати швидкий пошук схожих текстових документів за рахунок збереження векторних представлень текстів, реалізації специфічної індексації та обчислення схожості основі подібності між векторами за косинусом кута між ними. Реляційна база даних у свою чергу є більш гнучкою з точки зору зручності інтеграції ORM та міграцій. Для оптимізації пошуку релевантних вакансій було вирішено класифікувати резюме за категорією, тобто сферою в рамках IT, до якої відноситься релевантна посада кандидата. Таким чином, пошук вакансій виконується лише вакансій цієї категорії. Також для кожного резюме створюється  та зберігається у вигляді байтів векторне представлення при його завантаженні. Аналогічно для кожної з нових вакансій. За такого підходу досягається висока швидкість та точність пошуку релевантних вакансій навіть у реляційній базі даних. Таким чином, враховуючи вище зазначені переваги реляційної бази даних та при цьому оптимізація пошуку вакансій, було обрано саме цей тип баз даних.

## Конструювання програмного забезпечення

### Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання користувачів, даних про їхні резюме, а також вакансій та посилання на відповідні компанії, які опублікували ці вакансії. Опис таблиць бази даних наведено у таблицях 3.1-3.4. Модель бази даних наведена на рисунку 3.2.

Таблиця 3.1 – Опис таблиці Users

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| Id | serial | ідентифікаційний номер користувача |
| Uid | text | ідентифікатор користувача в Firebase |
| Name | varchar(50) | ім'я користувача |
| Email | varchar(50) | електронна пошта користувача |

Таблиця 3.2 – Опис таблиці Resumes

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| Id | serial | ідентифікаційний номер резюме |
| Text | text | текст резюме |
| CleanedText | text | очищений текст резюме від HTML тегів, спецсимволів та контактних даних |
| Vector | bytea | векторне представлення тексту резюме |
| Category | integer | кодування сфери діяльності, до якої відноситься релевантна посада для резюме, визначена нейромережою |
| UserId | bigint | ідентифікаційний номер користувача, якому належить резюме |

Таблиця 3.3 – Опис таблиці Vacancies

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| Id | serial | ідентифікаційний номер вакансії |
| Title | text | назва вакансії |
| Text | text | текст вакансії |
| Url | text | посилання на сторінку вакансії |
| Salary | text | зарплата (або вилка зарплат), яка вказана у вакансії |
| Location | text | розташування офісів компанії та/або формат роботи |

Продовження таблиці 3.3

|  |  |  |
| --- | --- | --- |
| Vector | Bytea | векторне представлення тексту вакансії |
| Category | integer | кодування сфери діяльності, до якої відноситься посада, зазначена у вакансії |
| CategoryText | text | категорія посади, зазначеної у вакансії, визначена сайтом, з якого була отримана вакансія |
| Source | integer | кодування сайту, з якого була отримана вакансія |
| CleanedText | text | очищений текст вакансії від HTML тегів, спецсимволів та контактних даних |

Таблиця 3.4 – Опис таблиці Companies

|  |  |  |
| --- | --- | --- |
| Назва поля | Тип даних | Опис |
| Id | serial | ідентифікаційний номер компанії |
| Url | text | посилання на сайт компанії |

## Аналіз безпеки даних

Розроблене програмне є однокористувацьким, при чому авторизація користувача відбувається за допомогою Firebase, який відповідає в тому числі за безпечне зберігання даних користувача. За рахунок цього в базі даних розробленого програмного забезпечення зберігаються тільки такі дані користувача як ім’я та текст резюме, що є публічними, а значить навіть потенційні вразливості в програмному забезпеченні не є критичними, адже воно не зберігає приватних даних.

Програмне забезпечення було розроблено притримуючись рекомендацій Microsoft щодо роботи з даними [20]. Однією з головних проблем яку це вирішує це атаки з використанням SQL ін’єкцій, оскільки при використанні LINQ підходу роботи з Entity Framework Microsoft гарантує неможливість таких атак.

## Висновки до розділу

В рамках 3 розділу пояснювальної записки було промодельовано архітектуру програмного забезпечення за допомогою нотації С4 і описано архітектурні рішення, прийняті в ході його конструювання. Далі було описано базу даних програмного забезпечення та проаналізовано безпеку даних програмного забезпечення.

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Опис контрольного прикладу

<У підрозділі викладають:

повний опис контрольного прикладу з усіма можливими розгалуженнями та особливостями. Кроки доповнюють ілюстраціями. Контрольний приклад описує ключовий функціонал.>

## Висновки до розділу

<Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

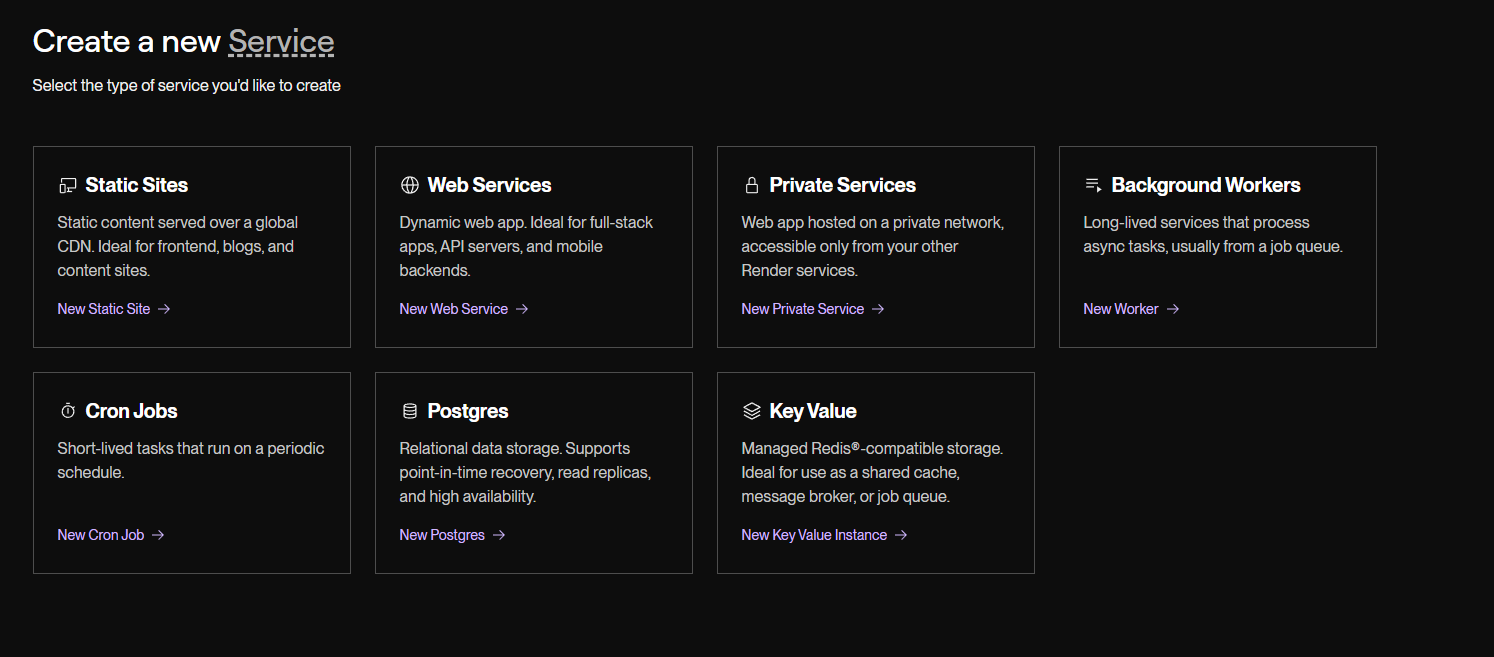
Для розгортання програмного забезпечення було розглянуто 3 основних варіанти: розгортання з використанням віртуальної машини на сервісі Digital Ocean [19], розгортання на Render [20] чи розгортання на платформі Azure [20].

Перевагою використання віртуальної машини є висока гнучкість налаштування і отримання середовища що близьке до локального. Головним недоліком такого підходу є складність розгортання.

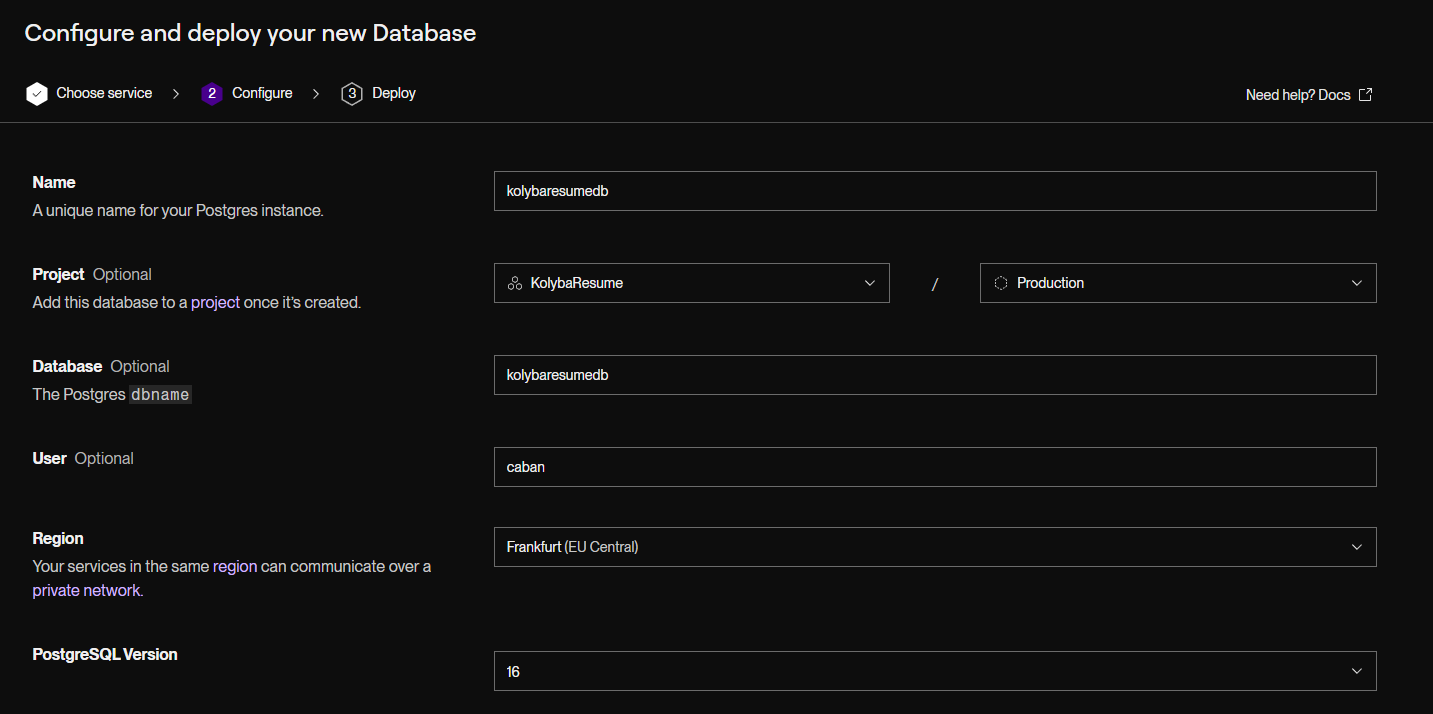
Головними перевагами Azure є дуже розвинена інфраструктура завдяки популярності сервісу та відносна простота розгортання, але при використанні цього сервісу використовується система Pay-as-you-go, що може призвести до неочікуваних надмірних витрат.

Render же поєднує в собі простоту розгортання, оскільки він реалізує модель Platform as a Service та можливість платити за точні значення ресурсів, які виділяються кожному сервісу, завдяки чому підтримується як можливість гнучко розподіляти ресурси для сервісів, так і контроль над кількістю коштів, витрачених на них. Завдяки цьому балансу було обрано розгортання саме на платформі Render.

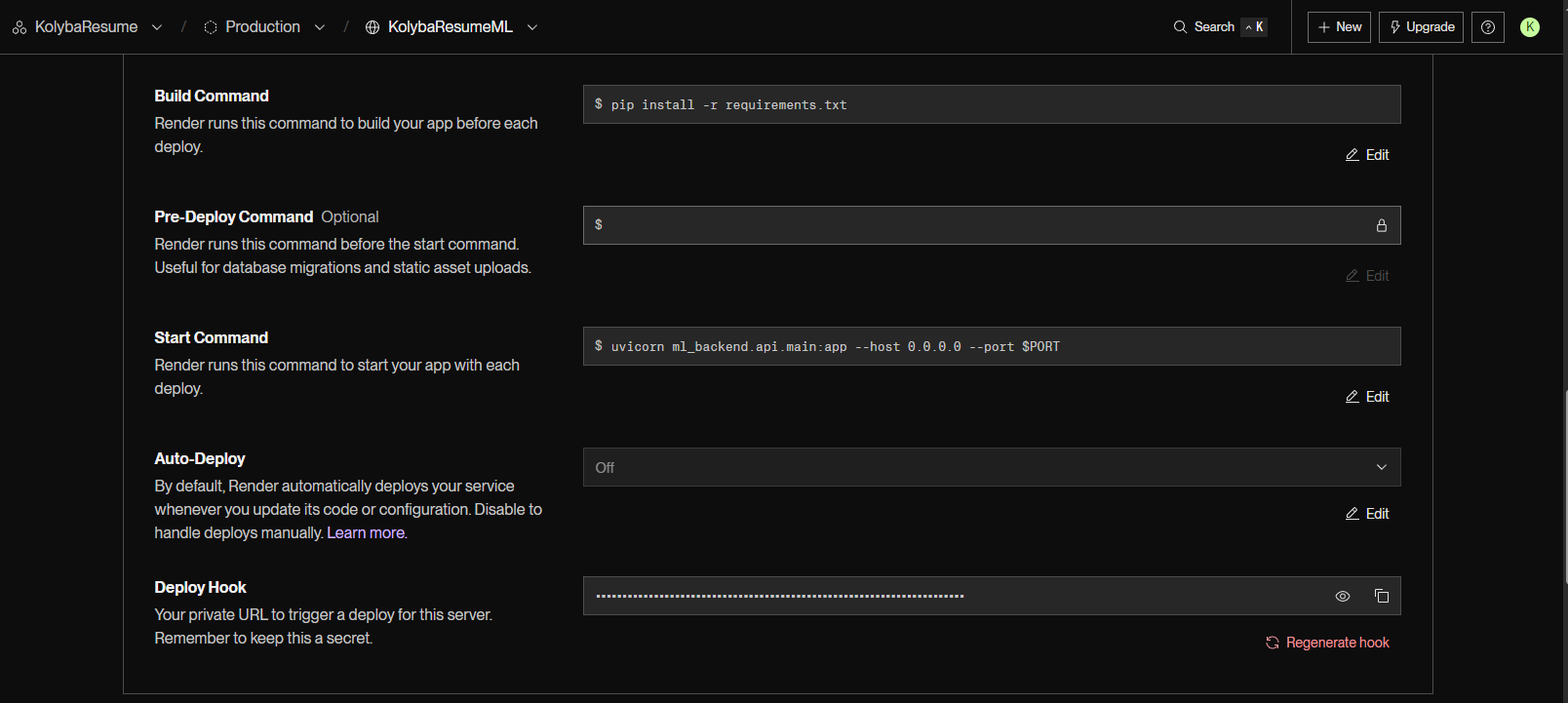
Після створення проєкту в Render користувачу пропонується створити один з сервісів, наведених на рисунку 4.1. В рамках дипломного проєкту необхідно створити базу даних Postgres, 2 веб-сервіси для бекенду застосунку та API машинного навчання, а також статичний веб-сайт для фронтенду застосунку.

Рисунок 4.1 – Підтримувані види сервісів в Render

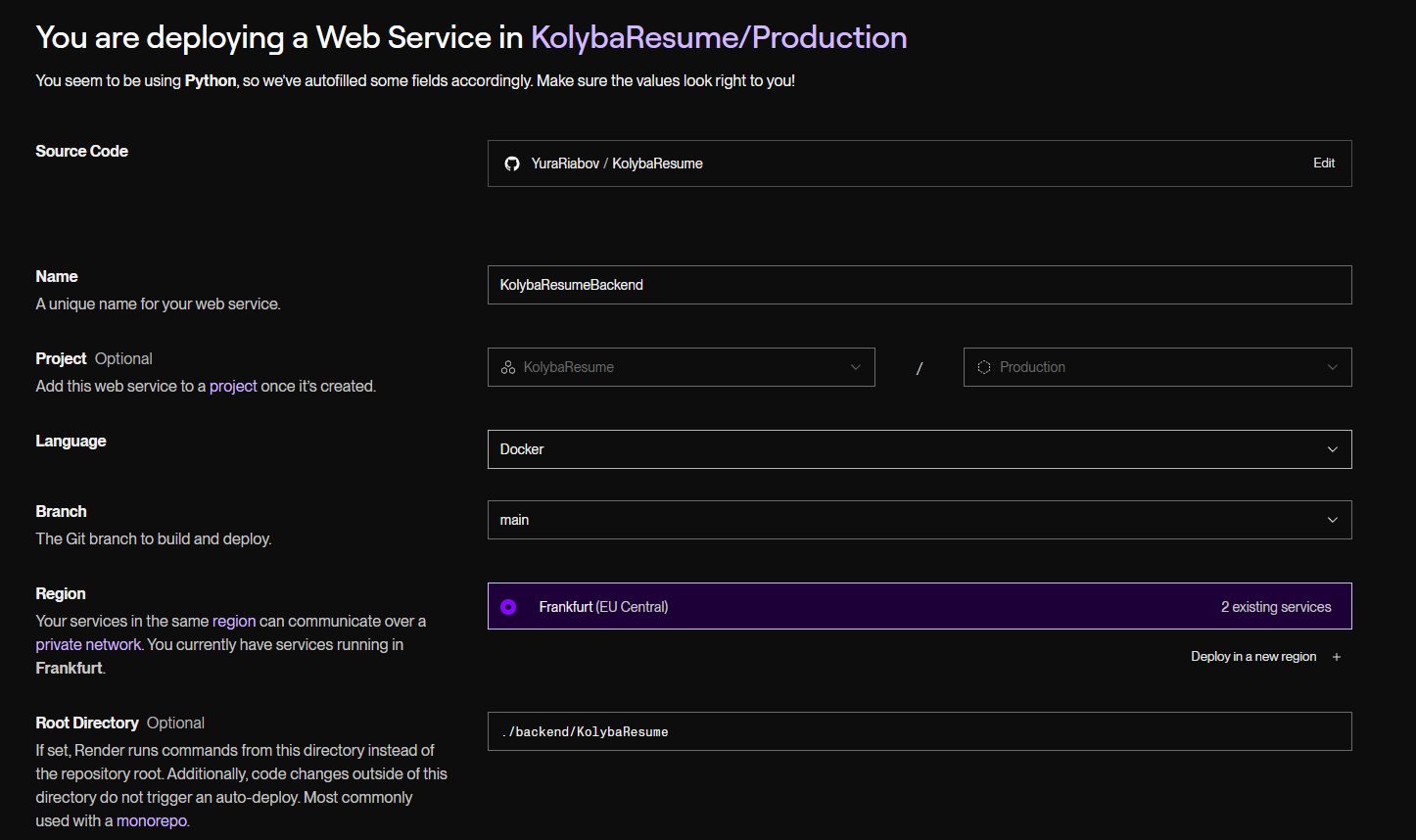
Першим чином було створено базу даних Posgres. Скріншоти налаштування наведено на рисунку 4.2

Рисунок 4.2 – Конфігурація бази даних

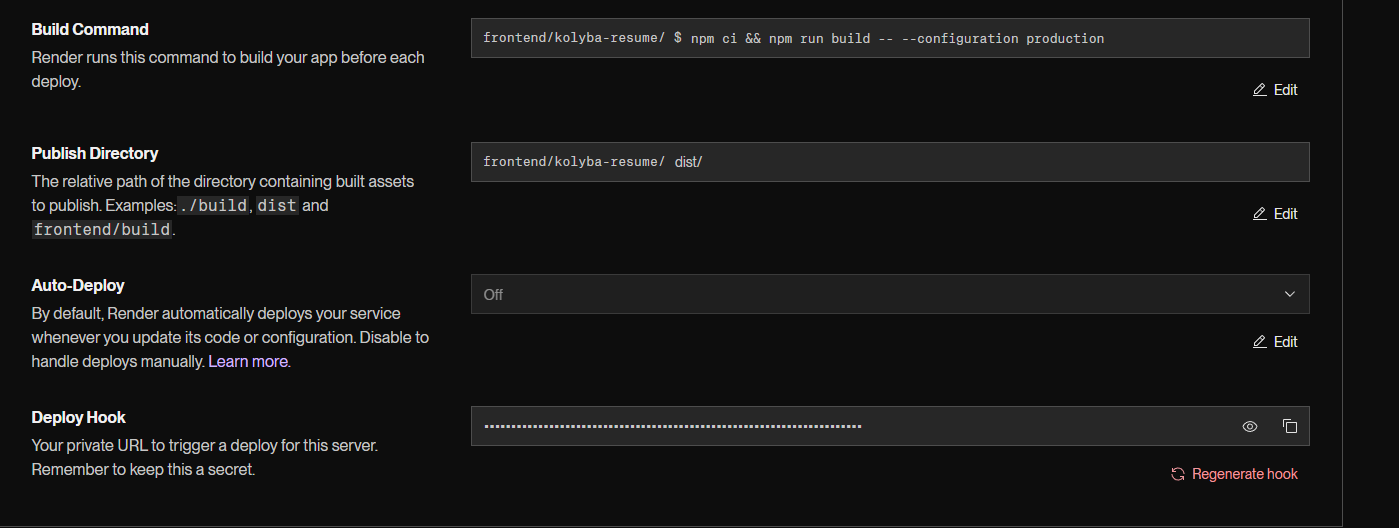
Після успішного створення бази даних було оновлено строку підключення до бази даних в конфігураціях в бекенд сервісах і перейдено до розгортання сервісу машинного навчання. Render нативно підтримує розгортання застосунків, написаних на Python3, тому було достатньо обрати цю опцію, встановити змінні середовища та вказати скрипти для збірки та початку роботи сервісу. Конфігурацію цього сервісу наведено на рисунку 4.3:

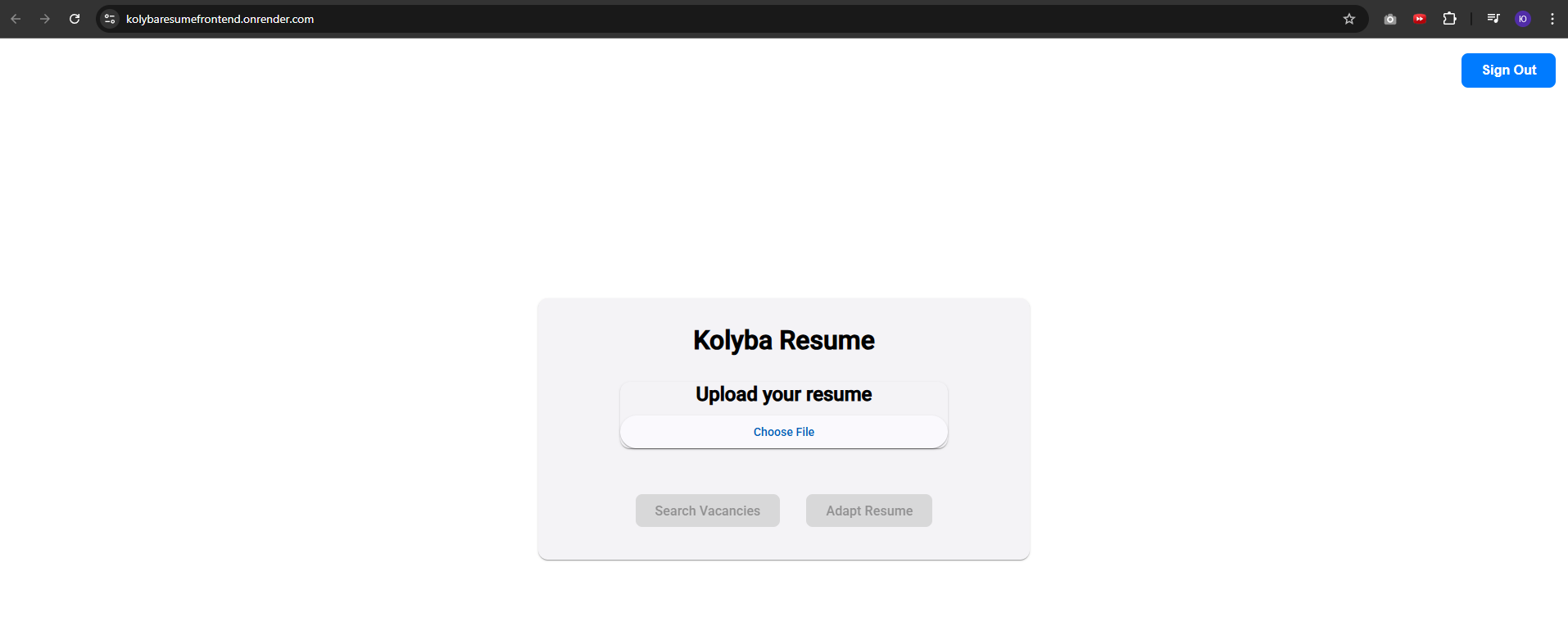
Рисунок 4.3 – Команди збірки та початку роботи сервісу машинного навчання

Після конфігурації сервісу машинного навчання було отримано посилання на цей сервіс та оновлено його в конфігурації бекенду веб-застосунку, після чого перейдено до розгортання цього сервісу. Render не підтримує нативно застосунки, написані на платформі .NET, але у нього присутня опція використання Dockerfile для розгортання, що і було обрано. Після написання Dockerfile та додавання його до репозиторію, було виконано конфігурацію сервісу, наведену на рисунку 4.4, та виконано розгортання.

Рисунок 4.4 – Конфігурація сервісу бекенду веб-застосунку

Після успішного розгортання бекенду застосунку, посилання на нього було додано до конфігурації фронтенду застосунку, який було розгорнуто на Render як статичний вебсайт. Конфігурації вебсайту, та знімок екрану результату розгортання наведено на рисунках 4.5-4.6:

Рисунок 4.5 – Конфігурація статичного веб-сайту

Рисунок 4.6 – Результат розгортання

## Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво Користувача».

Користувачі повинні мати можливість отримати доступ до останньої версії застосунку з кожним оновленням. Для досягнення цього використовується Auto-Deploy функція сервісу Render, завдяки після кожного оновлення головної гілки Github репозиторії усі сервіси розгортаються заново. Таким чином при оновленні застосунку користувач одразу отримує доступ до оновлених сервісів серверної частини, а клієнтська частина оновлюється при оновленні сторінки за рахунок завантаження нової версії статичного сайту з хостингу.

## Висновки до розділу

У розділі було проведено порівняльний аналіз опцій розгортання застосунку, і обрано використання сервісу Render завдяки його поєднанню простоти розгортання з контролем вартості.

Було описано кроки створення бази даних застосунку та розгортання обох сервісів та статичного вебсайту, після чого було описано вимоги до супроводу застосунку та за рахунок чого вони будуть виконуватись.

# ВИСНОВКИ

У ході виконання було розроблено програмне забезпечення для автоматизації пошуку релевантних вакансій, отримання рекомендацій для адаптації резюме та отримання повідомлень про нові вакансії, що забезпечує пришвидшення пошуку вакансій кандидатами та адаптації резюме під знайдені оголошення.

Було виконано мету пришвидшення та автоматизації пошуку релевантних вакансій пошукачами роботи та адаптації під них резюме користувача за рахунок реалізації агрегації вакансій з різних сайтів пошуку роботи та їх автоматизованого пошуку з використанням методів машинного навчання, завдяки чому користувач витрачає менше часу на пошук вакансій на різних сайтах та їх підбір під своє резюме, а також за рахунок реалізації отримання рекомендацій до кожної зі знайдених вакансій або за наданою користувачем вакансією. Також було реалізовано автоматизоване надсилання сповіщень користувачам про нові вакансії, релевантні до їх резюме.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналітика Djinni за 2024 рік. URL: https://blog.djinni.co/post/2024-report (дата звернення: 16.05.2025)
2. 99+ Must-Know Resume Statistics To Get You Hired in 2025. URL: https://novoresume.com/career-blog/resume-statistics (дата звернення: 16.05.2025)
3. Applicant Tracking System Statistics. URL: https://www.selectsoftwarereviews.com/blog/applicant-tracking-system-statistics (дата звернення: 16.05.2025)
4. Collaboration among recruiters and artificial intelligence. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9516509/ (дата звернення: 16.05.2025)
5. Jobscan. URL: https://www.jobscan.co/ (дата звернення: 16.05.2025)
6. Djinni. URL: https://djinni.co/ (дата звернення: 16.05.2025)
7. Enhancv. URL: https://enhancv.com/ (дата звернення: 16.05.2025)
8. The Hubness Phenomenon. URL: https://borgelt.net/papers/cost\_hubs.pdf (дата звернення: 16.05.2025)
9. Text Classification Using Machine Learning Techniques. URL: https://www.researchgate.net/publication/228084521\_Text\_Classification\_Using\_Machine\_Learning\_Techniques (дата звернення: 16.05.2025)
10. LSTM and Bidirectional GRU Comparison for Text Classification. URL: https://www.researchgate.net/publication/374378920\_LSTM\_and\_Bidirectional\_GRU\_Comparison\_for\_Text\_Classification#fullTextFileContent (дата звернення: 16.05.2025)
11. Ensuring Brand Safety by Using Contextual Text Features: A Study of Text Classification with BERT. URL: https://uu.diva-portal.org/smash/get/diva2:1775452/FULLTEXT01.pdf (дата звернення: 16.05.2025)
12. Naive Bayes and Text Classification I. URL: https://arxiv.org/pdf/1410.5329 (дата звернення: 16.05.2025)
13. Text Categorization with Support Vector Machines. URL: https://www.researchgate.net/publication/28351286\_Text\_Categorization\_with\_Support\_Vector\_Machines (дата звернення: 16.05.2025)
14. A Survey on Text Classification: From Traditional to Deep Learning. URL: https://arxiv.org/pdf/2008.00364 (дата звернення: 16.05.2025)
15. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL: https://arxiv.org/pdf/1810.04805v2 (дата звернення: 16.05.2025)
16. How to Fine-Tune BERT for Text Classification. URL: https://arxiv.org/pdf/1905.05583 (дата звернення: 16.05.2025)
17. A Comparative Review of Microservices and Monolithic Architectures. URL: https://arxiv.org/pdf/1905.07997 (дата звернення: 16.05.2025)
18. Entity Framework Security Considerations. URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations (дата звернення: 24.05.2025)
19. DigitalOcean. URL: https://www.digitalocean.com/ (дата звернення: 16.05.2025)
20. Render. URL: https://render.com/ (дата звернення: 24.05.2025)
21. Azure. URL: https://portal.azure.com/ (дата звернення: 24.05.2025)

# ДОДАТКИ