МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
19 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інтелектуальна технологія автоматизації відбору персоналу»
здобувача групи ІН.м-23 Клименка Богдана Михайловича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Богдан КЛИМЕНКО
(підпис)

Керівник,
асистент кафедри комп'ютерних наук,
к.ф.-м.н. Олександр ВЛАСЕНКО _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-23 Клименка Богдана Михайловича

1. Тема роботи: «Інтелектуальна технологія автоматизації відбору персоналу
затверджую наказом по СумДУ від *«06» грудня 2023 р. № 1412-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до18 грудня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для створення інтелектуальної технології автоматизації відбору персоналу. 3) Створення інтелектуальної технології автоматизації відбору персоналу. 4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для створення інтелектуальної технології автоматизації відбору персоналу*		
3	*Створення інтелектуальної технології автоматизації відбору персоналу*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 101 стор., 52 рис., 3 додатка, 41 джерело.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі автоматизації відбору персоналу.

Об'єкт дослідження — процес автоматизованого підбору кандидатів.

Мета роботи — створення інтелектуальної технології автоматизації відбору персоналу на основі нейромережі (слабкого штучного інтелекту).

Методи дослідження — методи збору та аналізу даних.

Результати — Реалізована інтелектуальна технологія автоматизації відбору персоналу. Вона задовольняє наступним вимогам: аналізує резюме кандидатів; оцінює основні необхідні характеристики кандидата; готує рекомендацію щодо кожного кандидата; готує невеликий список питань, щоб провести пре-інтерв'ю для підтвердження навичок вказаних в резюме.

LLM, GPT, ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ,

ВІДБІР ПЕРСОНАЛУ, НЕЙРОМЕРЕЖІ,

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі, де технології стрімко змінюють способи ведення бізнесу, ключовою стає потреба в ефективному відборі персоналу. Однією з ключових задач є автоматизація відбору персоналу, яка передбачає використання передових технологій, зокрема слабкого штучного інтелекту. Очікується, що з вдосконаленням алгоритмів та забезпеченням більшої об'єктивності та етичності, ці системи стануть ще більш ефективними та невід'ємними у процесах HR-менеджменту. Наразі, інтелектуальні системи відбору персоналу вже використовуються в різних секторах, значно спрощуючи та оптимізуючи процес пошуку та підбору співробітників. Актуальність дослідження полягає у необхідності оптимізації процесу підбору кандидатів, підвищення його ефективності та об'єктивності.

Об’єкт дослідження. Процес автоматизованого підбору кандидатів, який включає аналіз резюме, оцінку кваліфікацій та формування рекомендацій.

Предмет дослідження. Методи інтелектуальної автоматизації відбору персоналу з використанням слабкого штучного інтелекту.

Гіпотеза. Використання нейромереж у процесі відбору персоналу може значно покращити якість та швидкість рішень, забезпечуючи більшу точність у виборі кандидатів.

Структура. Робота складається з декількох розділів, що включають теоретичний огляд сучасних технологій у сфері HR, аналіз методів відбору персоналу, створення інформаційної системи, яка виконує поставлене завдання та оцінку її ефективності.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Постійна глобалізація нашого світу змушує бізнес змінювати, поліпшувати, реорганізовувати свої тактики та стратегії росту і розвитку. Конкурентами стають не тільки місцеві компанії, але і організації міжнародного, або навіть світового рівня. Завдяки новим технологіям, світ стає «тіснішим . Це означає, що для організації, яка прагне залишатися в тренді і зберігати конкурентну перевагу, життєво важливо приймати ці нові технологічні досягнення.

Управління людськими ресурсами включає в себе багато аспектів, таких як навчання працівників, підбір кадрів, взаємовідносини з працівниками та розвиток організації [2]. Люди є загальновідомим джерелом знань і експертизи, на яке кожна організація може та повинна робити ставку. Тому залучення і утримання таких працівників грають важливу роль сьогодні. Єдиний очевидний спосіб вирішення цієї проблеми – підбирати кандидатів. У зв'язку з важливістю управління людськими ресурсами для організації, процес підбору ресурсів є ключем до успіху [3]. Раніше такий процес підбору був довшим, забирав багато часу та вимагав великої кількості документації для рекрутерів. Однак вже починає змінюватися завдяки популярності онлайн-рекрутингу [4].

У пізніших роках через технологічні зміни проводилися дослідження щодо того, як можна поєднати два важливих аспекти управління людськими ресурсами і технологій. Зазвичай проводяться дослідження щодо того, як можна зробити процес підбору співробітників більш гладким і оптимізованим за допомогою технологій [5]. Зараз увага більше приділяється технологічним досягненням, які допомагають рекрутерам, наприклад, процес стає більш автоматизованим. З цього приводу можна сказати, що людський фактор у рекрутингу зменшується[6]. У статті Бакстера [7] розглядаються тенденції, які мали б стати актуальними в рекрутингу в майбутньому році. Він пропонує використовувати прогнозуючий аналіз для усунення частини припущень, які зазвичай виникають у рекрутингу, а також згадує штучний інтелект (ШІ) як інструмент,

який буде використовуватися під час співбесіди з кандидатами [7]. Ця дисертація спрямована на дослідження аспекту однієї з новітніх технологій - штучного інтелекту (ШІ). Застосування ШІ в управлінні людськими ресурсами було однією з найважливіших тенденцій серед професіоналів у сфері рекрутингу. Впровадження ШІ в управління людськими ресурсами та рекрутинг можна назвати "новою ерою управління людськими ресурсами", оскільки ШІ змінює галузь рекрутингу, замінюючи рутинні завдання, які раніше виконували люди[8].

Зазначається [9], що штучний інтелект, як галузь, є широким та мультидисциплінарним напрямком, який може бути використаний не лише в обчислювальних дисциплінах, а також у лінгвістиці та філософії. ШІ може набувати різних форм, таких як роботи, боти чи програмне забезпечення [9]. Концепція штучного інтелекту є однією з найновіших областей в інженерії та науці, і її вивчали з часів Другої світової війни. Назва "Штучний Інтелект" була запроваджена в 1956 році [10]. Відомі вчені Салін та Уінстон визначають ШІ як набір технік, що дозволяють комп'ютерам виконувати завдання, які інакше вимагали б розумових здібностей, які природні для людського інтелекту [11]. За словами Нільссона [12], машини повинні бути здатні виконувати більшість завдань, які вимагають рівня інтелекту, характерного для людини, і це він називає інтелектом на рівні людини.

На сьогоднішній день процес підбору персоналу в основному здійснюється людськими рекрутерами, які особисто переглядають резюме, онлайн-профілі та інші джерела для пошуку кандидатів. Рекрутери проводять усі початкові контакти, надають зворотній зв'язок відхиленим кандидатам і проводять співбесіди з кандидатами [4]. Оскільки у людей є обмежені можливості, виконання всіх необхідних завдань не є легкою справою і, як правило, вимагає великої кількості відданого часу від кожного окремого рекрутера. Виявлено, що існують людські обмеження, такі як упередження, попередні уявлення та обмеження часу, які можуть ускладнити ефективність процесу підбору

персоналу [13]. Це проблема, оскільки в свою чергу це може призвести до втрат кращих кандидатів на роботу та фінансових витрат[14].

Було встановлено, що методи вивчення технологій рекрутингу потребують розвитку і відстають від сучасної практики. Тому в майбутньому потрібно провести більш глибокі емпіричні дослідження з урахуванням нових технологій, які надають більше гнучкості та кращий доступ, ніж раніше [15]. Проте через кілька років та сама проблема залишається незмінною, оскільки Марлер і Фішер [16] зазначають, що поточна література не містить нових методів технологічного рекрутингу, які потрібно вивчити. Крім того, наслідки нових технологій для управління людськими ресурсами все ще не зовсім ясні для рекрутерів, чи нові та ефективні технології призводять до викликів чи можливостей для роботи рекрутерів [17] [6]. Оскільки поточна література все ще стикається зі схожою проблемою, як і у 2000-х роках, потрібно провести більш глибоке розуміння цієї теми з урахуванням того, що нові технології стають частиною повсякденної роботи рекрутерів.

З розвитком цифрових технологій , маємо ситуацію, коли інтелектуальні технології автоматизації відбору персоналу пропонують нові можливості для вирішення цих викликів, проте вони також ставлять певні питання та вимагають глибокого дослідження. Оскільки кількість доступних даних про кандидатів зросла експоненційно, то сучасні компанії отримують сотні, а іноді й тисячі резюме на одну вакансію. Традиційні методи відбору, що базуються переважно на ручному аналізі, стають дуже затратними за часом та не завжди ефективними. Тут інтелектуальна автоматизація може відіграти ключову роль, дозволяючи швидко обробляти великі обсяги даних та виділяти найбільш перспективних кандидатів.

Штучний інтелект (AI) та машинне навчання набувають все більшого значення у сфері управління людськими ресурсами. Вони використовуються для автоматизації різних процесів, від скринінгу резюме до аналізу відповідей кандидатів на співбесіді. Інноваційні системи, такі як програми для аналізу мовлення або нейромережі для оцінки навичок, допомагають виявляти найбільш

підходящих кандидатів, знижуючи при цьому суб'єктивний фактор у прийнятті рішень.

Однією з головних переваг інтелектуальної автоматизації є здатність значно підвищити ефективність процесу відбору. Це досягається за рахунок зменшення часу, необхідного для обробки кандидатур, та підвищення точності відбору. Автоматизовані системи можуть аналізувати великі обсяги даних, виявляючи складні взаємозв'язки та використовуючи передові методи оцінки кваліфікацій, що забезпечує об'єктивність та прозорість процесу.

Використання інтелектуальних технологій у відборі персоналу порушує важливі етичні питання, особливо щодо упередженості та конфіденційності даних. Існує ризик, що алгоритми, які не були належним чином налаштовані, можуть демонструвати упереджене ставлення до кандидатів на основі їхнього віку, статі, раси чи інших характеристик. Важливо забезпечити, щоб системи були прозорими та справедливими.

Майбутнє інтелектуальних технологій у відборі персоналу виглядає досить гарно, особливо з огляду на постійне вдосконалення AI та машинного навчання. Прогрес у цих областях обіцяє ще більшу точність та ефективність у відборі кандидатів, а також інтеграцію з іншими HR-процесами, забезпечуючи комплексний підхід до управління персоналом.

Актуальність інтелектуальної автоматизації у відборі персоналу визначається її спроможністю вирішувати ключові виклики сучасного HR-менеджменту.

1.2 Постановка задачі

Метою є створення програмного забезпечення для функціонування інтелектуальної системи для автоматизації відбору персоналу. Програмне забезпечення повинно бути відкритим, регульованим. Потрібно створити платформу, де кандидати змогли б показати свій рівень підготовки – глибину теоретичних та практичних знань, а рекрутери, не витрачаючи зайвого часу, знаходити

потенційних працівників спираючись на отримані рекомендації від інтелектуальної системи.

Дане дослідження проводиться та використовується в рамках кваліфікаційної роботи магістра за спеціальністю 122 «Комп’ютерні науки».

Розроблений програмний продукт має задовольняти наступні вимоги:

1. Аналізувати резюме кандидата.

2. За результатами комплексного аналізу оцінити такі характеристики кандидата:

a. Відповідність професійних навичок вимогам вакансії.

b. Відповідність досвіду роботи до цієї ролі.

c. Софтскіли, включаючи комунікаційні навички, адаптивність та критичне мислення.

d. Відповідність корпоративним культурним цінностям і культурі компанії.

3. За отриманим аналізом розглядати перспективність кандидата.

4. На основі попередніх оцінок зробити загальну рейтингову оцінку.

5. Зробити рекомендацію відносно резюме кандидата.

6. Підготувати список питань, щоб провести пре-інтерв’ю для підтвердження навичок вказаних в резюме.

2. ВИБІР МЕТОДУ РІШЕННЯ

2.1 Вибір системи

2.1.1 Загальні відомості

Швидкий прогрес та вражаючі досягнення великих попередньо навчених великих мовних моделей (BMM/LLM) викликали революційні зміни в області обробки природної мови [18-20]. Ці моделі продемонстрували значні покращення у різних застосуваннях, таких як діалог [21], узагальнення [22] та генерація коду [23]. Більшість завдань включають в себе відкриті, інтелектуально обумовлені та безпосередньо не супроводжені відповіді, а не вибір із фіксованого набору відповідей. Внаслідок цього оцінка відповідності їх створених відповідей потенційним людини стає викликом [24]. Традиційні автоматичні метрики, такі як BLEU [25] та ROUGE [26], виявили відносно низьку кореляцію з оцінками людини, особливо в завданнях з відкритим генеруванням [27], тоді як людська оцінка часто вимагає багато часу і витрат. Таким чином, існує зростаючий попит на автоматизовані методи оцінки, які можуть послідовно відповідати оцінкам людини, при цьому бути більш ефективними та економічними [28, 29].

Не дивно, що такі моделі видко стали популярними у різноманітних галузях, таких як охорона здоров'я, право, фінанси і багато інших. Вони стали невід'ємними інструментами у різних сферах застосування, включаючи медичну діагностику, аналіз юридичних документів і оцінку фінансових ризиків завдяки своєму унікальному набору функцій та збільшеній експертизі у конкретних галузях [30].

2.1.2 GPT-4

Моделі GPT [31] є підмножиною LLM моделей, оскільки термін «LLM» використовується для позначення будь-якої великомасштабної моделі, призначеної для завдань обробки природної мови. GPT або «Генеративний

Попередньо Навчений Трансформер» - це клас моделей обробки природної мови, розроблених OpenAI. Ці моделі призначені для розуміння та генерування тексту, схожого на людський, на основі отриманого введення. GPT-4, четверта та найновіша версія, є найбільшою та найвідомішою моделлю у серії GPT [32].

Вона є яскравим прикладом постійного бажання людства досліджувати та розвивати передові технологій штучного інтелекту. OpenAI безперервно представляє інноваційні моделі, а вищеназвана модель - це кульмінація їхніх зусиль. Зі своєю величезною, у1,5 трильйона, кількістю параметрів, ця модель володіє неймовірними можливостями. Велика кількість параметрів надає їй здатності розпізнавати складні зв'язки, що підвищує якість генерації тексту та його розуміння.

Використовувати можливості цієї ЛЛМ дуже легко завдяки OpenAI API, який дозволяє розробникам та дослідникам легко інтегрувати її здібності у свої проекти та дослідження. Це забезпечує простий доступ до грандіозних потужностей у генерації мови, роблячи його універсальним інструментом для різноманітних застосувань.

GPT-4 навчається на великому наборі даних, який включає книги, статті, вебсайти та інший письмовий контент. Це забезпечує моделі всебічне розуміння мови та знань у різних областях. Хоча в основі моделі лежить текстова модель, вона також ефективно працює з текстовими та візуальними входами. Модель підтримує багатомовність, дозволяючи розуміти та генерувати текст на багатьох мовах. Це розширює її доступність та універсальність, роблячи її потужним інструментом для міжнародного спілкування та створення контенту.

2.1.3 BARD

BARD є втіленням наполегливої праці команди дослідників, спрямованої на розвиток обробки мови, заснованої на міркуваннях [33]. Розроблений під керівництвом Google, ця інноваційна модель пройшла шлях від досліджень

до ретельного тестування та удосконалення. Зі своїми вражаючими 1,6 трильйона параметрів, дана модель вирізняється своєю здатністю до розуміння нюансів і залежностей.

Дана ЛЛМ відрізняється своєю унікальною здатністю генерувати науково точні пояснення та розкладати складні питання, надаючи логічні відповіді. Це робить його значущим у світі моделей. Він навчається на добірці даних, що включає різноманітні джерела, такі як наукові тексти, дослідницькі роботи та книги. Мета цього навчання - створити модель, що відмінно справляється з завданнями, пов'язаними з міркуваннями. Завдяки цьому, модель набуває глибокого розуміння наукових термінів і логіки.

Для використання її можливостей є спеціальний API, який дозволяє легко інтегрувати його функції у різноманітні застосування. BARD може обробляти не тільки текст, а й інші типи даних, що робить його відповіді більш контекстуально точними. Модель підтримує різні мови, розширюючи її застосування в глобальних завданнях.

2.1.4 LLaMA

LLaMA є провідною розробкою MetaAI у сфері досліджень та розробок у галузі штучного інтелекту [34]. Це свідчить про їхню непохитну прихильність до просування розуміння та створення природної мови. Завдяки вражаючим 1.2 трильйонам параметрів, вона демонструє своє майстерство. Її винятткові можливості завершення тексту наділяють користувачів здатністю генерувати природний і зв'язний текст з неповного введення. Більше того, семантичне розуміння моделі дозволяє їй розшифровувати складні запити та надавати точні та просвітлені відповіді. Модель також демонструє вражаючі мультимодальні можливості, що дозволяють їй обробляти та генерувати текст у поєднанні з іншими сенсорними входами. Включаючи візуальні, аудіо чи інші сенсорні дані, ця модель може створювати більш всебічні та контекстуально відповідні результати.

Дані для навчання LLaMA походять з обширного та різноманітного корпусу, що охоплює широкий спектр текстових джерел. Цей ретельно відібраний набір даних включає різноманітні текстові джерела, уважно вибрані для охоплення широкого спектру тем, жанрів та мовних варіацій. Використовуючи ці всебічні дані для навчання, вона досягає глибокого розуміння мови, що сприяє створенню контекстуально відповідного та зв'язного тексту на різноманітні теми.

Навчальні цілі тут зосереджені на моделюванні мови, дозволяючи моделі передбачати наступне слово у послідовності та розуміти структури мови. Таке цілеспрямоване навчання наділяє її здатністю виробляти виразний та значущий текст, встановлюючи її як незамінний інструмент для широкого спектру завдань. Крім того, LLaMA пропонує багатомовну підтримку, що дозволяє здійснювати комунікацію та генерувати текст на багатьох мовах. Ця особливість долає різні бар'єри, сприяючи міжмовній комунікації та розумінню.

Доступ до могутніх можливостей даної ЛЛМ забезпечується простим у використанні інтерфейсом, розробленим для безшовної інтеграції. Користувачі можуть використовувати цей інтерфейс для інтеграції навичок генерації мови у власні застосунки, платформи або творчі проекти, повністю розкриваючи потенціал цієї потужної моделі.

2.1.5 Flan-UL2

Flan-UL2 – це новий проект від Google Research, що демонструє їхнє прагнення до розвитку здатності розуміти та створювати мову [35]. Ця модель має аж 20 мільярдів параметрів, тому вона дійсно привертає увагу. Вона вирізняється серед інших моделей своїми унікальними особливостями. Дана ЛЛМ дозволяє користувачам створювати природні та логічні тексти, навіть якщо вони мають лише частину інформації для початку. Користуватися цією моделлю дуже легко, завдяки зручному інтерфейсу. Цей інтерфейс можна використовувати, щоб додати можливості генерації мови від вище названої моделі у власні

програми, платформи або творчі проекти, що допоможе повноцінно використовувати її навички.

Основна мета навчання полягає в тому, щоб модель могла передбачити наступне слово у послідовності та розуміти структуру мови. Цей підхід до навчання дозволяє Flan-UL2 створювати плавний і значущий текст, що може бути використаний для різних мовно-орієнтованих завдань. Дані для навчання охоплюють різноманітний корпус, складений з численних текстових ресурсів. Використовуючи ці обширні дані, Flan-UL2 здобуває комплексне розуміння мови, що призводить до створення зв'язного та відповідного тексту у різних областях та темах. Значна кількість параметрів дозволяє моделі розрізняти складні шаблони, що веде до точного та контекстно відповідного генерування тексту.

Навчальний набір даних включає ретельно підібрану колекцію різноманітних текстових джерел. Ці джерела, уважно відібрані для охоплення широкого спектру тем, жанрів і варіацій, допомагають Flan-UL2 досягти глибокого розуміння нюансів мови та створювати відповідний і точний текст. Модель демонструє вражаючі мультимодальні можливості, що дозволяють їй обробляти та генерувати текст у гармонії з іншими сенсорними модальностями. Включаючи візуальні, аудіо чи інші сенсорні дані, модель може створювати більш всебічні та відповідні результати, розширюючи спектр її застосування та сприяючи підвищенню рівня комунікації та розуміння. Вона відзначається багатомовною підтримкою, що сприяє генерації тексту та комунікації на численних мовах. Flan-UL2 може розуміти та генерувати текст на різних мовах, тим самим долаючи бар'єри та сприяючи міжмовній комунікації та розумінню.

2.1.6 BLOOM

Модель BLOOM, розроблена командою BigScience Workshop [36]. Ця ЛЛМ має 176 мільярдами параметрів, що свідчить про її відносно (якщо порівняти з БАРД або ГПТ) передові можливості. Завдяки такій кількості параметрів, BLOOM ефективно розпізнає складні шаблони мовлення, що дозволяє

створювати точний та відповідний контексту текст. Відрізняється від інших моделей своїми унікальними функціями. Його можливості оптимізації тексту допомагають користувачам створювати тексти, налаштовані під конкретні параметри, як-от стиль, тон або читабельність. Використання цієї моделі спрощено завдяки інтуїтивно зрозумілому інтерфейсу, розробленому для легкої інтеграції. Користувачі можуть використовувати цей інтерфейс для додавання функцій оптимізації мови у свої програми, платформи чи творчі проекти, використовуючи потенціал моделі для задоволення своїх потреб. BLOOM також має мультимодальні можливості, що дозволяють обробляти текст разом з іншими видами інформації. Інтегруючи візуальні, аудіо чи інші дані, модель покращує генерацію тексту, створюючи більш комплексні та відповідні за контекстом результати.

Основна мета навчання цієї моделі полягає в удосконаленні та оптимізації моделювання мови. Для навчання моделі використовуються дані з різноманітних текстових матеріалів, включаючи літературу, статті та інший відповідний контент. Завдання навчання зосереджені на покращенні та уточненні генерації тексту. Навчаючись на цьому різноманітному наборі даних, BLOOM досягає глибокого розуміння нюансів мови. Використовуючи це розуміння, він оптимізує генерацію тексту відповідно до конкретних цілей або критеріїв. Ця модель ефективно оптимізує генерацію тексту на багатьох мовах.

2.1.7 Висновки

Зведемо всі отримані дані до таблиці, щоб «наглядно» побачити всі ключові відмінності:

Таблиця 2.1 – Порівння ЛЛМ моделей

Параметр/Модель	**GPT-4**	**Bard**	**LaMDA**	**Flan-UJ2**	**BLOOM**
Параметри	1,5 трильйона	1,6 трильйона	1,2 трильйона	20 мільярдів	176 мільярдів

Навчальні дані	Корпус подібний до Web Text	Корпус подібний до Web Text	Корпус подібний до Web Text	Публічно доступні дані	Багатомовний веб-корпус
Навчальні цілі	Моделювання мови	Моделювання мови	Моделювання мови	Суміш денойзерів (MoD)	Не вказано
Особливі можливості	Покращений дизайн підказок	Покращений дизайн підказок	Покращений дизайн підказок	Універсально ефективний для задач НМП	Відкритий доступ, багатомовність
Як отримати доступ	Через API OpenAI	Через робочу область Google	Потрібен додаток	Не вказано	Відкритий код
Хто випустив	OpenAI	Google	Meta AI	Google	Майстерня Big Science
Набір даних для навчання	Корпус подібний до Web Text	Корпус подібний до Web Text	Корпус подібний до Web Text	Публічно доступні дані	Багатомовний веб-корпус
Мультимодальні можливості	Ні	Ні	Ні	Так	Так
Підтримка багатьох мов	Так	Обмежено	Так	Так	Так

Ця таблиця дає нам зрозуміти, що кожна з цих моделей мови - GPT-4, Bard, LLaMA, Flan-UL2 та BLOOM - має свої особливі характеристики та сильні сторони. Вони різняться за кількістю параметрів, джерелами даних для навчання, цілями навчання, унікальними функціями та доступністю. Більшість з них навчаються на текстових корпусах, подібних до WebText, і фокусуються на моделюванні мови. Flan-UL2 вирізняється своїм підходом до навчання з використанням суміші денойзерів, що робить її універсальною в багатьох задачах обробки природної мови. А BLOOM пропонує себе як відкриту для всіх модель, що підтримує багато мов [30].

Проаналізувавши - бачимо, що для реалізації системи за поставленим завданням, найкраще нам підходить система «GPT-4». У неї є найвагоміша зв'язка - вона маж одну з найбільших кількість параметрів (другу після Клода) та доступ через АПІ, який потрібен нам для створення додатка для системи.

2.2 Вибір фреймворків

2.2.1 Фронтенд частина

Тепер потрібно обрати фреймворк, за допомогою якого створимо інтерфейс системи. Зупинимо свій вибір на React, або React.js [37]. Це бібліотека JavaScript для створення користувацьких інтерфейсів. Вона була розроблена компанією Facebook і здобула велику популярність серед веб-розробників завдяки своїм перевагам та особливостям. Ось декілька ключових характеристик React та причин, чому він вважається гарним інструментом для веб-розробки:

✓ Компонентна архітектура: React сприяє розробці веб-додатків за допомогою компонентів - малих, незалежних блоків коду, які можуть бути легко створені, перевикористані та комбіновані. Це полегшує організацію коду і підтримку великих проектів.

✓ Віртуальний DOM: React використовує віртуальний DOM (Document Object Model) для ефективного оновлення сторінок інтерфейсу. Він автоматично визначає зміни в даних і мінімізує кількість маніпуляцій з реальним DOM, що робить додатки на React швидкими і ефективними.

✓ Односторінковий додаток (SPA) і роутинг: React дозволяє легко створювати SPA, де перехід між сторінками відбувається без перезавантаження сторінки. Бібліотеки, такі як React Router, допомагають забезпечити роутинг в додатках React.

✓ Розширюваність та спільнота: У React існує велика та активна спільнота розробників, яка надає безліч розширень (або бібліотек) і плагінів, які допомагають вирішувати різні завдання в розробці веб-додатків.

✓ Підтримка серверної рендерінгу: React може використовуватися для створення додатків з серверним рендерінгом, що полегшує SEO і поліпшує швидкість завантаження сторінок.

✓ Офіційні інструменти та документація: Facebook і активна спільнота розробників надають високоякісну документацію, інструменти для розробки та відлагодження, такі як React DevTools, що полегшують роботу з бібліотекою.

Загалом, React - це потужна бібліотека, яка дозволяє розробникам створювати сучасні та ефективні веб-додатки з використанням компонентної архітектури та інших передових технологій. Він є популярним вибором для розробки інтерфейсів в багатьох веб-проектах.

2.2.2 Бекенд

З бекенд частиною не все так просто. Нам потрібно декілька систем:

1. Модуль для збереження резюме.
2. Модуль для аналізу резюме
3. Модуль для управління користувачами та інформацією про резюме.

2.2.2.1 Модуль для збереження резюме

Для опрацювання та збереження резюме оберемо Express.js. Чому саме його? Приведемо аргументацію. Express.js є одним з найпопулярніших вебфреймворків для Node.js, але важливо зазначити, що сам по собі Express не спеціалізується на зберіганні файлів. Замість цього, він використовується для створення веб-серверів та API [38, 39]. Проте, ми можемо використовувати Express разом із іншими пакетами та бібліотеками для управління збереженням файлів. Ось декілька аргументів на користь використання Express.js у цьому контексті:

✓ Інтеграція з Node.js: Оскільки Express працює на Node.js, він легко інтегрується з файловою системою Node.js для зберігання та обробки файлів.

✓ Використання Middleware для Завантаження Файлів: Ми можемо використовувати middleware, таке як multer, для обробки завантаження файлів. multer дозволяє легко обробляти multipart/form-data, яке часто використовується для завантаження файлів.

✓ Гнучкість у Маршрутизації: Express дозволяє легко налаштовувати маршрути для завантаження та доступу до файлів, надаючи вам контроль над тим, як файли зберігаються та обробляються.

✓ Широка Спільнота та Підтримка: Завдяки своїй популярності, існує багато ресурсів, навчальних матеріалів, та готових рішень, які можна використовувати для роботи з файлами в Express.

✓ Інтеграція з Базами Даних та Хмарними Сховищами: Express можна інтегрувати з різними базами даних та хмарними сховищами для ефективного зберігання файлів.

✓ Безпека: Ми можемо використовувати різні пакети безпеки з Express для забезпечення безпечного зберігання та передачі файлів.

✓

2.2.2.2 Модуль аналізу резюме

Для аналізу резюме нам потрібне підключення через режим апі до нашої ГПТ-4. Для цього вже створено декілька бібліотек: для Python та для Node.js. Нам підходить та бібліотека, що працює на «ноді», оскільки краще, якщо система буде працювати в спорідненому середовищі.

2.2.2.3 Модуль для управління користувачами та інформацією про резюме

Для управління користувачами та інформацією про резюме нам потрібно теж обрати систему. Проаналізувавши фреймворки обираємо ASP.NET Core [40]. Наведемо аргументацію:

1. Інтегрована система ідентифікації: ASP.NET Core поставляється з вбудованою системою ідентифікації, яка підтримує реєстрацію користувачів,

вхід, управління профілями та захист паролів. Це полегшує реалізацію складних вимог щодо аутентифікації та авторизації.

2. Підтримка ролей та прав Користувачів: ASP.NET Core дозволяє легко створювати та керувати ролями користувачів, що є ключовим для управління доступом до різних частин додатку.

3. Розширюваність та гнучкість: Система ідентифікації в ASP.NET Core є високо налаштовуваною, дозволяючи вам розширювати стандартні функції та інтегрувати сторонні служби.

4. Безпека: ASP.NET Core надає різноманітні функції безпеки, такі як захист від міжсайтового скриптингу (XSS), міжсайтової підробки запиту (CSRF) та SQL ін'єкцій. Це допомагає забезпечити безпечне управління даними користувачів.

5. Підтримка JWT та токенів: ASP.NET Core підтримує JSON Web Tokens (JWT) та інші механізми токенів для API аутентифікації, що є важливим для сучасних веб-додатків і мобільних додатків.

6. Висока продуктивність та масштабованість: ASP.NET Core відомий своєю високою продуктивністю та масштабованістю, що є важливим для додатків з великою кількістю користувачів.

7. Підтримка різних даних зберігання: ASP.NET Core може легко інтегруватися з різними базами даних, що дає гнучкість у виборі засобів зберігання інформації про користувачів.

8. Спільнота та ресурси: Існує велика спільнота розробників та багато ресурсів для навчання та підтримки, що може бути корисним при розробці та підтримці нашого проекту.

Користуючись ASP.NET Core, ми отримуємо міцну основу для створення надійного та безпечного веб-додатку з ефективним управлінням користувачами.

2.3 Вибір середовища розробки

Для програмного створення нашої системи, її компіляції та тестування будемо використовувати середовище розробки VSCode [41].

Visual Studio Code (VS Code або VSCode) - це безкоштовне і відкрите середовище розробки (IDE) розроблене компанією Microsoft. Воно стало надзвичайно популярним серед програмістів та розробників завдяки численним перевагам. Ось декілька ключових особливостей та переваг Visual Studio Code:

✓ Безкоштовність та відкритість: VS Code доступний для завантаження та використання абсолютно безкоштовно. Він також є відкритим програмним забезпеченням, що означає, що користувачі можуть вносити внески у його розвиток та створювати розширення.

✓ Кросплатформенність: VS Code підтримує операційні системи Windows, macOS і Linux. Це робить його ідеальним інструментом для розробки незалежно від вашої операційної системи.

✓ Зручний редактор коду: Він має потужний редактор коду з підсвічуванням синтаксису, автодоповненням, функціями пошуку та заміни, а також багатьма іншими корисними функціями, які сприяють зручності розробки.

✓ Розширюваність: VS Code підтримує велику кількість розширень, що дозволяє налаштовувати та розширювати функціональність редактора під власні потреби. Є розширення для різних мов програмування, фреймворків та інструментів розробки.

✓ Інтеграція з іншими інструментами: VS Code легко інтегрується з іншими популярними інструментами розробки, такими як Git для керування версіями, віртуальні середовища, інструменти для роботи з контейнерами і багато інших.

✓ Спільнота та підтримка: Visual Studio Code має велику та активну спільноту користувачів, що означає, що ви можете знайти багато корисних ресурсів, документацію та підтримку для вирішення будь-яких проблем.

✓ Платформа розробки: Visual Studio Code підтримує різні мови програмування, включаючи JavaScript, Python, Java, C++, Ruby, PHP та багато інших,

що робить його універсальним інструментом для розробників різних напрямків.

Загалом, Visual Studio Code є потужним та зручним інструментом для розробників. Він значно сприяє збільшенню продуктивності та забезпечує зручні умови для написання якісного коду, тому обираємо саме його.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Опис програмної реалізації

3.1.1 Менеджер управління резюме

3.1.1.1 Сервер

Для того, щоб створити менеджер управління резюме, щоб ми могли їх завантажувати у систему та прив’язувати за конкретними користувачами, потрібно створити сам сервер (Рисунок 3.1).

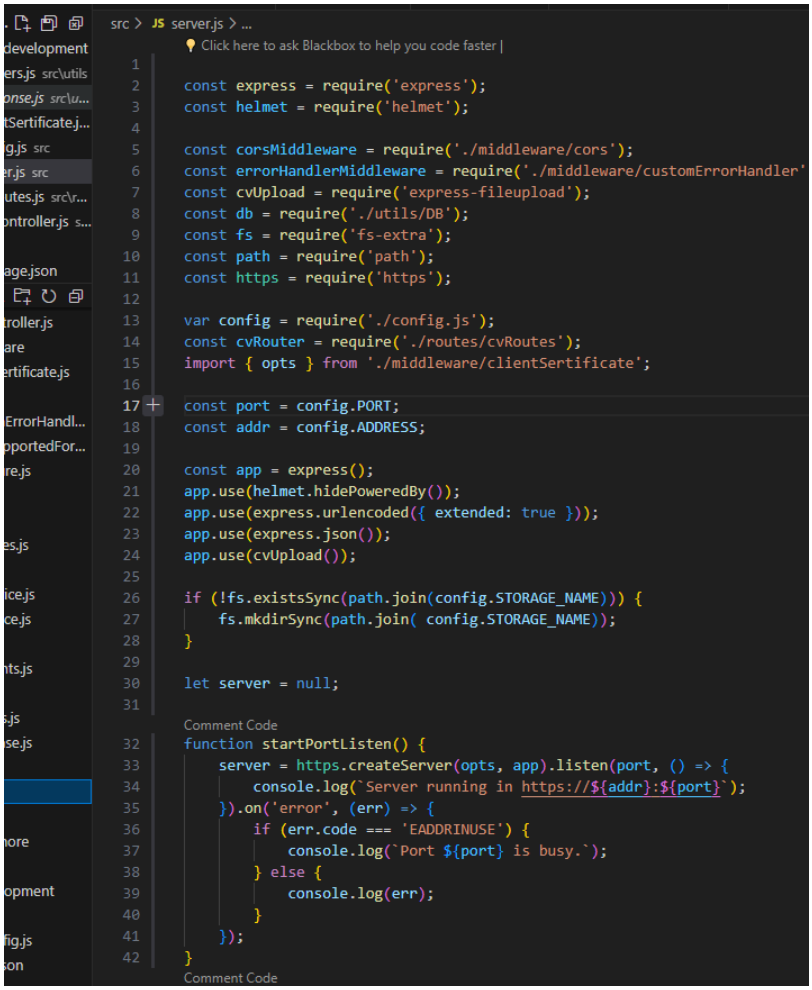


Рисунок 3.1- Створення серверу

3.1.1.2 Контролер

Для того, щоб сервер працював штатно, нам потрібно декілька важливих деталей. Перше на потрібно, щоб працювала обробка функцій нашої системи, тобто нам потрібно, щоб наші резюме зберігалися, видалялися, їх можна було замінити. Також нам потрібно, щоб після збереження ми могли отримати їх назад, крім цього нам потрібно реалізувати ці фунццї у контролера .

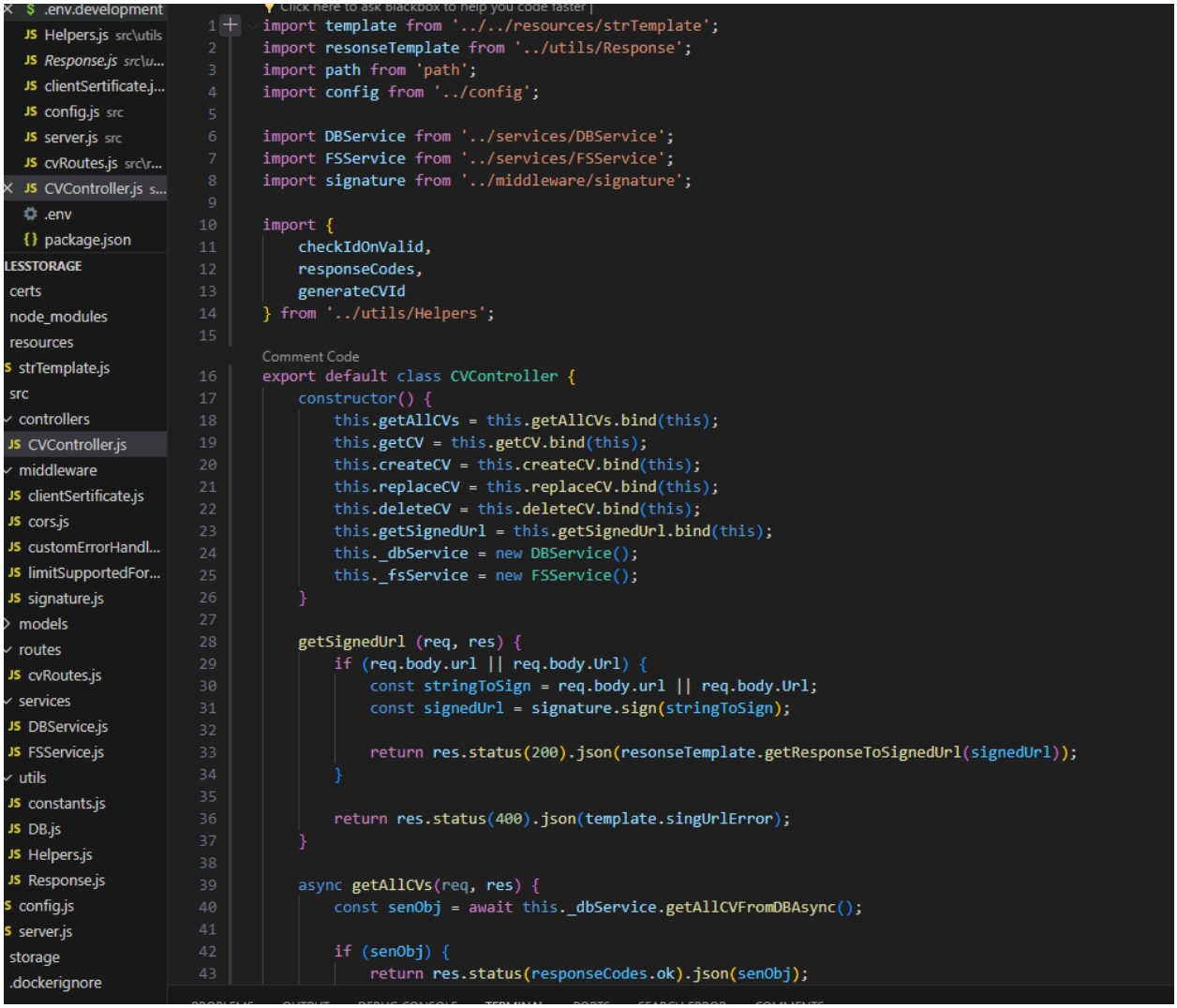


Рисунок 3.2 - Створення контролеру для управління резюме

3.1.1.3 Сервіси

Для того, щоб виконувати логіку методів, які записані до контролера, потрібні сервіси, які виконують спілкування з файловою системою для

збереження резюме фізично на диску та базою даних, де дані про ці резюме зберігаються.

3.1.1.3.1 Сервіс підключення і роботи з базою даних

Для роботи з БД створено DBService – саме тут описані всі методи для збереження, видалення та змінення даних про резюме.

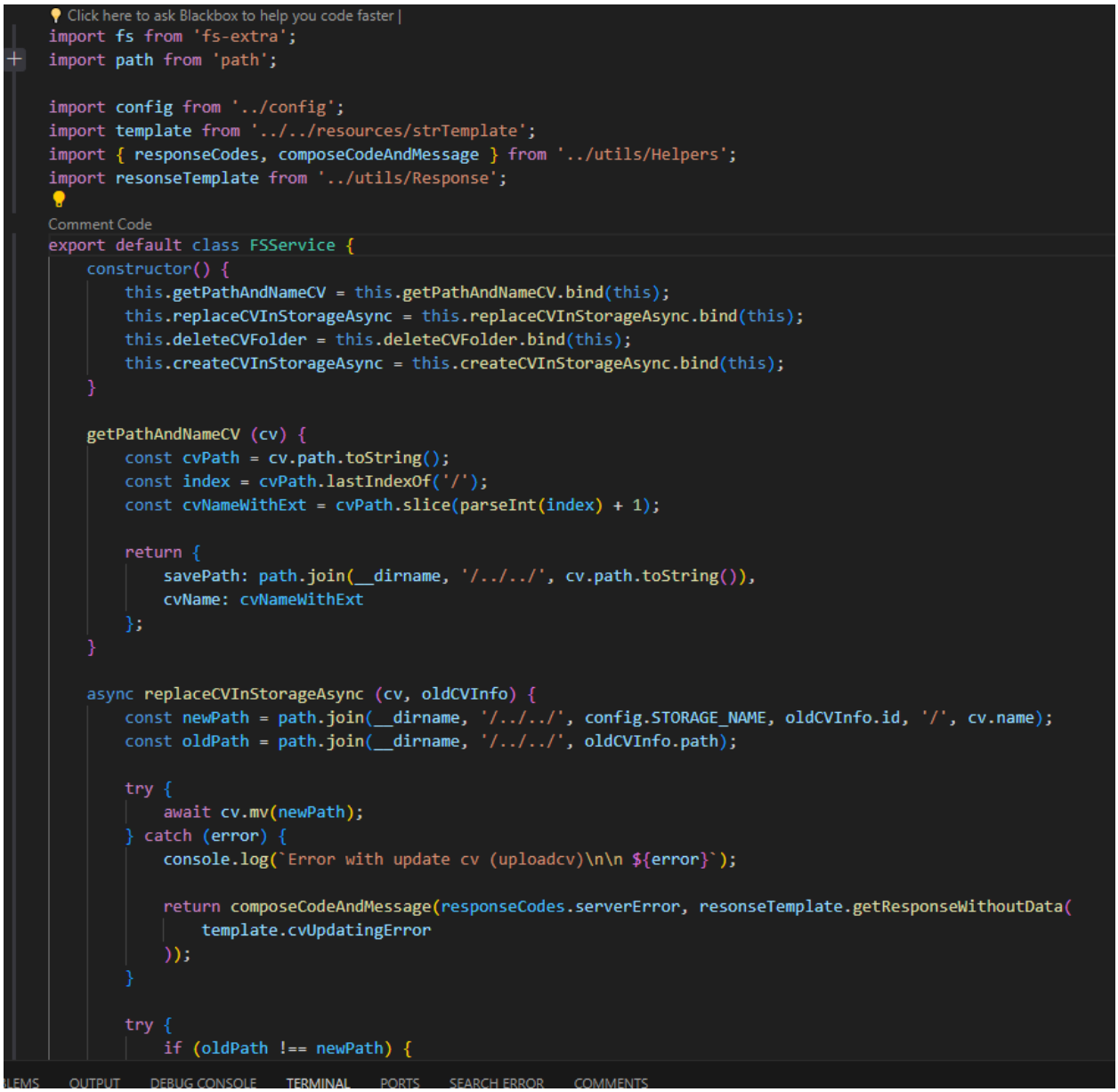
```
import template from '../../resources/strTemplate';
import { responseCodes, composeCodeAndMessage } from '../utils/Helpers';
import resonseTemplate from '../utils/Response';

Comment Code
export default class DBService {
    constructor() {
        this.getAllCVFromDBAsync = this.getAllCVFromDBAsync.bind(this);
        this.getCVFromDBByIdAsync = this.getCVFromDBByIdAsync.bind(this);
        this.createCVInDBAsync = this.createCVInDBAsync.bind(this);
        this.replaceCVInDBAsync = this.replaceCVInDBAsync.bind(this);
        this.deleteCVInDBAsync = this.deleteCVInDBAsync.bind(this);
    }

    async getAllCVFromDBAsync (){
        const cvs = await CV.find({});
        if (cvs) {
            cvs.forEach((cv) => {
                const cvPpath = cv.path;
                const index = cvPpath.lastIndexOf('/');

                cv.path = cvPpath.slice(parseInt(index) + 1);
            });

            return cvs;
        }
        return undefined;
    }

    async getCVFromDBByIdAsync (receivedCVId) {
        try {
            const findedCV =  await CV.findById(receivedCVId);
            return findedCV;
        } catch (error) {
            console.log(error);
            return undefined;
        }
    }

    async createCVInDBAsync (dbCVPath, bdCVId) {
        try {
            const bdCV = new CV();
            bdCV._id = bdCVId;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS

Рисунок 3.3 – DBService

3.1.1.3.2 Сервіс для роботи з файловою системою

А цей сервіс нам потрібен для того, щоб працювати з файлами у локальній системі – зберігати, видаляти, заміняти наші резюме.



```
Click here to ask Blackbox to help you code faster |
import fs from 'fs-extra';
import path from 'path';

import config from '../config';
import template from '../../resources/strTemplate';
import { responseCodes, composeCodeAndMessage } from '../utils/Helpers';
import resonseTemplate from '../utils/Response';

Comment Code
export default class FSService {
    constructor() {
        this.getPathAndNameCV = this.getPathAndNameCV.bind(this);
        this.replaceCVInStorageAsync = this.replaceCVInStorageAsync.bind(this);
        this.deleteCVFolder = this.deleteCVFolder.bind(this);
        this.createCVInStorageAsync = this.createCVInStorageAsync.bind(this);
    }

    getPathAndNameCV (cv) {
        const cvPath = cv.path.toString();
        const index = cvPath.lastIndexOf('/');
        const cvNameWithExt = cvPath.slice(parseInt(index) + 1);

        return {
            savePath: path.join(__dirname, '/../../', cv.path.toString()),
            cvName: cvNameWithExt
        };
    }

    async replaceCVInStorageAsync (cv, oldCVInfo) {
        const newPath = path.join(__dirname, '/../../', config.STORAGE_NAME, oldCVInfo.id, '/', cv.name);
        const oldPath = path.join(__dirname, '/../../', oldCVInfo.path);

        try {
            await cv.mv(newPath);
        } catch (error) {
            console.log(`Error with update cv (uploadcv)\n\n ${error}`);

            return composeCodeAndMessage(responseCodes.serverError, resonseTemplate.getResponseWithoutData(
                template.cvUpdatingError
            ));
        }

        try {
            if (oldPath !== newPath) {
LEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR  COMMENTS
```

Рисунок 3.4 – FSService

3.1.1.4 Маршрутизація

Для того, щоб користувачі могли мати доступ до методів контролера які, як раз таки виконують бізнес-логіку, потрібно прокласти маршрути (Рисунок 3.5) до цих кінцевих точок даної частини системи.

Рисунок 3.5 – Маршрути

3.1.2 Менеджер управління даними про користувачів та резюме

Сервіс, який виконує управління юзерами та прив’язаними до них резюме має досить велику кількість файлів та складну структуру. Опишемо лише ключові частини.

Веб-апі скаладається з двох глобальних частин (Рисунок 3.6): перша частина відповідає за роботу з БД. Саме тут описані класи, методи та контексти для збереження, змінення та відтворення інформації про користувачів системи та резюме, а друга – за управління користувачами, їхніми ролями, правами та зв’язаними сіві.

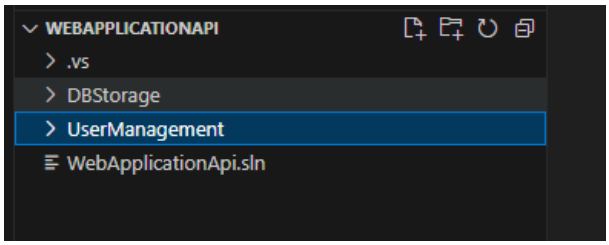


Рисунок 3.6 - Частини веб-апі

3.1.2.1 DBStorage

Робота базою даних налаштована та виконується з допомогою шаблону проектування «Репозиторій». В межах проектування та створення цієї частини системи було створено:

1. Моделі (Рисунок 3.7) – тут описані всі сутності нашої системи.

2. Інтерфейси (Рисунок 3.8) для узагальнення роботи зі сховищем: описані конкретні шаблони класів та методів, що потребують реалізації.

3. Репозиторії (Рисунок 3.9) – саме тут створені та описані класи на основі інтерфейсів, що відповідають за правильну роботу з моделями в БД.

4. Міграції (Рисунок 3.10) для того, щоб описати створення структури таблиць в базах даних, занести відомості про правила, виключення, зв’язки. Це потрібно для полегшення розгортання додатка в інших середовищах.

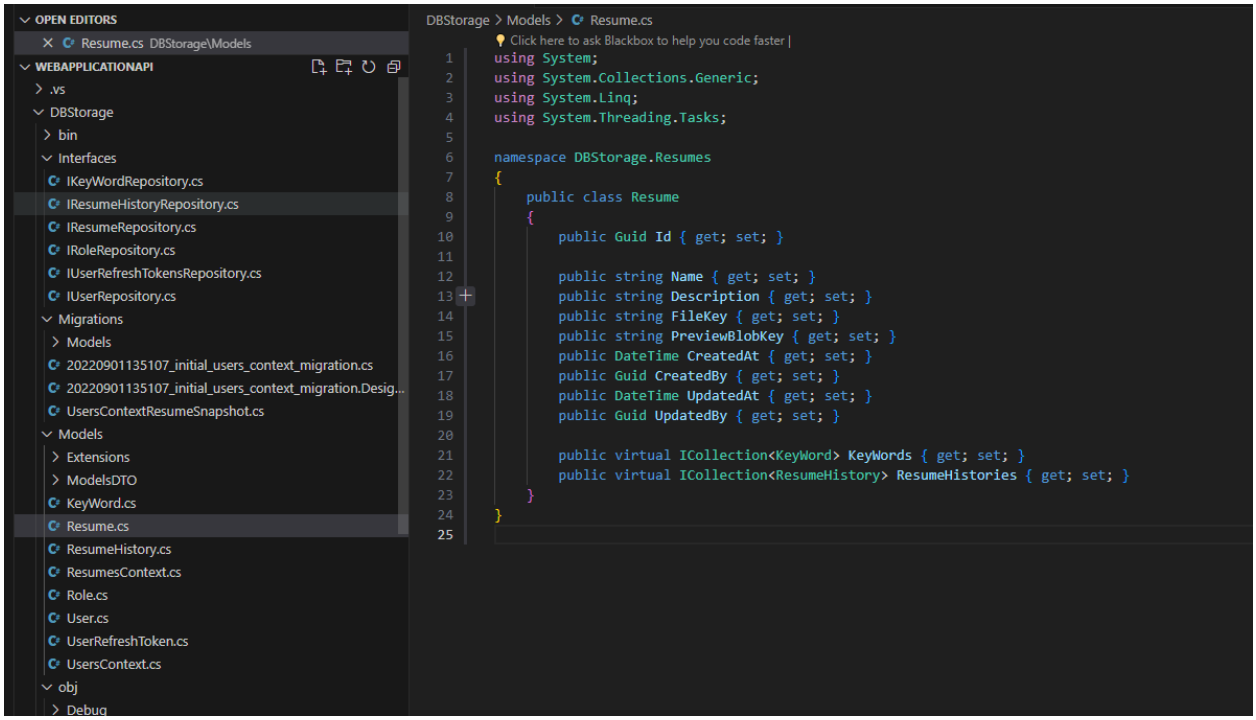


Рисунок 3.7 - Моделі. На рисунку представлений приклад моделі резюме

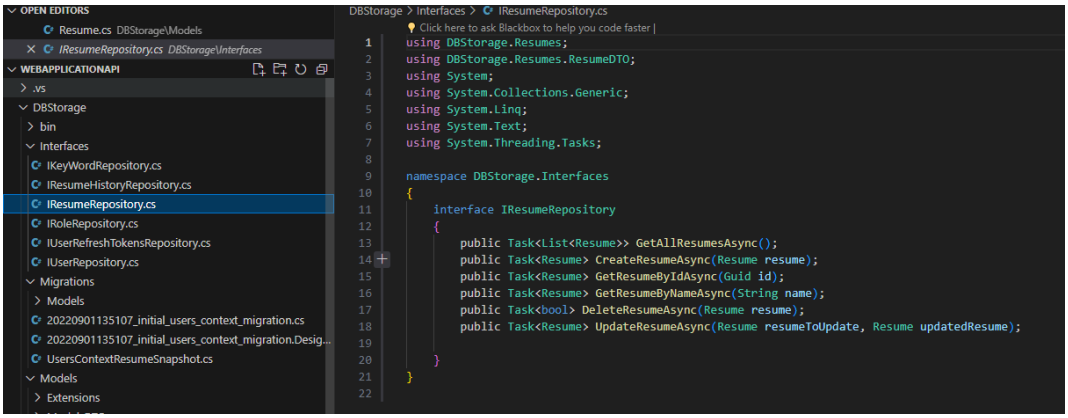


Рисунок 3.8 - Інтерфейси. Приклад на рисунку - інтерфейс репозиторію резюме

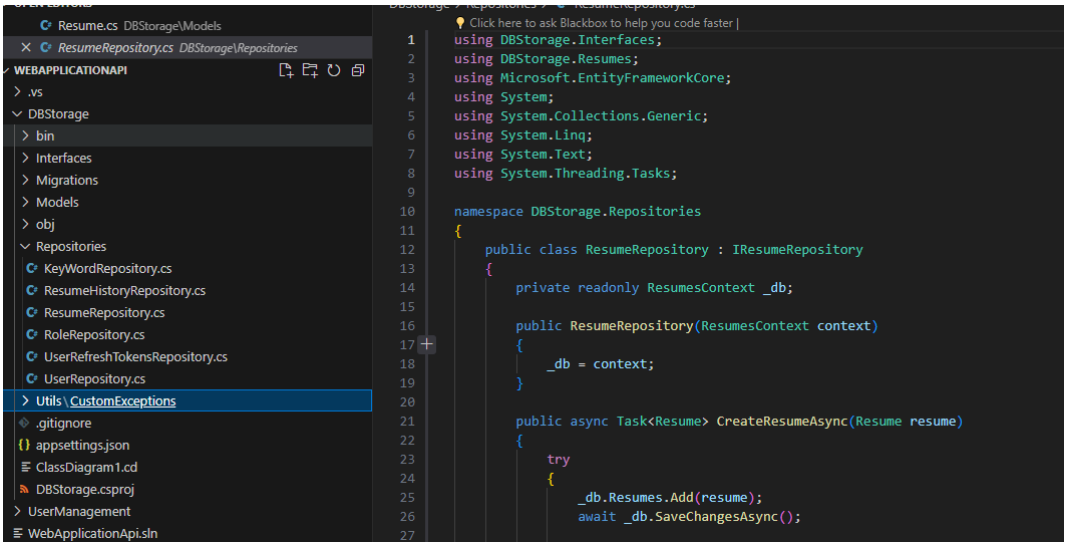


Рисунок 3.9 - Репозиторії. На рисунку репозиторій для резюме

Рисунок 3.10 - Міграції. Початкова міграція для створення таблиці з резюме

3.1.2.2 UserManagement

В межах проектування та створення системи управління користувачами та резюме було створено:

1. Контролери для обробки вхідних HTTP запитів і повернення відповідей до клієнта (Рисунок 3.11).
2. Моделі запитів і відповідей для роботи з аутентифікацією (Рисунок 3.12).
3. Інтерфейс для роботи з JWT (Рисунок 3.13).
4. Репозиторій для роботи з JWT (Рисунок 3.14).

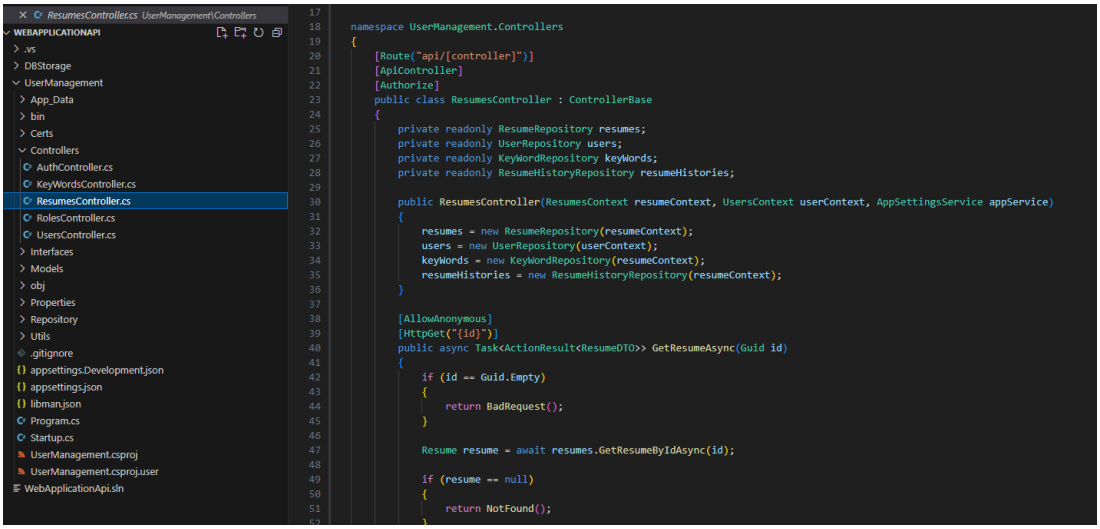


Рисунок 3.11 - Контролери. Контролер роботи з резюме

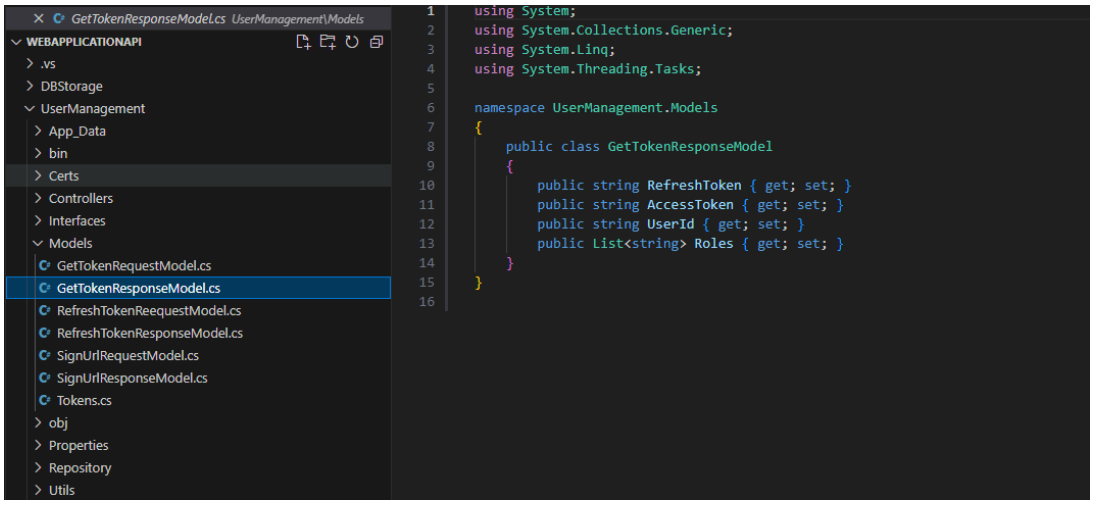


Рисунок 3.12 - Моделі запитів та відповідей. Модель запиту токену

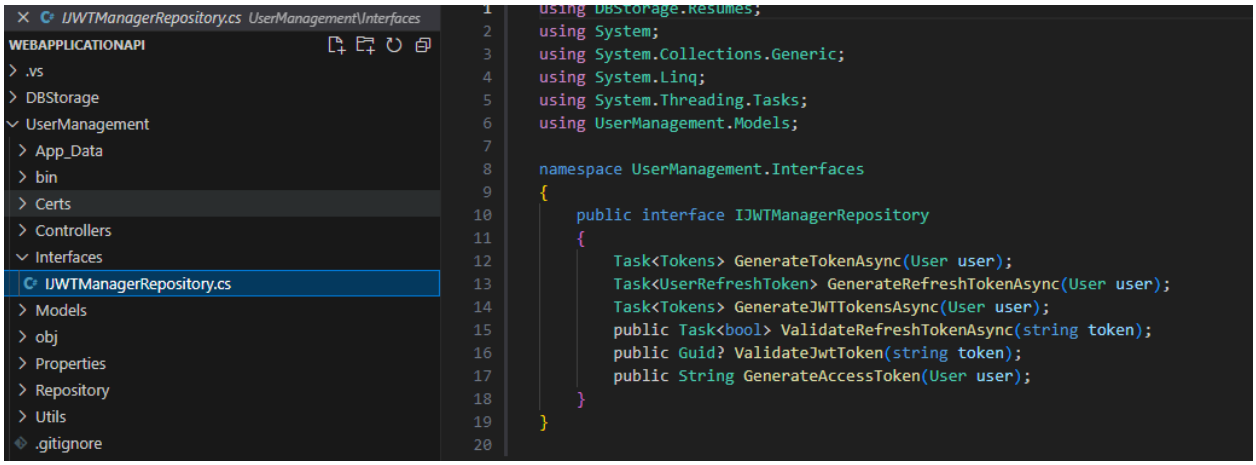


Рисунок 3.13 - Інтерфейс для роботи з JWT

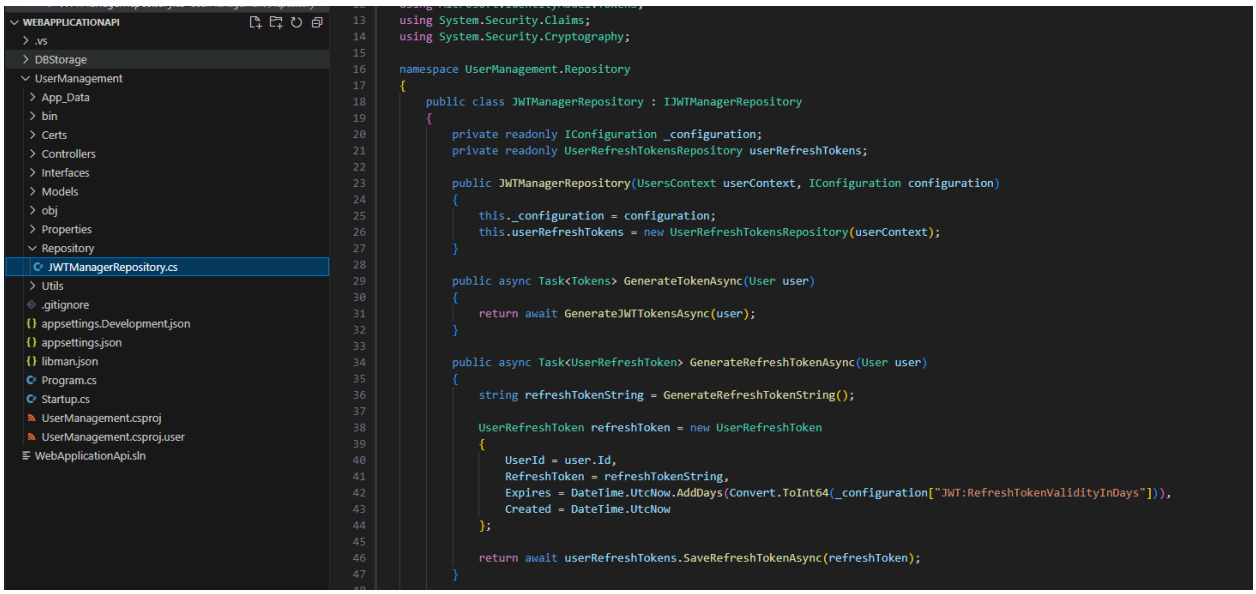


Рисунок 3.14 - Репозиторій для роботи з JWT

3.1.3 Сервіс аналізування резюме

«Серце нашої системи», сервіс, який займається саме аналізом резюме. За допомогою саме цієї частини програмного продукту ми визначаємо відповідність резюме вакансії, робимо рейтинг кандидатів та готуємо список питань на пре-інтерв'ю. Складається з

1. Сервера, який все запускає (Рисунок 3.15).
2. Контролера, який виконує аналізування (Рисунок 3.16).
3. Маршрутів, які описують варіанти мережевого доступу до методів контролера (Рисунок 3.17).
4. Додаткових налаштувань (Рисунок 3.18)

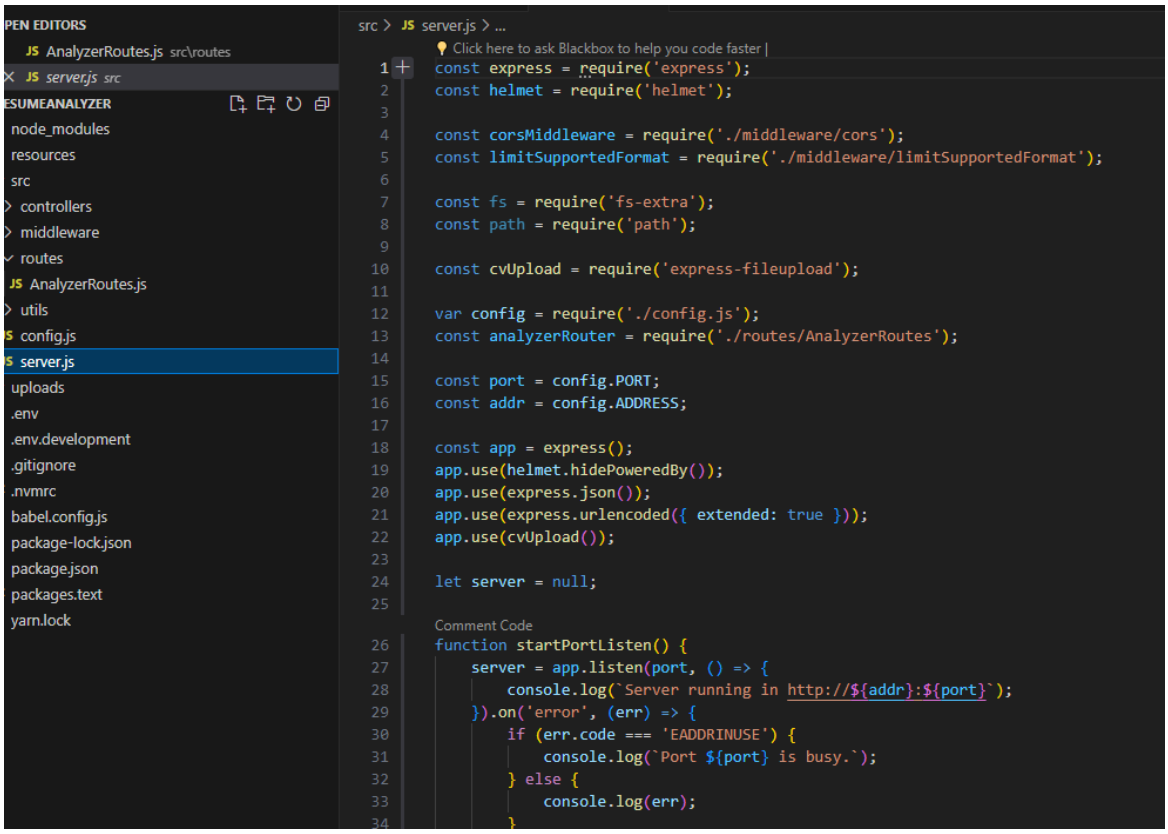


Рисунок 3.15 - Сервер

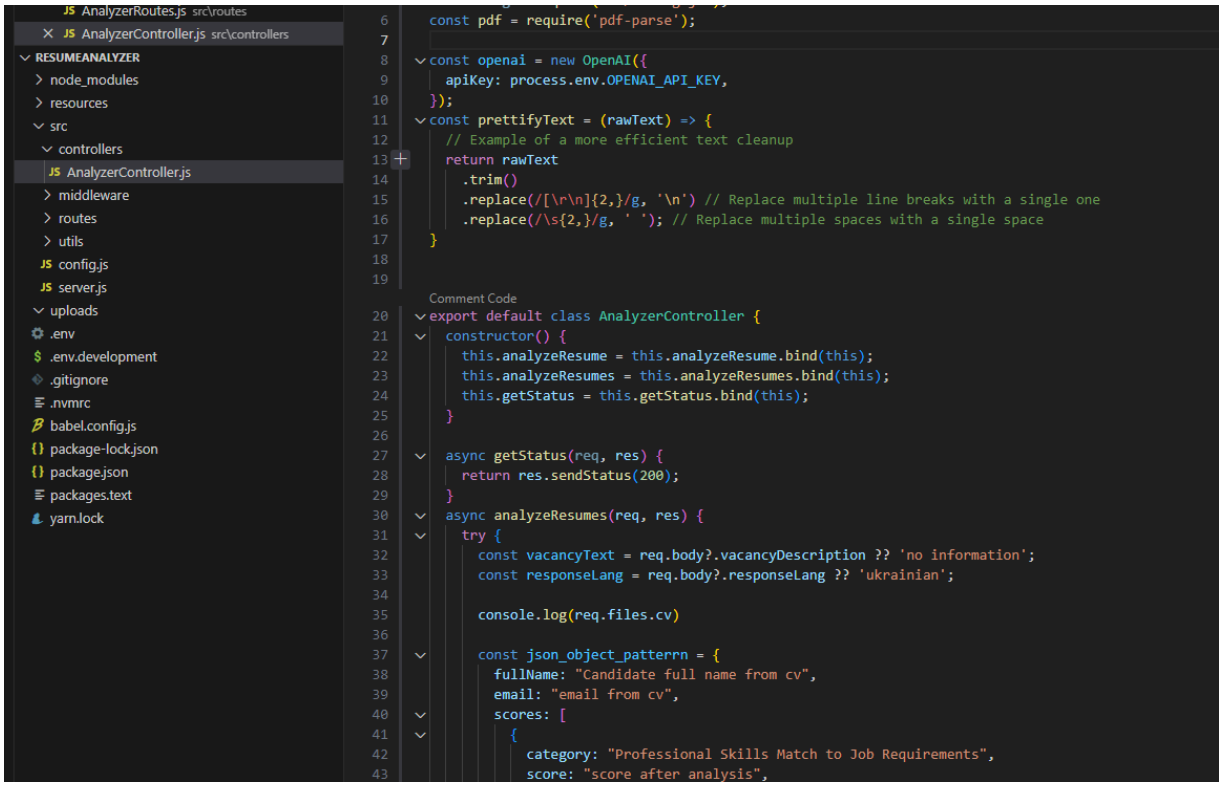


Рисунок 3.16 - Контролер

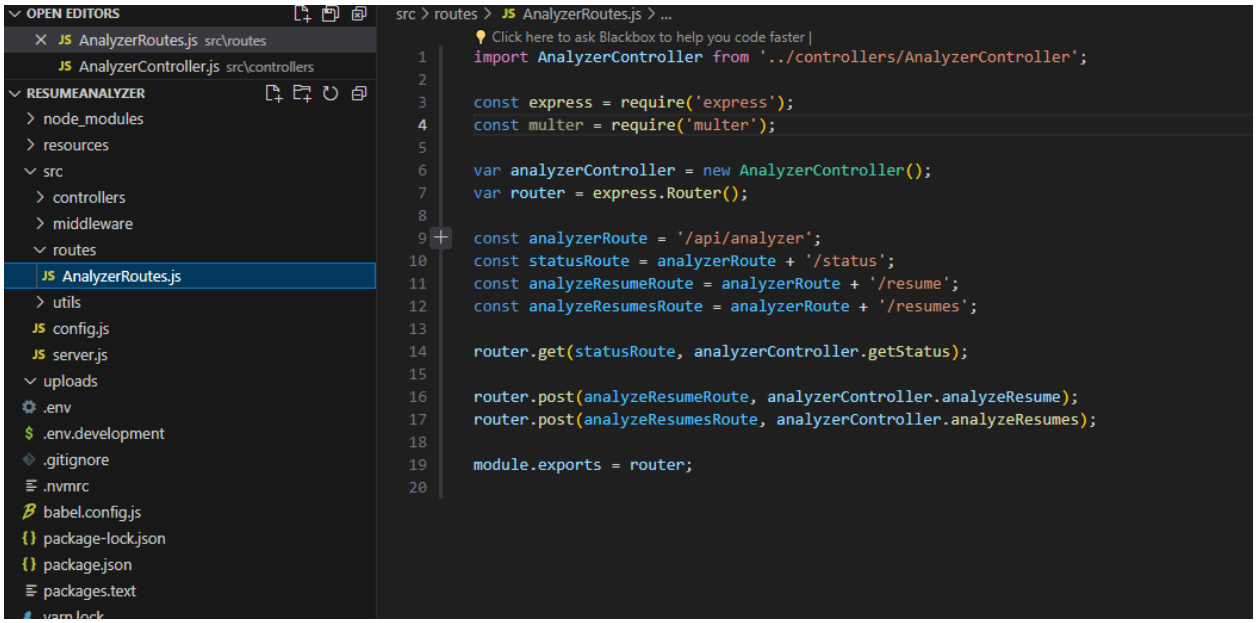


Рисунок 3.17 – Маршрути

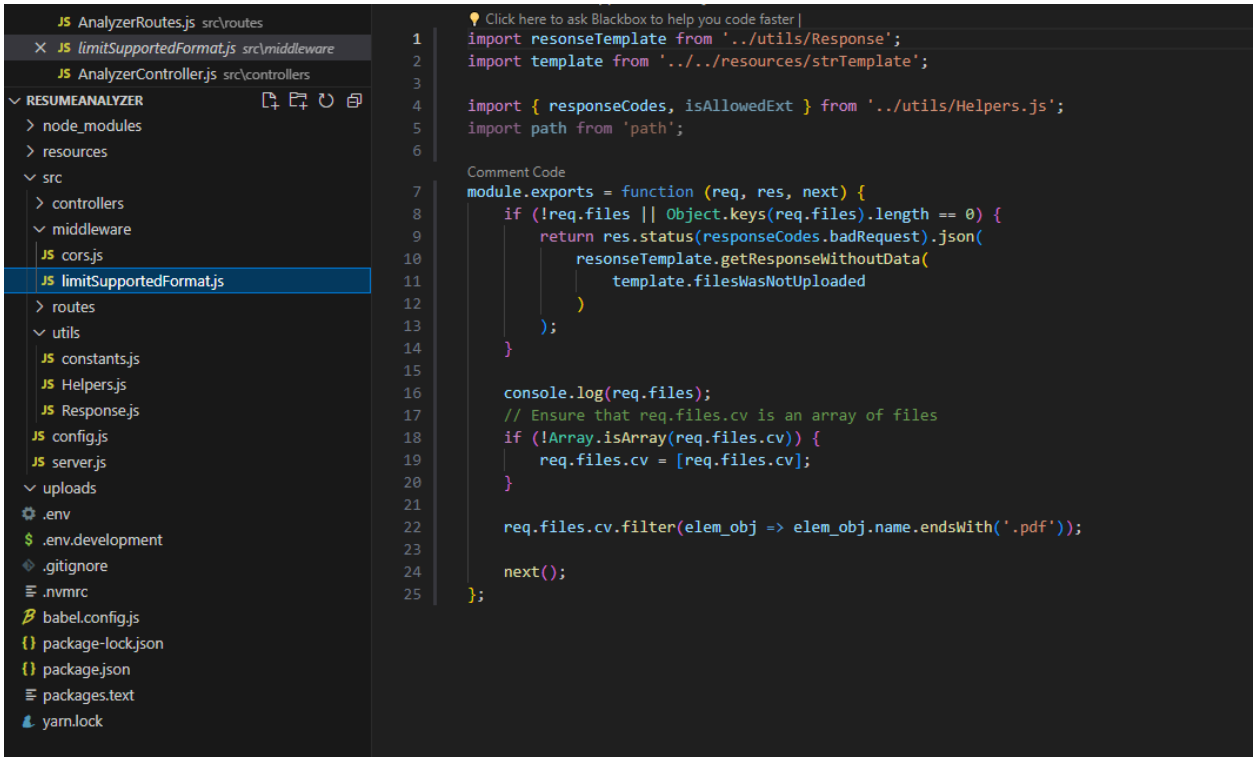


Рисунок 3.18 - Додаткові налаштування. На рисунку модуль управління форматами, що підтримуються аналізатором

3.1.4 Веб інтерфейс

Частина системи, яка відповідає за інтерфейс – найбільша за кодом. Вона включає в себе велику кількість підсистем:

1. Головна функція, яка запускає всі інші підсистеми (Рисунок 3.19).

2. Компоненти – саме тут знаходять всі частини інтерфейсу користувача (Рисунок 3.20).

3. Сервіси – тут описане підключення до бекенд частин системи (Рисунок 3.21).

4. Сховище – менеджер станів (Рисунок 3.22).

Рисунок 3.19 - App.js

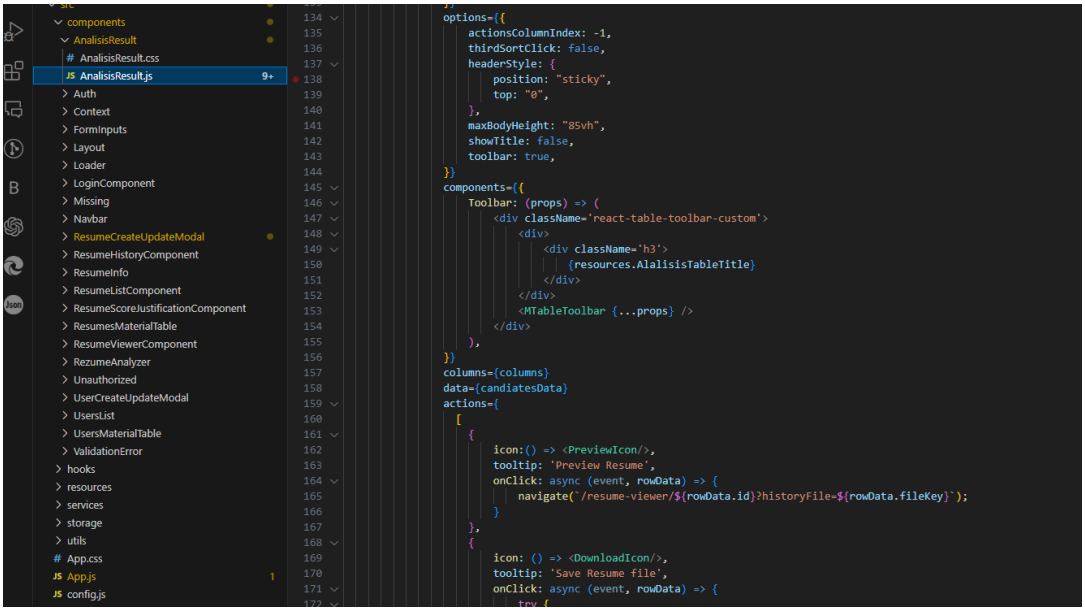


Рисунок 3.20 - Компоненти. На рисунку компонент, який відповідає за виведення результатів аналізу

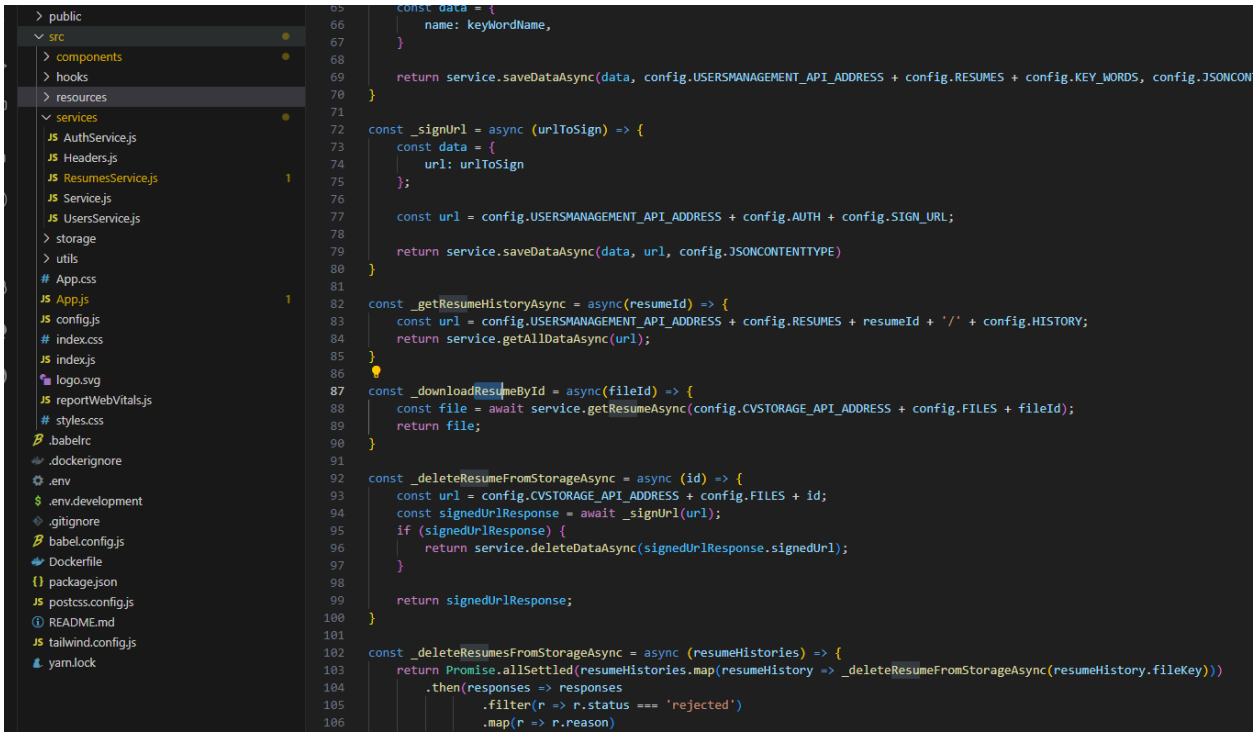


Рисунок 3.21 - Сервіси. На рисунку сервіс, який відповідає за роботу з резюме (CVStorage)

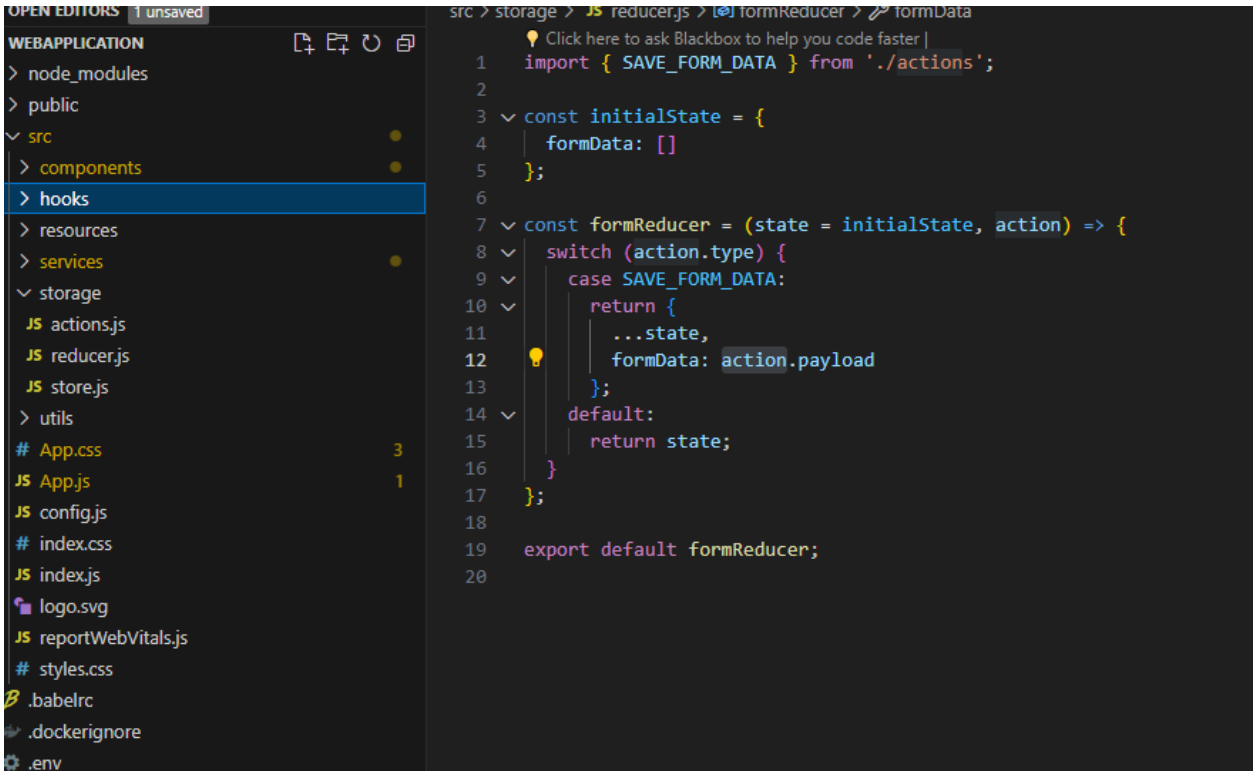


Рисунок 3.22 - Менеджер станів

3.2 Аналіз результатів

Переходимо до практичного використання системи. Для цього запускаємо додаток. На початку роботи з системою користувачу треба авторизуватись (Рисунок 3.23).

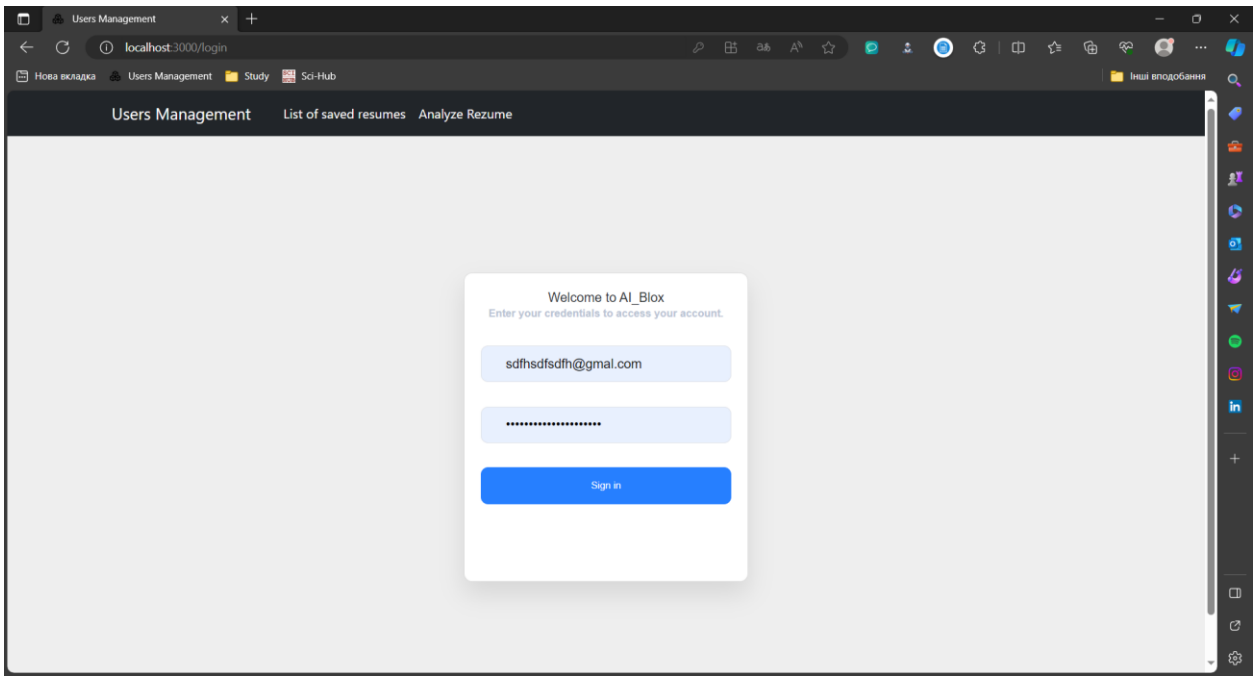


Рисунок 3.23 – Авторизація

Після того, як авторизація успішно виконана, переходимо на сторінку «Analyze Rezume» . Тут бачимо декілька полів для вводу. Перше використовується для опису вакансії (Рисунок 3.24). Сюди в текстовому форматі описуємо характеристики бажаного кандидата, щоб з цими характеристиками потім проаналізувати завантажені резюме. Шукаємо в інтернеті будь-яку вакансію, з текстовим описом. Для того, щоб протестувати створену - систему йдемо на один з найвідоміших ресурсів для пошуку роботи – dou.ua. Там знаходимо першу ж вакансію, яка нам подобається (Рисунок 3.25).

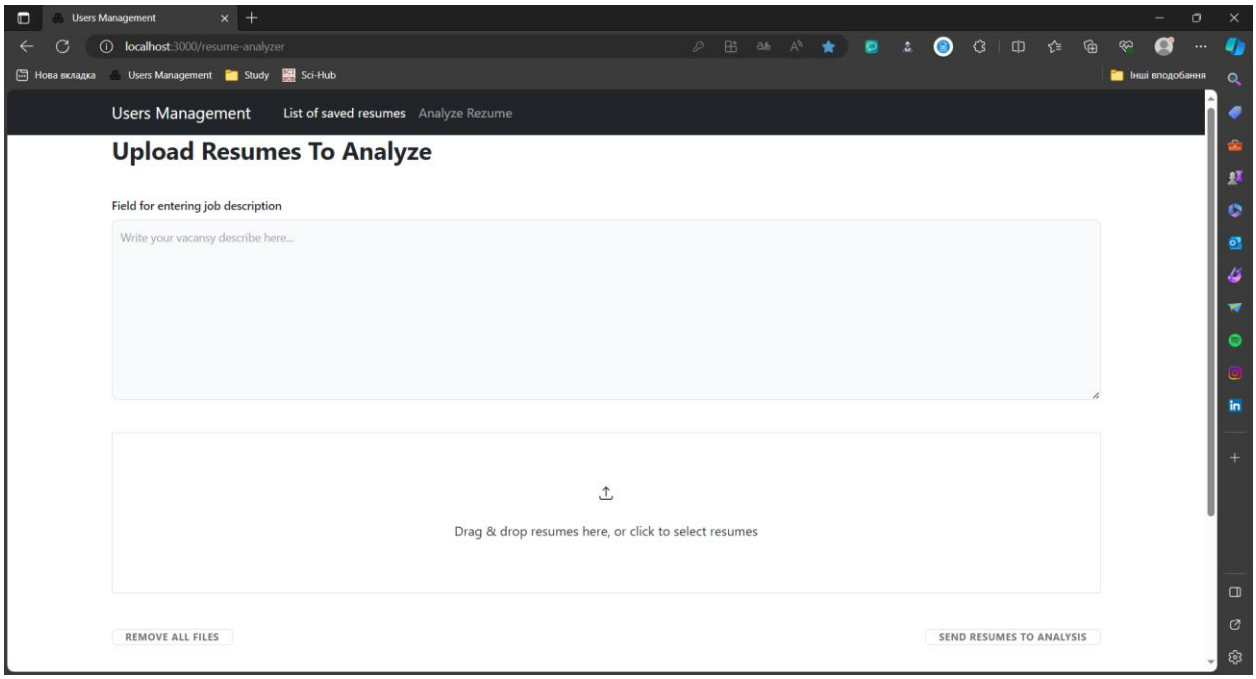


Рисунок 3.24 - Analyze Rezume. Поля вводу

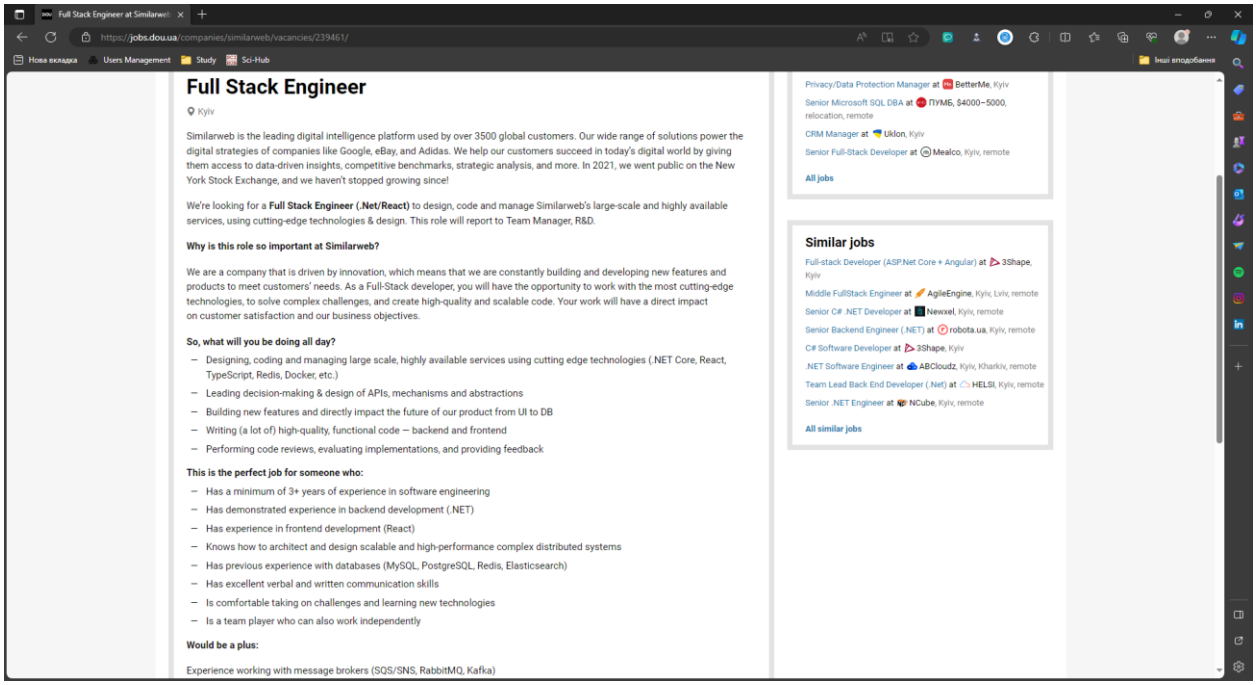


Рисунок 3.25 - Знаходимо вакансію з текстовим описом

Тепер, як вакансія була знайдена (ДОДАТОК Б), копіюємо її собі (Рисунок 3.26). Переходимо до наступного пункту.

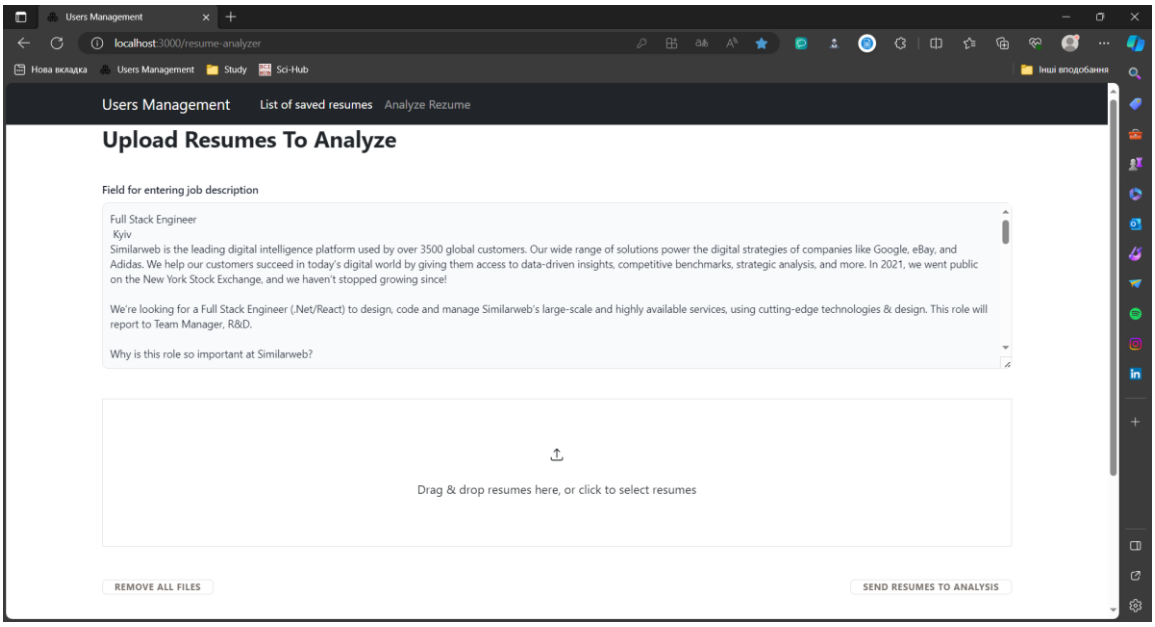


Рисунок 3.26 - Копіюємо знайдений опис у форму

На черзі вибір резюме кандидатів. Знаходимо ресурс з відкритими файлами (найпростіший варіант -лінкедін, там і беремо). Нам потрібні так резюме, щоб і підходили і не підходили до вакансії. Обраємо 4 резюме (ДОДАТОК В), з них одне підходить краще всього, друге трошки менш, а інші двоє – зовсім не підходять. Завантажуємо їх у систему (Рисунок 3.27).

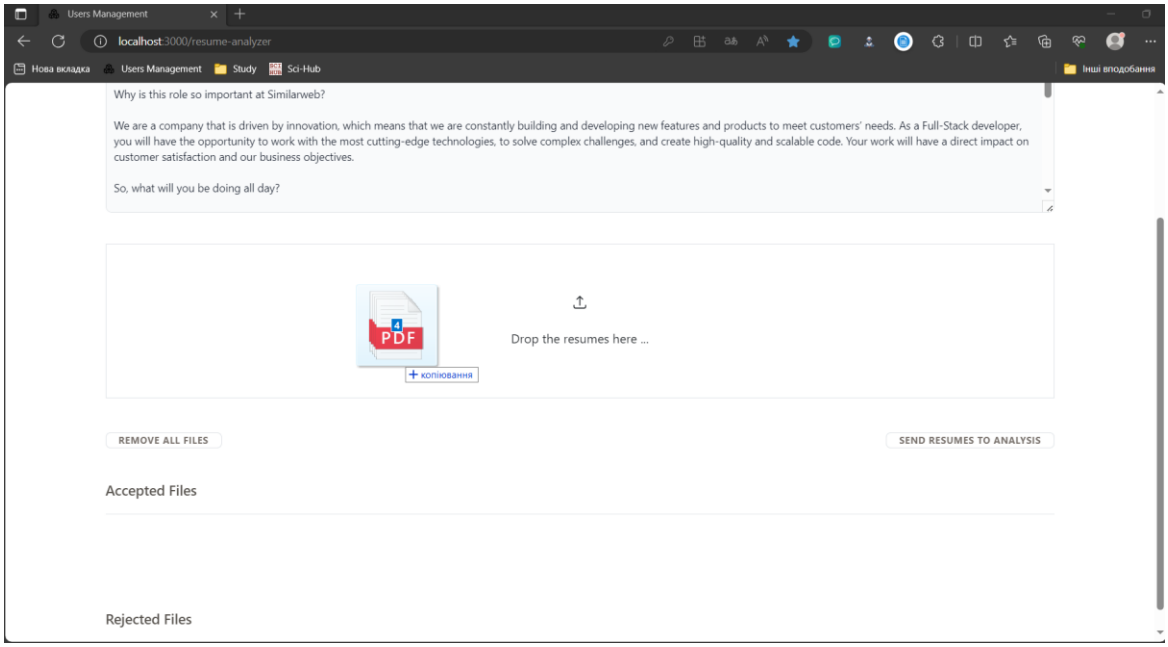


Рисунок 3.27 - Завантажуємо резюме

Завантажені файли автоматично перевіряються на відповідність формату (Рисунок 3.28). Якщо файли валідні – вони відобразяться у списку

затверджених файлів, якщо ні – у списку відмовлених. З обох списків файли можна видаляти та перезавантажувати

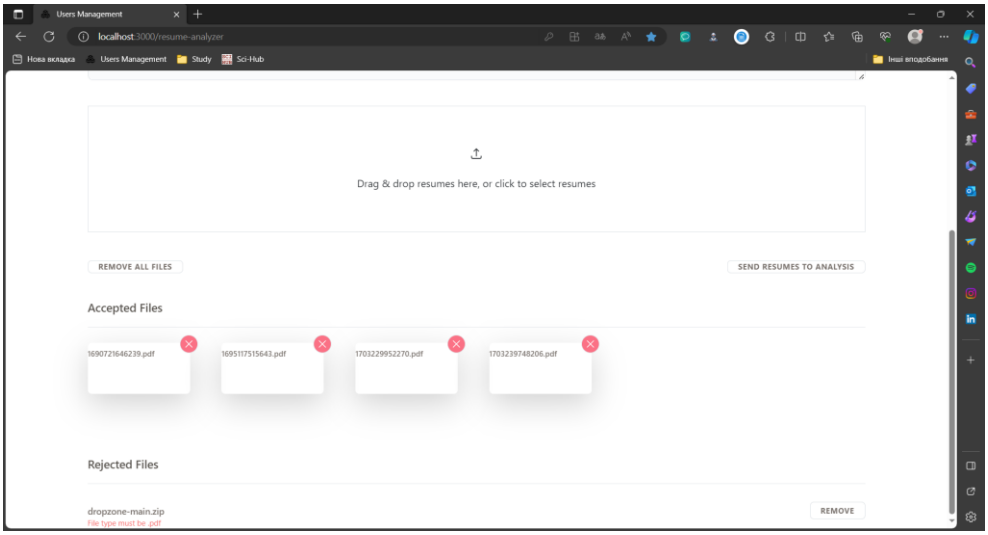


Рисунок 3.28 - Завантажені резюме, валідовані за розширенням

Коли потрібні резюме завантажені і опис вакансії та є можемо надсилати дані на аналіз. Для цього натискаємо кнопку «Send resumes to analysis». Чекаємо, поки файли проаналізуються, виставляться оцінки, підготуються рекомендації та питання (Рисунок 3.29).

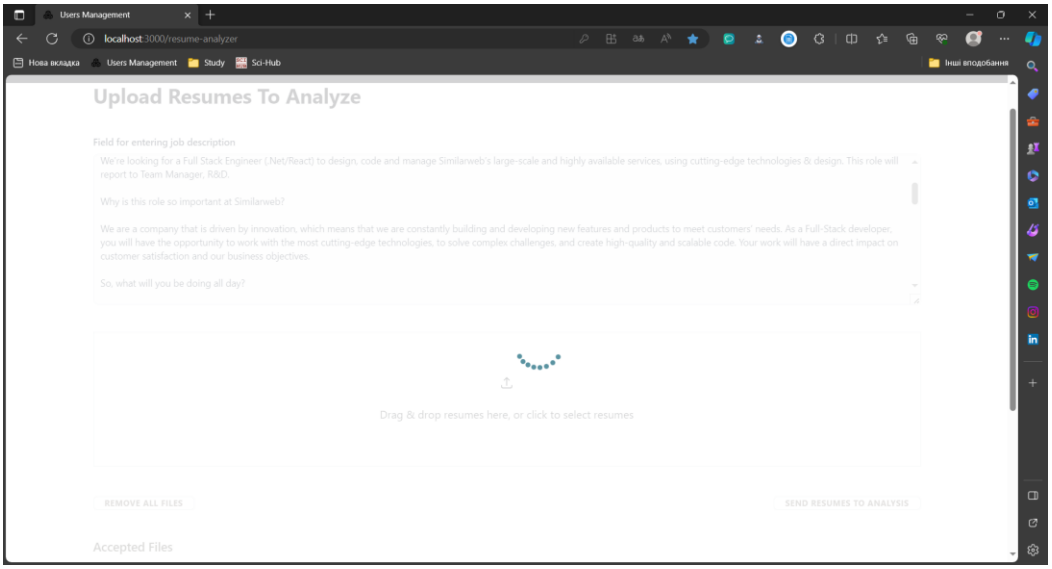


Рисунок 3.29 - Очікування результату

Нарешті аналіз закінчився і ми отримуємо таблицю з результатами (Рисунок 3.30). Для кожного рядочка ми маємо назву завантаженого резюме. Електронну пошту, яку ми дістаємо з резюме, а також оцінки на вибраними

категоріями та активні дії. Результати, які нам надала система відповідають очікуваним.

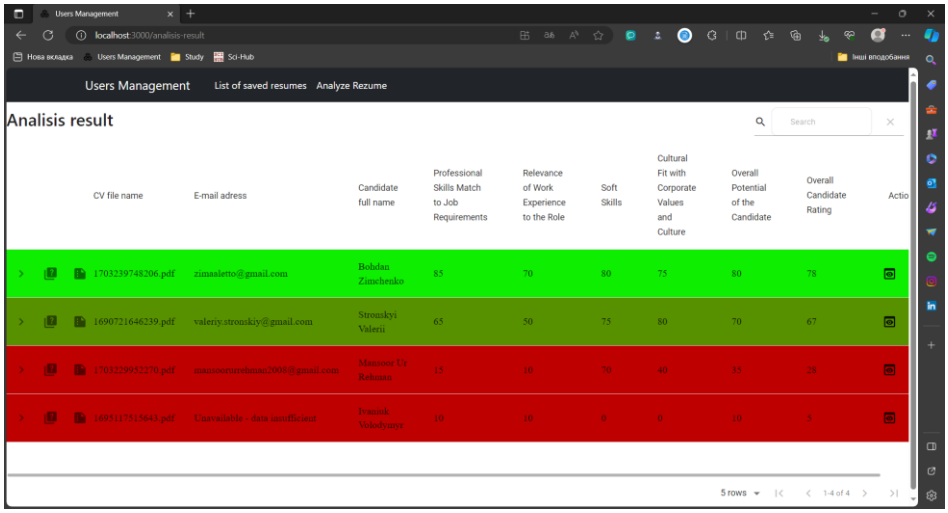


Рисунок 3.30 - Результат

Серед активних дій: розгортання пояснень до поставлених оцінок (Рисунок 3.31), перегляд списку питань (Рисунок 3.32), рекомендації (Рисунок 3.33) та завантаження резюме. Таблицю можна сортувати, фільтрувати тощо.

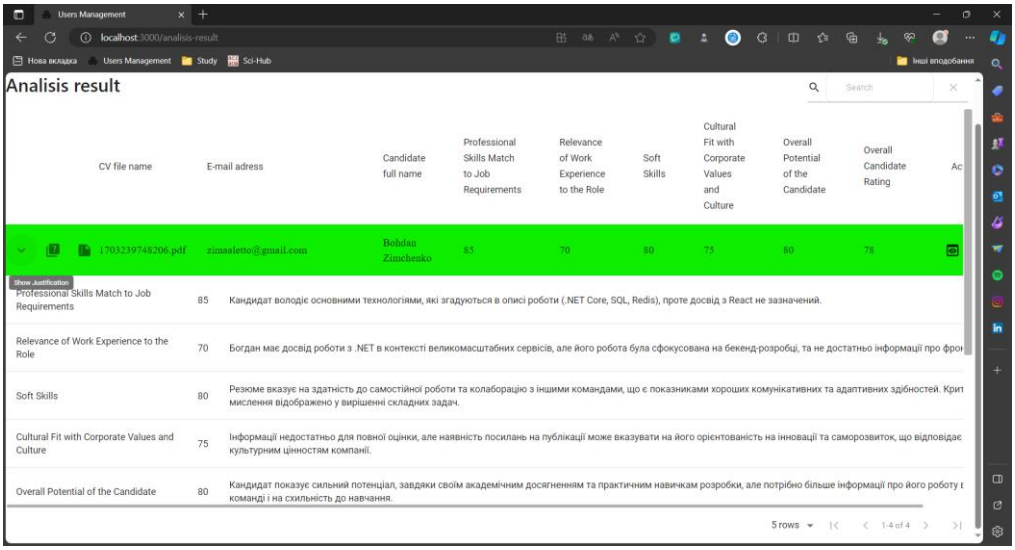


Рисунок 3.31 - Пояснення оцінок для випадково обраного кандидата

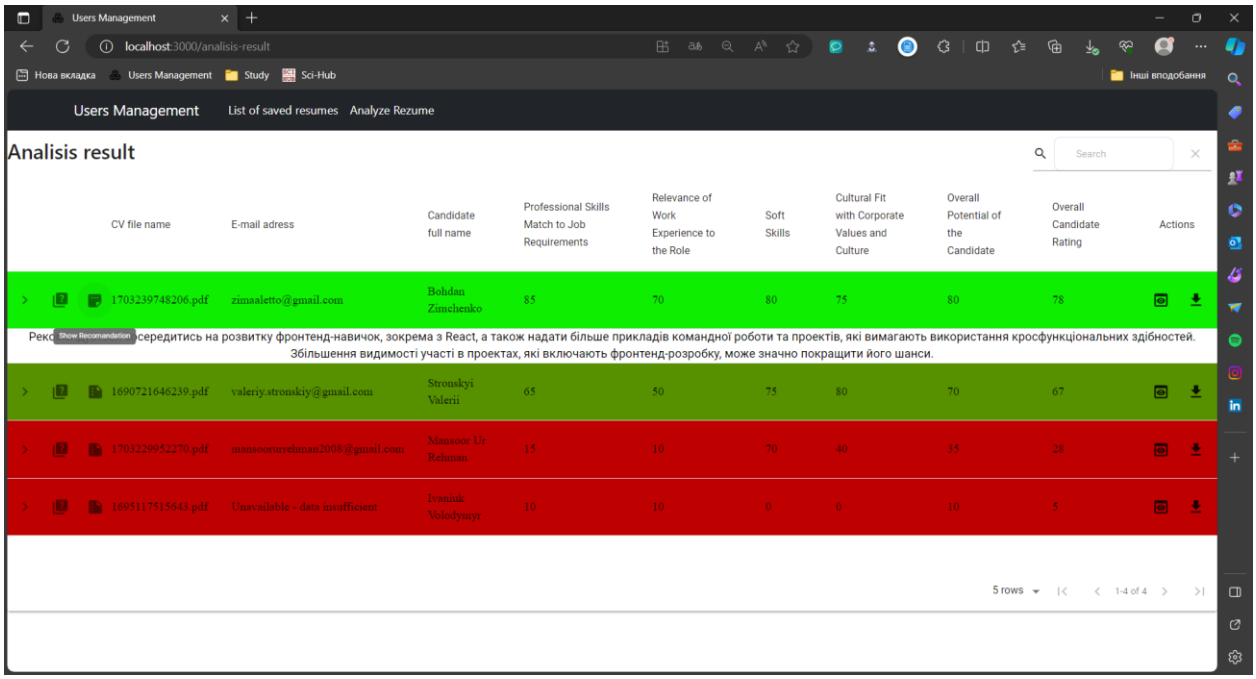


Рисунок 3.32 - Рекомендація за резюме кандидата

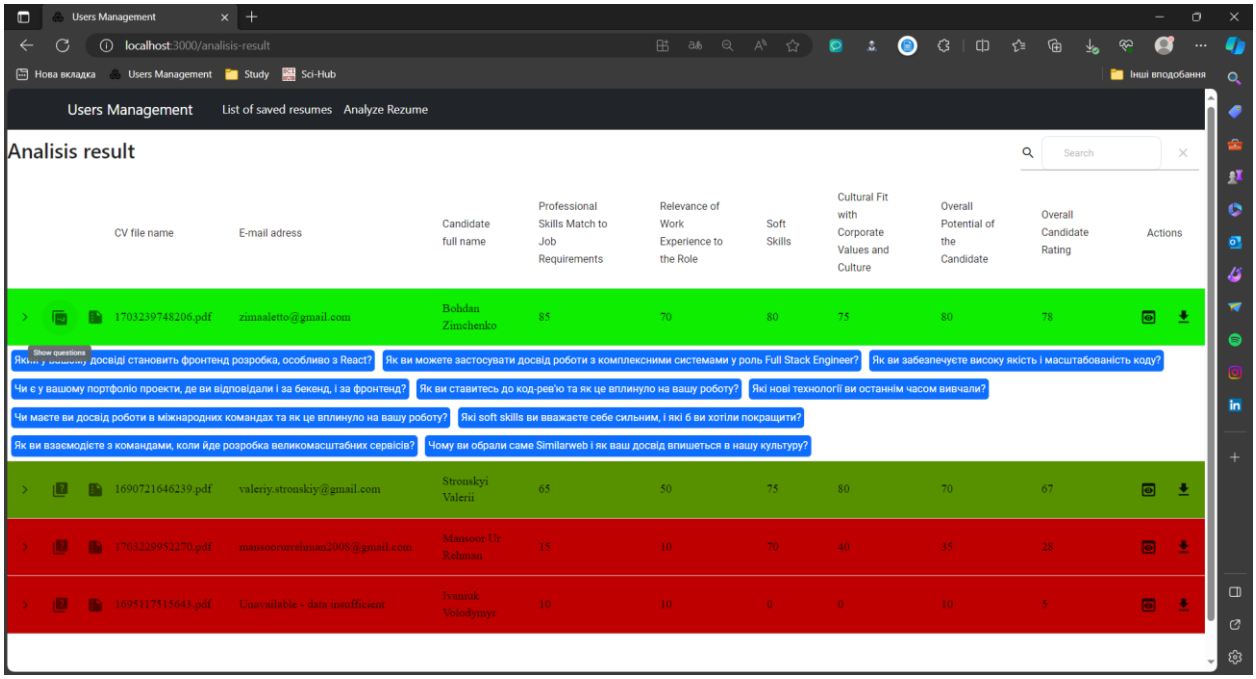


Рисунок 3.33 - Список питань кандидату

Бачимо, система працює штатно. Оцінки розраховуються вірно, пояснення до них чіткі і зрозумілі, рекомендації написані теж вірно. Питання до кандидатів змістовні. Для того, щоб ще доперевірити систему, на ту саму вакансію (ДОДАТОК Б) запропонуємо нові резюме (ДОДАТОК Г). Наведемо скріншот результату (Рисунок 3.34 - Результат аналізу 2).

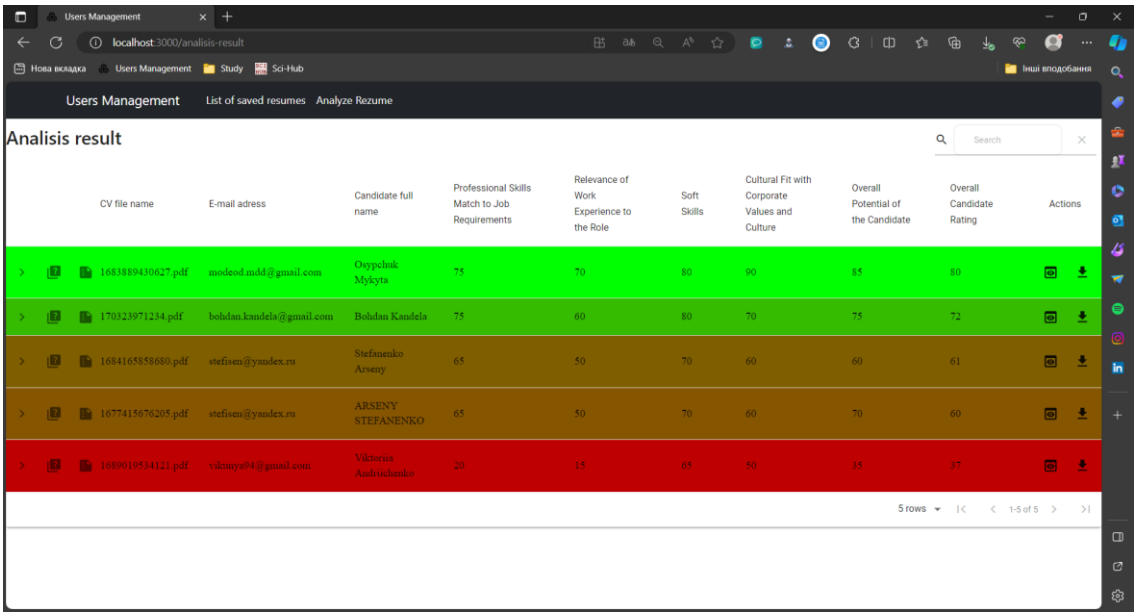


Рисунок 3.34 - Результат аналізу 2

Зробивши новий аналіз – знову бачимо що, система працює штатно (Рисунок 3.34). Оцінки розраховуються вірно (Рисунок 3.35), пояснення до них чіткі і зрозумілі (Рисунок 3.35), рекомендації написані теж вірно (Рисунок 3.36). Питання до кандидатів змістовні (Рисунок 3.37). Система виконує поставлені перед нею завдання.

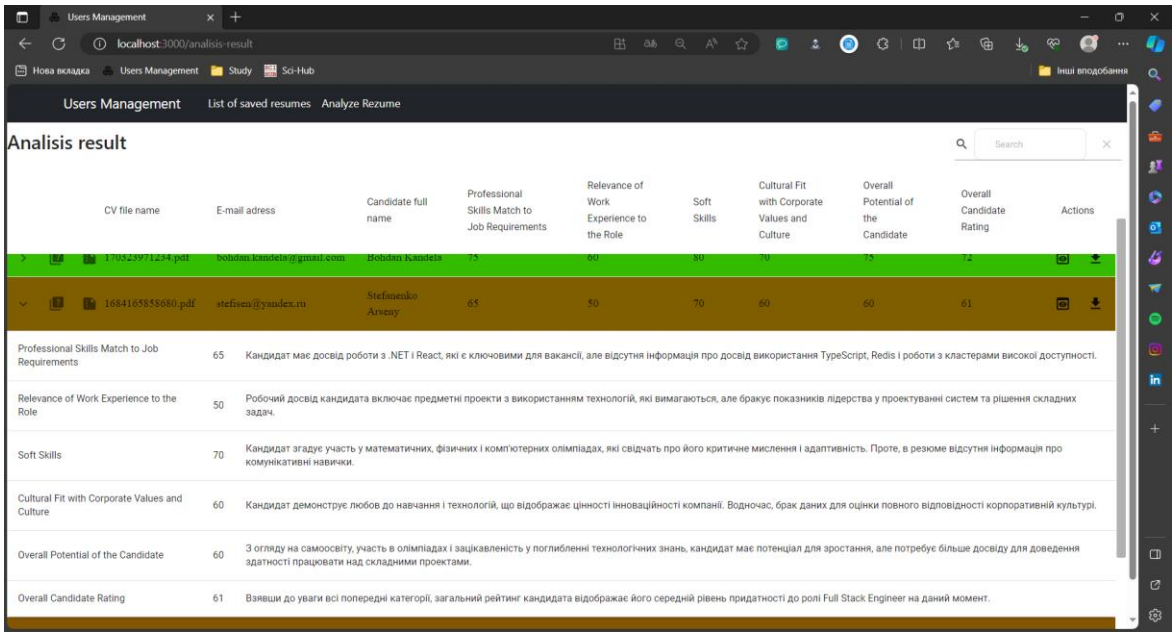


Рисунок 3.35 - Пояснення оцінок випадково обраного кандидата

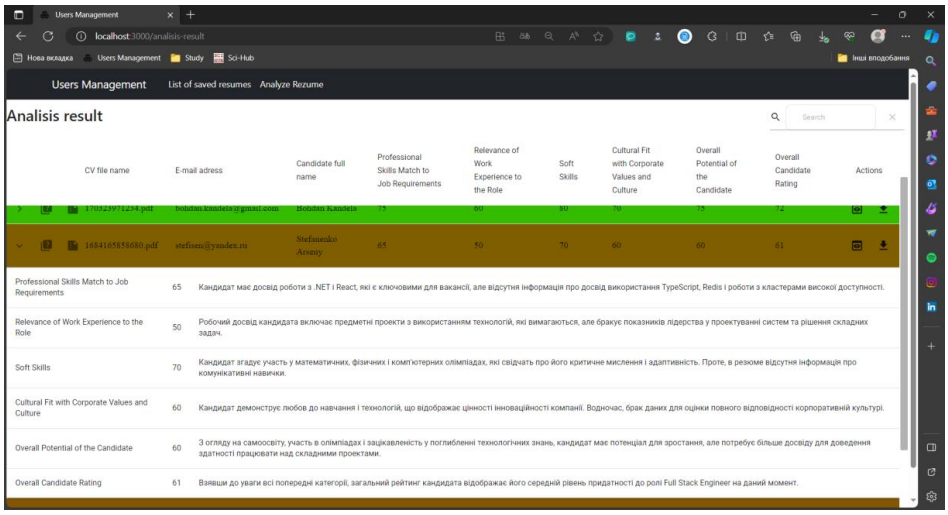


Рисунок 3.36 - Рекомендації за резюме випадково вибраного кандидата

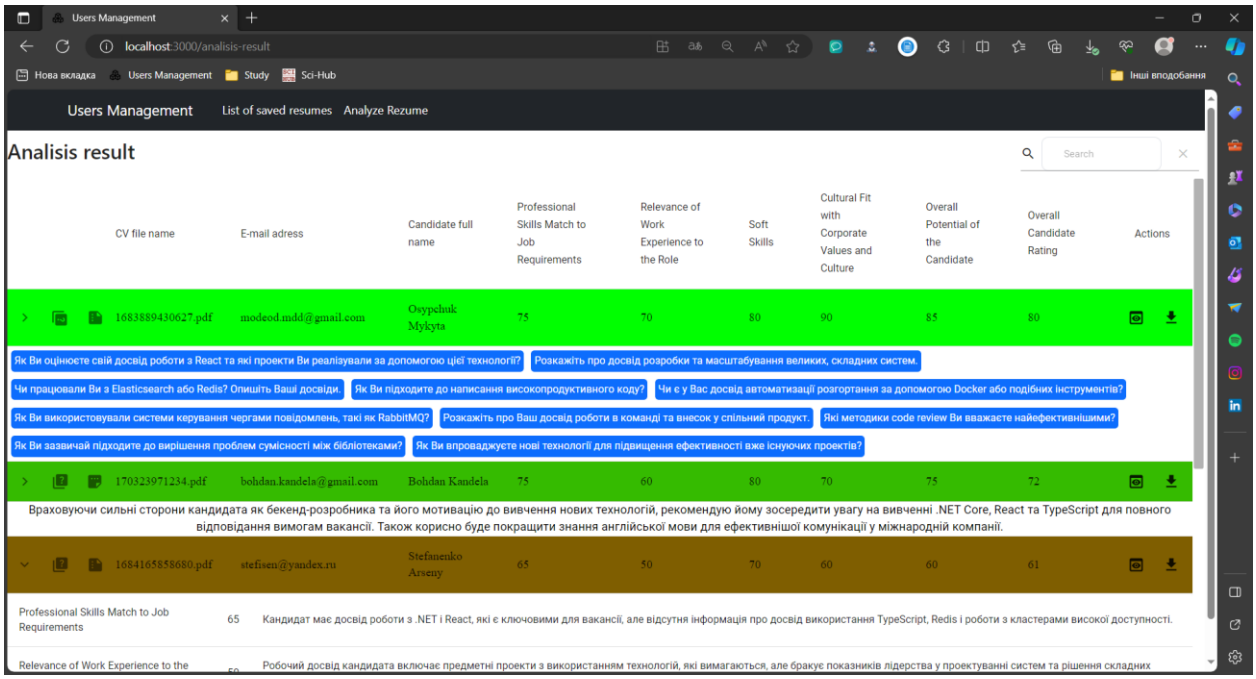


Рисунок 3.37 - Список питань до випадково обраного кандидата

ВИСНОВКИ

Отже, під час виконання кваліфікаційної магістерської роботи було. розглянуто поняття великих мовних моделей (LLM), сформовано загальні положення та принципи роботи даних технологій. Виконано ознайомлення з -платформами, де розглядаються такі моделі. Методами й принципами створення додатків з використанням моделей OpenAI (моделі серії GPT). За допомогою мов програмування js та c#, фреймворків Express, React, ASP Net Core вирішено поставлене завдання.

Реалізована інтелектуальна технологія автоматизації відбору персоналу. Вона задовольняє наступним вимогам:

1. Аналізує резюме кандидатів.

2. За результатами комплексного аналізу оцінює такі характеристики кандидата:

2.1. Відповідність професійних навичок вимогам вакансії.

2.2. Відповідність досвіду роботи до вимог вакансії.

2.3. М’які навички, включаючи комунікаційні навички, адаптивність та критичне мислення.

2.4. Відповідність корпоративним культурним цінностям і культурі компанії. За отриманим аналізом розглядати перспективність кандидата.

3. Робить рекомендацію щодо кожного кандидата.

4. Готує невеликий список питань, щоб провести пре-інтерв’ю для підтвердження навичок вказаних в резюме. На основі яких можна знову переоцінити рейтинг кандидата.

Для подальшого розширення проекту є декілька варіантів розвитку.

По-перше, можна реалізувати систему, яка сама буде проводити опитування кандидата і проводити інтерв’ю, за результатом якого рейтинг кандидата буде змінюватись. Для цього потрібно реалізувати систему з мовленням, а не текстом. Або ще краще з відео, де роль HR буде виконувати 3-Д модель.

По-друге, потрібно додати аналіз мотиваційних листів, які можуть бути прикріплені до резюме.

По-третє, потрібно реалізувати інтеграцію з відомими ресурсами для пошуку роботи – доу.юа, ворк.юа, лінкедін, джинні тощо.

По-четверте, потрібно створити сервіс, для генерації резюме, які краще вичитуватимуться системою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Erixon F., «Retrieved from The economic benefits of globalization for business and consumers», Січень 2018, [Електронний ресурс] − Режим доступу: https://cutt.ly/KwSF7fub.
2. Wall T.D. & Wood S.J., «The romance of human resource management and business performance, and the case for big science. Human Relations», 2005, с. 429-462.
3. Kok, J.D., & Uhlaner L.M., «Organization context and Human resource management in the Small Firm. Small Business Economics», 17(4), с.273-291, 2011.
4. D. O'Donovan, «HRM in the organization: An overview. Management Science. Management and industrial engineering», 2019, с. 75-110.
5. Galanaki, E., Lazazzara, A., & Parry, E., A cross-national analysis of e-HRM configurations: integrating the information technology and HRM perspectives, *Organizing for digital innovation* , 2019, 27, с. 261-276.
6. T. Bondarouk & C. Brewster, Conceptualizing the future of HRM and technology research. *The International Journal of Human Resource Management*, 2016, 27(21), с. 2652–2671.
7. Baxter, M. «Information-Age», Грудень 2018, [Електронний ресурс] − Режим доступу:https://www.information-age.com/business-analytics-intelligence-12194/
8. Upadhyay A.K., Khandelwal K. Applying artificial intelligence: implications for recruitment // Strategic HR Review, 2018, Vol. 17, No. 5, с. 255-258.
9. Tecuci G., Artificial Intelligence, Wires computational statistics, 2012, Vol. 4, No. 2, с. 168-180.
10. Stuart R., Norvig P., Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall Press Upper Saddle River, 2016.
11. Salin E.D., Winston P.H., Machine Learning and Artificial Intelligence, Analytical chemistry, 1992, Vol. 64, No. 1.

12. Nilsson N.J., Human-level artificial intelligence? be serious!, AI Magazine, 2005, Vol. 26, No. 4, c. 68-75.
13. McRobert C.J., Hill J.C., Smale T., Hay E.M., Van der Windt D.A., A multi-modal recruitment strategy using social media and internet-mediated methods to recruit a multidisciplinary, international sample of clinicians to an online research study, PLoS ONE, 2018, Vol. 13, No. 7.
14. Baron I.S., Mustafa, Agustina H., The challenges of recruitment and selection systems in Indonesia, Journal of management and marketing review, 2018, Vol. 3, No. 4, c. 185-192.
15. Chapman D.S., Webster J., The use of technologies in the recruiting, screening, and selection processes for job candidates, International journal of selection and assessment, 2003, Vol. 11, Nos. 2-3, c. 113–120.
16. Marler J.H., Fisher S.L., An evidence-based review of e-HRM and strategic human resource management, Human Resource Management Review, 2013, Vol. 23, No. 1, c. 18–36.
17. Stone D.L., Deadrick D.L., Lukaszewski K.M., Johnson R., The influence of technology on the future of human resource management, Human Resource Management Review, 2015, Vol. 25, c. 216–231.
18. Bau A., Belinkov Y., Sajjad H., Durrani N., Dalvi F., Glass J., Identifying and controlling important neurons in neural machine translation, arXiv preprint arXiv:1811.01157, 2018.
19. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample G., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2307.06018, 2023.
20. Wei X., Wei H., Lin H., Li T., Zhang P., Ren X., Li M., Wan Y., Cao Z., Xie B., et al., Polylm: An open source polyglot large language model, arXiv preprint arXiv:2307.06018, 2023.

21. Zhao J., Yu B., Yu H., Li B., Wang C., Huang F., Li Y., Zhang N.L., Causal document-grounded dialogue pre-training, arXiv preprint arXiv:2305.10927, 2023.
22. Bhaskar A., Fabbri A., Durrett G., Prompted opinion summarization with gpt-3.5, In Findings of the Association for Computational Linguistics: ACL 2023, 2023, с. 9282-9300.
23. Chen M., Tworek J., Jun H., Yuan Q., Pinto H.P.O., Kaplan J., Edwards H., Burda Y., Joseph N., Brockman G., et al., Evaluating large language models trained on code, arXiv preprint arXiv:2107.03374, 2021.
24. Saunders W., Yeh C., Wu J., Bills S., Ouyang L., Ward J., Leike J., Self-critiquing models for assisting human evaluators, arXiv preprint arXiv:2206.05802, 2022.
25. Papineni K., Roukos S., Ward T., Zhu W.-J., Bleu: a method for automatic evaluation of machine translation, In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, Philadelphia, Pennsylvania, USA, с. 311-318.
26. Lin C.-Y., ROUGE: A package for automatic evaluation of summaries, In Text Summarization Branches Out, 2004, Barcelona, Spain, с. 74-81.
27. Liu Y., Iter D., Xu Y., Wang S., Xu R., Zhu C., Gpteval: NLG evaluation using gpt-4 with better human alignment, arXiv preprint arXiv:2303.16634, 2023.
28. Jain S., Ma X., Deoras A., Xiang B., Self-consistency for open-ended generations, arXiv preprint arXiv:2307.06857, 2023.
29. Li M., Song F., Yu B., Yu H., Li Z., Huang F., Li Y., Api-bank: A benchmark for tool-augmented llms, arXiv preprint arXiv:2304.08244, 2023.
30. Helder, S. «LLM Models Comparison: GPT-4, Bard, LLaMA, Flan-UL2, BLOOM», Грудень 2022, [Електронний ресурс] − Режим доступу: https://medium.com/@aiwizard/llm-models-comparison-gpt-4-bard-llama-flan-ul2-bloom-9ad7c0c56ba5
31. OpenAI API Docs // Офіційна документація. [Електронний ресурс] - Режим доступу: Overview - OpenAI API

32. Harsha N., Nicholas K., Scott M. McKinney, Dean C., Eric H., «Capabilities of GPT-4 on Medical Challenge Problems», Березень 2023, [Електронний ресурс] − Режим доступу: https://doi.org/10.48550/arXiv.2303.13375
33. Bard // Документація, [Електронний ресурс] - Режим доступу: google-about-bard.pdf (ai.google)
34. LLaMA // Документація, [Електронний ресурс] - Режим доступу: Llama 2 - Resource Overview - Meta AI
35. Flan-UL2 // Документація, [Електронний ресурс] - Режим доступу: FLAN-UL2 (huggingface.co)
36. BLOOM // Документація, [Електронний ресурс] - Режим доступу: BLOOM (huggingface.co)
37. React // Документація, [Електронний ресурс] - Режим доступу: https://legacy.reactjs.org/docs
38. Express // Документація, [Електронний ресурс] - Режим доступу: Express documentation — DevDocs
39. Rajeev, S. «Why Express.js Is Used For Enterprise App Development: Top 6 Reasons», Червень 2022, [Електронний ресурс] − Режим доступу: Top 6 Reasons To Use Express.js For Enterprise App Development (markovate.com)
40. ASP.NET Core // Документація, [Електронний ресурс] - Режим доступу: ASP.NET Core | Open-source web framework for .NET (microsoft.com)
41. VS Code // IDE, Документація, [Електронний ресурс] - Режим доступу: Documentation for Visual Studio Code

ДОДАТОК А

Server.js

```
const express = require('express');
const helmet = require('helmet');

const corsMiddleware = require('./middleware/cors');
const errorHandlerMiddleware = require('./middleware/customErrorHandler');
const cvUpload = require('express-fileupload');
const db = require('./utils/DB');
const fs = require('fs-extra');
const path = require('path');
const https = require('https');

var config = require('./config.js');
const cvRouter = require('./routes/cvRoutes');
import { opts } from './middleware/clientSertificate';

const port = config.PORT;
const addr = config.ADDRESS;

const app = express();
app.use(helmet.hidePoweredBy());
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cvUpload());

if (!fs.existsSync(path.join(config.STORAGE_NAME))) {
    fs.mkdirSync(path.join( config.STORAGE_NAME));
}

let server = null;

function startPortListen() {
    server = https.createServer(opts, app).listen(port, () => {
        console.log(`Server running in https://${addr}:${port}`);
    }).on('error', (err) => {
        if (err.code === 'EADDRINUSE') {
            console.log(`Port ${port} is busy.`);
        } else {
            console.log(err);
        }
    });
}
function runApp() {
    db.connect();
    startPortListen();
}

runApp();

app.use(corsMiddleware);
app.use('/', cvRouter);

app.use(errorHandlerMiddleware);
```

```
module.exports = server;
```

CVContoller.js

```
import template from '../../resources/strTemplate';
import resonseTemplate from '../utils/Response';
import path from 'path';
import config from '../config';

import DBService from '../services/DBService';
import FSService from '../services/FSService';
import signature from '../middleware/signature';

import {
    checkIdOnValid,
    responseCodes,
    generateCVId
} from '../utils/Helpers';

export default class CVController {
    constructor() {
        this.getAllCVs = this.getAllCVs.bind(this);
        this.getCV = this.getCV.bind(this);
        this.createCV = this.createCV.bind(this);
        this.replaceCV = this.replaceCV.bind(this);
        this.deleteCV = this.deleteCV.bind(this);
        this.getSignedUrl = this.getSignedUrl.bind(this);
        this._dbService = new DBService();
        this._fsService = new FSService();
    }

    getSignedUrl (req, res) {
        if (req.body.url || req.body.Url) {
            const stringToSign = req.body.url || req.body.Url;
            const signedUrl = signature.sign(stringToSign);

            return
res.status(200).json(resonseTemplate.getResponseToSignedUrl(signedUrl));
        }

        return res.status(400).json(template.singUrlError);
    }

    async getAllCVs(req, res) {
        const senObj = await this._dbService.getAllCVFromDBAsync();

        if (senObj) {
            return res.status(responseCodes.ok).json(senObj);
        }

        return res.status(responseCodes.notFound).json(
            resonseTemplate.getResponseWithoutData(
                template.cvsNotFound,
            )
        );
    }

    async getCV(req, res) {
```

```
            const receivedCVId = req.params.id;

            if (checkIdOnValid(res, receivedCVId)) {
                return undefined;
            }

            const findedCV = await
this._dbService.getCVFromDBByIdAsync(receivedCVId);

            try {
                if (findedCV) {
                    const respObj =
this._fsService.getPathAndNameCV(findedCV);
                    return
res.status(responseCodes.ok).download(respObj.savePath, respObj.cvName);
                }

                return res.status(responseCodes.notFound).json(
                    resonseTemplate.getResponseUsingId(
                        template.cvNotFoundError,
                        receivedCVId
                    )
                );
            } catch (error) {
                console.log('CV was not sent');
                console.log(error);

                return res.status(responseCodes.serverError).json(
                    resonseTemplate.getResponseUsingCVName(
                        template.cvWasNotSent
                    )
                );
            }
        }

        async createCV(req, res) {
            const bdCVId = generateCVId();
            const uploadedCV = req.cvs.cv;

            const cvInStorageCreatedResult = await
this._fsService.createCVInStorageAsync(uploadedCV, bdCVId);
            if (cvInStorageCreatedResult === undefined){

                return
res.status(responseCodes.serverError).json(resonseTemplate.getResponseWithoutData
(
                    template.storageCVSavingError
                ));
            }

            const cvInDbCreatedResult = await
this._dbService.createCVInDBAsync(cvInStorageCreatedResult, bdCVId);
            if (cvInDbCreatedResult === undefined){
                this._fsService.deleteCVFolder(bdCVId);

                return
res.status(responseCodes.serverError).json(resonseTemplate.getResponseWithoutData
(
                    template.dbCVSavingError
```

```
                ));
            }

            return
res.status(responseCodes.cvCreated).json(resonseTemplate.getResponseUseUploadcvAn
dID(
                template.cvWasUploaded,
                uploadedCV,
                bdCVId
            ));
        }

        async replaceCV(req, res) {
            const receivedCVId = req.params.id;

            if (checkIdOnValid(res, receivedCVId)) {
                return undefined;
            }

            const uploadedCV = req.cvs.cv;
            const cvPath = path.join(config.STORAGE_NAME,
receivedCVId.toString(), '/', uploadedCV.name);

            try {
                const replaceCVInDbResult = await
this._dbService.replaceCVInDBAsync(receivedCVId, cvPath);

                if (replaceCVInDbResult !== undefined)
                {
                    const replaceResult = await
this._fsService.replaceCVInStorageAsync(uploadedCV, replaceCVInDbResult);
                    return
res.status(replaceResult.code).json(replaceResult.message);
                }

                return
res.status(responseCodes.notFound).json(resonseTemplate.getResponseUsingId(
                    template.cvUpdatingError +
                    template.cvNotFoundError,
                    receivedCVId
                ));

            } catch (error) {

                return
res.status(responseCodes.serverError).json(resonseTemplate.getResponseUsingId(
                    template.cvUpdatingError,
                    receivedCVId
                ));
            }
        }

        async deleteCV(req, res) {
            const receivedCVId = req.params.id;

            if (checkIdOnValid(res, receivedCVId)) {
                return undefined;
            }
```

```
            try {
                const deletedCV =
this._fsService.deleteCVFolder(receivedCVId);

                if (deletedCV) {
                    const deleteFromDB = await
this._dbService.deleteCVInDBAsync(receivedCVId);
                    return
res.status(deleteFromDB.code).json(deleteFromDB.message);
                }

                return
res.status(responseCodes.notFound).json(resonseTemplate.getResponseUsingId(
                    template.cvNotFoundError,
                    receivedCVId
                ));
            } catch (error) {
                return
res.status(responseCodes.serverError).json(error.message);
            }
        }
    }
```

cvRoute.js

```
    const express = require('express');
    const clientCertificateAuth = require('client-certificate-auth');

    import CVController from '../controllers/CVController';
    import signature from '../middleware/signature';
    import { checkAuth } from '../middleware/clientSertificate';
    import limitSupportedFormat from '../middleware/limitSupportedFormat';

    var cvController = new CVController();

    var router = express.Router();

    const cvsRoute = '/api/cvs';
    const cvRoute = cvsRoute + '/:id';
    const getSignedUrlRoute = cvsRoute + '/get-signed-url';

    router.get(cvsRoute, cvController.getAllCVs,);
    router.get(cvRoute,  cvController.getCV);
    router.post(getSignedUrlRoute, clientCertificateAuth(checkAuth),
cvController.getSignedUrl);
    router.post(cvsRoute, signature.verifier(), limitSupportedFormat,
cvController.createCV);
    router.put(cvRoute, signature.verifier(), limitSupportedFormat,
cvController.replaceCV);
    router.delete(cvRoute, signature.verifier(),  cvController.deleteCV);

    module.exports = router;
```

Resume.cs

```
    using System;
    using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;

namespace DBStorage.Resumes
{
    public class Resume
    {
        public Guid Id { get; set; }

        public string Name { get; set; }
        public string Description { get; set; }
        public string FileKey { get; set; }
        public string PreviewBlobKey { get; set; }
        public DateTime CreatedAt { get; set; }
        public Guid CreatedBy { get; set; }
        public DateTime UpdatedAt { get; set; }
        public Guid UpdatedBy { get; set; }

        public virtual ICollection<KeyWord> KeyWords { get; set; }
        public virtual ICollection<ResumeHistory> ResumeHistories { get;
set; }
    }
}
```

ResumrHistory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace DBStorage.Resumes
{
    public class Resume
    {
        public Guid Id { get; set; }

        public string Name { get; set; }
        public string Description { get; set; }
        public string FileKey { get; set; }
        public string PreviewBlobKey { get; set; }
        public DateTime CreatedAt { get; set; }
        public Guid CreatedBy { get; set; }
        public DateTime UpdatedAt { get; set; }
        public Guid UpdatedBy { get; set; }

        public virtual ICollection<KeyWord> KeyWords { get; set; }
        public virtual ICollection<ResumeHistory> ResumeHistories { get;
set; }
    }
}
```

ResumeRepository.cs

```
using DBStorage.Interfaces;
using DBStorage.Resumes;
using Microsoft.EntityFrameworkCore;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DBStorage.Repositories
{
    public class ResumeRepository : IResumeRepository
    {
        private readonly ResumesContext _db;

        public ResumeRepository(ResumesContext context)
        {
            _db = context;
        }

        public async Task<Resume> CreateResumeAsync(Resume resume)
        {
            try
            {
                _db.Resumes.Add(resume);
                await _db.SaveChangesAsync();

                Resume createdResume = await _db.Resumes
                    .Include(resume => resume.KeyWords)
                    .ThenInclude(keyWord => keyWord.Resumes)
                    .FirstOrDefaultAsync(u => u.Id.Equals(resume.Id));

                ResumeHistory resumeHistory = new ResumeHistory
                {
                    Id = Guid.NewGuid(),
                    ResumeId = createdResume.Id,
                    FileKey = createdResume.FileKey,
                    CreatedAt = createdResume.CreatedAt,
                    CreatedBy = createdResume.CreatedBy
                };

                _db.ResumeHistories.Add(resumeHistory);
                await _db.SaveChangesAsync();

                return createdResume;
            }
            catch (DbUpdateException ex)
            {
                Console.WriteLine(ex);
                throw new DbUpdateException(ex.Message);
            }
        }

        public async Task<bool> DeleteResumeAsync(Resume resume)
        {
            try
            {
                Resume resumeToDelete = _db.Resumes.Find(resume.Id);

                if (resumeToDelete != null)
                {
                    _db.Resumes.Remove(resumeToDelete);
                    await _db.SaveChangesAsync();
```

```
                }
                else
                {
                    throw new Exception("Resume was not founded");
                }

            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }
            return true;
        }

        public async Task<List<Resume>> GetAllResumesAsync()
        {
            IQueryable<Resume> Resumes = _db.Resumes
                .Include(resume => resume.KeyWords)
                .ThenInclude(keyWord => keyWord.Resumes)
                .OrderByDescending(resume => resume.UpdatedAt)
                .AsQueryable();

            List<Resume> result = await Resumes.ToListAsync();

            return result;
        }

        public async Task<Resume> GetResumeByIdAsync(Guid id)
        {
            Resume resume = await _db.Resumes
                .Include(resume => resume.KeyWords)
                .ThenInclude(keyWord => keyWord.Resumes)
                .FirstOrDefaultAsync(u => u.Id.Equals(id));

            return resume;
        }

        public async Task<Resume> GetResumeByNameAsync(String name)
        {
            Resume resume = await _db.Resumes
                .Include(resume => resume.KeyWords)
                .ThenInclude(keyWord => keyWord.Resumes)
                .FirstOrDefaultAsync(m => m.Name.Equals(name));

            return resume;
        }

        public async Task<Resume> UpdateResumeAsync(Resume resumeToUpdate,
Resume resumeUpdated)
        {
            string oldFileKey = resumeToUpdate.FileKey;

            _db.Entry(resumeToUpdate).State = EntityState.Modified;
            resumeToUpdate.KeyWords.Clear();
            resumeToUpdate.Name = resumeUpdated.Name;
            resumeToUpdate.Description = resumeUpdated.Description;
            resumeToUpdate.FileKey = resumeUpdated.FileKey;
            resumeToUpdate.PreviewBlobKey = resumeUpdated.PreviewBlobKey;
            resumeToUpdate.UpdatedBy = resumeUpdated.UpdatedBy;
```

```
                    resumeToUpdate.KeyWords = resumeUpdated.KeyWords;

                    try
                    {
                        await _db.SaveChangesAsync();

                        Resume updatedResume = await _db.Resumes
                            .Include(resume => resume.KeyWords)
                            .ThenInclude(keyWord => keyWord.Resumes)
                            .FirstOrDefaultAsync(u =>
u.Id.Equals(resumeToUpdate.Id));

                        string newFileKey = updatedResume.FileKey;

                        if (oldFileKey != newFileKey)
                        {
                            ResumeHistory resumeHistory = new ResumeHistory
                            {
                                Id = Guid.NewGuid(),
                                ResumeId = updatedResume.Id,
                                FileKey = updatedResume.FileKey,
                                CreatedAt = updatedResume.UpdatedAt,
                                CreatedBy = updatedResume.UpdatedBy
                            };

                            _db.ResumeHistories.Add(resumeHistory);
                            await _db.SaveChangesAsync();
                        }

                        return updatedResume;
                    }
                    catch (DbUpdateConcurrencyException ex)
                    {
                        Console.WriteLine(ex);
                        throw new DbUpdateConcurrencyException(ex.Message);
                    }
                    catch (DbUpdateException ex)
                    {
                        Console.WriteLine(ex);
                        throw new DbUpdateException(ex.Message);
                    }
                }
            }
        }
```

ResumeController.cs

```
        using System;
        using System.Collections.Generic;
        using System.IO;
        using System.Linq;
        using System.Net.Http;
        using System.Net.Http.Headers;
        using System.Threading.Tasks;
        using DBStorage.Resumes;
        using DBStorage.Resumes.ResumeDTO;
        using DBStorage.Repositories;
        using Microsoft.AspNetCore.Authorization;
```

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using UserManagement.Utils.CustomExceptions;
using UserManagement.Utils.Services;

namespace UserManagement.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class ResumesController : ControllerBase
    {
        private readonly ResumeRepository resumes;
        private readonly UserRepository users;
        private readonly KeyWordRepository keyWords;
        private readonly ResumeHistoryRepository resumeHistories;

        public ResumesController(ResumesContext resumeContext,
UsersContext userContext, AppSettingsService appService)
        {
            resumes = new ResumeRepository(resumeContext);
            users = new UserRepository(userContext);
            keyWords = new KeyWordRepository(resumeContext);
            resumeHistories = new ResumeHistoryRepository(resumeContext);
        }

        [AllowAnonymous]
        [HttpGet("{id}")]
        public async Task<ActionResult<ResumeDTO>> GetResumeAsync(Guid id)
        {
            if (id == Guid.Empty)
            {
                return BadRequest();
            }

            Resume resume = await resumes.GetResumeByIdAsync(id);

            if (resume == null)
            {
                return NotFound();
            }

            return ResumeToDTOWithUserEmailInsteadOfId(resume);
        }

        [AllowAnonymous]
        [HttpGet]
        public async Task<ActionResult<IEnumerable<ResumeDTO>>>
GetResumesAsync()
        {
            List<Resume> receivedResumes = await
resumes.GetAllResumesAsync();
            List<ResumeDTO> resumeDTOs = receivedResumes.Select(resume =>
ResumeToDTOWithUserEmailInsteadOfId(resume)).ToList();
            return resumeDTOs;

        }
```

```
            [AllowAnonymous]
            [HttpGet("{id}/history")]
            public async Task<ActionResult<List<ResumeHistoryDTO>>>
GetResumeHistoriesAsync(Guid id)
            {
                List<ResumeHistory> receivedResumeHistories = await
resumeHistories.GetResumeHistoryByResumeIdAsync(id);

                List<ResumeHistoryDTO> resumeHistoryDTOs =
receivedResumeHistories.Select(resumeHistory =>
ResumeHistoryToDTOWithUserEmailInsteadOfId(resumeHistory)).ToList();

                return resumeHistoryDTOs;
            }

            [HttpPut("{id}")]
            public async Task<IActionResult> PutResumeAsync(Guid id, ResumeDTO
resumeDTO)
            {
                if (resumeDTO == null || id == Guid.Empty)
                {
                    return BadRequest();
                }

                Resume findResume = await resumes.GetResumeByIdAsync(id);

                if (findResume == null)
                {
                    return NotFound();
                }

                Resume generatedResume = await
GenerateResumeToUpdateAsync(resumeDTO, id);

                try
                {
                    findResume = await resumes.UpdateResumeAsync(findResume,
generatedResume);
                    return
Ok(ResumeToDTOWithUserEmailInsteadOfId(findResume));
                }
                catch (DbUpdateConcurrencyException ex)
                {
                    return Problem(ex.Message);
                }
            }

            [HttpPost]
            public async Task<ActionResult<Resume>> PostResumeAsync(ResumeDTO
resume)
            {
                try
                {
                    Resume generatedResume = await
GenerateResumeAsync(resume);
                    generatedResume = await
resumes.CreateResumeAsync(generatedResume);
                    return
Ok(ResumeToDTOWithUserEmailInsteadOfId(generatedResume));
```

```
                }
                catch (DbUpdateException ex)
                {
                    return new BadRequestObjectResult(ex.Message);
                }
                catch (KeyWordNotFoundException ex)
                {
                    return new NotFoundObjectResult(ex.Message);
                }
            }

            [HttpDelete("{id}")]
            public async Task<IActionResult> DeleteResumeAsync(Guid id)
            {
                Resume resume = await resumes.GetResumeByIdAsync(id);
                if (resume == null)
                {
                    return NotFound();
                }
                try
                {
                    if (await resumes.DeleteResumeAsync(resume))
                    {
                        return Ok();
                    }

                    return StatusCode(500, new { message =
Properties.Resources.ErrDeleteResume });
                }
                catch (Exception ex)
                {
                    return Problem(ex.Message);
                }
            }

            private async Task<List<KeyWord>>
ComposeReceivedAndDBKeyWords(List<string> receivedKeyWordsId)
            {
                List<KeyWord> composedKeyWords = new List<KeyWord>();

                foreach (var keyWordId in receivedKeyWordsId)
                {
                    KeyWord findedKeyWord = await
keyWords.GetKeyWordByIdAsync(new Guid(keyWordId));

                    if (findedKeyWord == null)
                    {
                        throw new
KeyWordNotFoundException(string.Format(Properties.Resources.ObjWithIdNotExist,
"KeyWord", keyWordId));
                    }
                    composedKeyWords.Add(findedKeyWord);
                }
                return composedKeyWords;
            }

            private async Task<Resume> GenerateResumeAsync(ResumeDTO resume)
            {
```

```
                List<KeyWord> composedKeyWords = await
ComposeReceivedAndDBKeyWords(resume.KeyWords);

                return new Resume
                {
                    Id = Guid.NewGuid(),
                    Name = resume.Name,
                    Description = resume.Description,
                    FileKey = resume.FileKey,
                    PreviewBlobKey = resume.PreviewBlobKey,
                    CreatedBy = resume.UserId,
                    UpdatedBy = resume.UserId,
                    KeyWords = composedKeyWords
                };
            }

            private async Task<Resume> GenerateResumeToUpdateAsync(ResumeDTO
resume, Guid id)
            {
                List<KeyWord> composedKeyWords = await
ComposeReceivedAndDBKeyWords(resume.KeyWords);

                return new Resume
                {
                    Id = id,
                    Name = resume.Name,
                    Description = resume.Description,
                    FileKey = resume.FileKey,
                    PreviewBlobKey = resume.PreviewBlobKey,
                    UpdatedBy = resume.UserId,
                    KeyWords = composedKeyWords
                };
            }

            private ResumeHistoryDTO
ResumeHistoryToDTOWithUserEmailInsteadOfId(ResumeHistory history)
            {
                ResumeHistoryDTO replacedHistory =
ResumeHistoryDTO.ResumeHistoryToDTO(history);
                replacedHistory.CreatedBy =
GetUserEmailOrSetUserDeleted(history.CreatedBy).Result;

                return replacedHistory;
            }

            private ResumeDTO ResumeToDTOWithUserEmailInsteadOfId(Resume
resume)
            {
                ResumeDTO repalcedResume = ResumeDTO.ResumeToDTO(resume);
                repalcedResume.UpdatedBy =
GetUserEmailOrSetUserDeleted(resume.UpdatedBy).Result;

                return repalcedResume;
            }

            private async Task<String> GetUserEmailOrSetUserDeleted(Guid id)
            {
                User user = await users.GetUserByIdAsync(id);
                if (user is not null)
```

```
            {
                return user.Email;
            }
            return Properties.Resources.DeletedUser;
        }
    }
}
```

Analyser

Server.js

```
const express = require('express');
const helmet = require('helmet');

const corsMiddleware = require('./middleware/cors');
const limitSupportedFormat = require('./middleware/limitSupportedFormat');

const fs = require('fs-extra');
const path = require('path');

const cvUpload = require('express-fileupload');

var config = require('./config.js');
const analyzerRouter = require('./routes/AnalyzerRoutes');

const port = config.PORT;
const addr = config.ADDRESS;

const app = express();
app.use(helmet.hidePoweredBy());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cvUpload());

let server = null;

function startPortListen() {
    server = app.listen(port, () => {
        console.log(`Server running in http://${addr}:${port}`);
    }).on('error', (err) => {
        if (err.code === 'EADDRINUSE') {
            console.log(`Port ${port} is busy.`);
        } else {
            console.log(err);
        }
    });
}
function runApp() {
    startPortListen();
}

runApp();

app.use(corsMiddleware);
app.use(limitSupportedFormat);
app.use('/', analyzerRouter);
```

```
module.exports = server;
```

AnalyzerController.js

```
import OpenAI from 'openai';
import fs from 'fs';
import path from 'path';

var config = require('../config.js');
const pdf = require('pdf-parse');

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});
const prettifyText = (rawText) => {
  // Example of a more efficient text cleanup
  return rawText
    .trim()
    .replace(/[\r\n]{2,}/g, '\n') // Replace multiple line breaks with a
single one
    .replace(/\s{2,}/g, ' '); // Replace multiple spaces with a single
space
}

export default class AnalyzerController {
  constructor() {
    this.createQuestions = this.createQuestions.bind(this);
    this.analyzeResumes = this.analyzeResumes.bind(this);
    this.getStatus = this.getStatus.bind(this);
  }

  async getStatus(req, res) {
    return res.sendStatus(200);
  }
  async analyzeResumes(req, res) {
    try {
      const vacancyText = req.body?.vacancyDescription ?? 'no
information';
      const responseLang = req.body?.responseLang ?? 'ukrainian';

      console.log(req.files.cv)

      const json_object_patterrn = {
        fullName: "Candidate full name from cv",
        email: "email from cv",
        scores: [
          {
            category: "Professional Skills Match to Job Requirements",
            score: "score after analysis",
            justification: ""
          },
          {
            category: "Relevance of Work Experience to the Role",
            score: "score after analysis",
            justification: ""
          },
          {
```

```
                  category: "Soft Skills",
                  score: "score after analysis",
                  justification: ""
                },
                {
                  category: "Cultural Fit with Corporate Values and Culture",
                  score: "score after analysis",
                  justification: ""
                },
                {
                  category: "Overall Candidate Rating",
                  score: "score after analysis",
                  justification: ""
                },
                {
                  category: "Overall Potential of the Candidate",
                  score: "score after analysis",
                  justification: ""
                }
              ],
              recommendations: "",
              questions: []
            };

            const analysesPromises = req.files.cv.map(file =>
              pdf(file.data).then(async data => {
                const resumeText = prettifyText(data.text);
                const response = await openai.chat.completions.create({
                  model: "gpt-4-1106-preview",
                  messages: [
                    {
                      role: "system",
                      content: "As an HR expert, provide a comprehensive
evaluation of the candidate's resume. Assess the candidate in specified
categories, rating them from 1 to 100. If data is insufficient for a category,
note it and count as 0 in the overall rating. Present the analysis in JSON format
for JavaScript processing.  Use the specified language for all text."
                    },
                    {
                      role: "user",
                      content: `Evaluate the resume considering this job vacancy
[${vacancyText}]. Categories are:
                                1. Match of Professional Skills to Job
Requirements.
                                2. Relevance of Work Experience to the Role.
                                3. Soft Skills: Communication, Adaptability,
Critical Thinking.
                                4. Cultural Fit with Corporate Values and
Culture.
                                5. Overall Potential of the Candidate.
                                6. Overall Candidate Rating (weighted most
heavily, based on all other categories).
                                7. Recommendations (at least a few sentences)
                                8.After the evaluation, generate a list of
questions for the candidate to help them improve their skills and better align
with the job's demands. Format these questions in a clear and concise manner. (at
least 10 questions)
```

```
                                Follow this JSON format:
[${JSON.stringify(json_object_patternn)}]. Json property name MUST BE as in
pattern, but responses should be in [${responseLang}].`
                    },
                    {
                      role: "user",
                      content: `Provide justifications for each rating.
Candidate's Resume: [${resumeText}].`
                    }
                  ],
                  response_format: { type: "json_object" },
                });
                return {
                  cvName: file.name,
                  analysis: response
                };
              })
            );

            const analysesResults = await Promise.all(analysesPromises);
            analysesResults.forEach(res => console.log(res.analysis.choices))
            const analyses = analysesResults.map(result => ({
              cvName: result.cvName,
              analysis: JSON.parse(result.analysis.choices[0].message.content)
            }));

            res.json({ analyses });

          } catch (error) {
            console.error(error);
            res.status(500).send('Error processing resumes');
          }
        }

        async createQuestions(req, res) {
          try {
            const vacancyText = req.body.vacancyDescription;
            const responseLang = req.body?.responseLang ?? 'ukrainian';

            pdf(req.files.cv.data).then(async function (data) {
              const chatCompletion = await openai.chat.completions.create({
                model: "gpt-3.5-turbo-1106",
                messages: [
                  {
                    role: "system",
                    content: `As an HR expert, you are tasked with evaluating a
candidate's resume for a specific job vacancy. The job vacancy details are as
follows: [${vacancyText}]. Analyze the resume in the following categories:\n1.
Match of Professional Skills to Job Requirements.\n2. Relevance of Work
Experience to the Role.\n3. Soft Skills: Communication, Adaptability, Critical
Thinking.\n4. Cultural Fit with Corporate Values and Culture.\n\nProvide your
evaluation in a JSON format, following the pattern provided:
[${qestionResponseObject}]. Ensure that the property names in your response match
those in the pattern, and the responses are in the specified language
[${responseLang}].\n\nAfter the evaluation, generate a list of questions for the
candidate to help them improve their skills and better align with the job's
demands. The candidate's resume details are: [${data.text}]. Format these
questions in a clear and concise manner, following the specified language
requirements.`
```

```
              },
            ],
            response_format: { type: "json_object" },
          });
          res.json({ analysis: chatCompletion.choices[0] });
        }).catch(error => {
          console.error(error);
          res.status(500).send('Error parsing PDF');
        });

      } catch (error) {
        console.error(error);
        res.status(500).send('Error processing resume');
      }
    }
  }
```

AnalyzerRouter.js

```
import AnalyzerController from '../controllers/AnalyzerController';

const express = require('express');
const multer = require('multer');

var analyzerController = new AnalyzerController();
var router = express.Router();

const analyzerRoute = '/api/analyzer';
const statusRoute = analyzerRoute + '/status';
const analyzeResumeRoute = analyzerRoute + '/resume';
const analyzeResumesRoute = analyzerRoute + '/resumes';

router.get(statusRoute, analyzerController.getStatus);

router.post(analyzeResumeRoute, analyzerController.createQuestions);
router.post(analyzeResumesRoute, analyzerController.analyzeResumes);

module.exports = router;
```

WebInterface

AnalisisResult.js

```
import MaterialTable, { MTableToolbar } from 'material-table';
import React, {useEffect, useState} from 'react';
import DownloadIcon from '@mui/icons-material/Download';
import PreviewIcon from '@mui/icons-material/Preview';
import SummarizeIcon from '@mui/icons-material/Summarize';
import QuizIcon from '@mui/icons-material/Quiz';
import shortid from 'shortid';

import { saveAs } from 'file-saver';
import { useNavigate } from 'react-router-dom';

import ResumeScoreJustificationComponent from
'../ResumeScoreJustificationComponent/ResumeScoreJustificationComponent';
```

```
      import resources from '../../resources/Resource';
      import ResumesService from '../../services/ResumesService';
      import config from '../../config';
      import {ResumesKeyWordsStateContext} from
'../Context/ResumesKeyWordsContext';
      import Loader from '../Loader/Loader'
      import ResumeHistoryComponent from
'../ResumeHistoryComponent/ResumeHistoryComponent'
      import authService from '../../services/AuthService';
      import service from '../../services/Service';
      import FetchError from '../../utils/CustomErrors/FetchError';
      import { notifyErrorMessage } from '../../utils/Helper';
      import { useSelector } from 'react-redux';

      import 'react-toastify/dist/ReactToastify.css';
      import { options } from 'dropzone';

      export default function ResumesMaterialTable() {
          const [loading, setLoading] = useState(false);
          const navigate = useNavigate();

          const [noResultsMessage, setNoResultsMessage] = React.useState('No
analyses data were found.');
          const formData = useSelector((state) => state.formData);

          const columns = [
            {
              title: 'CV file name',
              field: 'cvName',
            },
            {
              title: 'E-mail adress',
              field: 'email',
            },
            {
              title: 'Candidate full name',
              field: 'fullName',
            },
            {
              title: 'Professional Skills Match to Job Requirements',
              field: 'professionalSkillsMatch',
            },
            {
              title: 'Relevance of Work Experience to the Role',
              field: 'workExperienceRelevance',
            },
            {
              title: 'Soft Skills',
              field: 'softSkills',
            },
            {
              title: 'Cultural Fit with Corporate Values and Culture',
              field: 'culturalFit',
            },
            {
              title: 'Overall Potential of the Candidate',
              field: 'overallCandidatePotential',
            },
            {
```

```
            title: 'Overall Candidate Rating',
            field: 'overallCandidateRating',
          },
        ]

        const candiatesData = formData?.analyses?.map(candidate => {
          const candidateData = { ...candidate, ...candidate.analysis };

          candidate.analysis.scores.forEach(score => {
            const column = columns.find(col => col.title === score.category);
            if (column) {
              candidateData[column.field] = score.score;
            }
          });
          return candidateData;
        }) ?? [];

        const sortedCandidatesData = candiatesData.sort((a, b) => {
          const scoreA = a["overallCandidateRating"];
          const scoreB = b["overallCandidateRating"];

          return scoreB - scoreA;
        });

        const getRowBackgroundColors = (rating) => {
          const parsedRating = parseInt(rating);

          if (isNaN(parsedRating)) {
            return "rgb(255, 0, 0)";
          }

          const minRating = 50;
          const redColor = [200, 0, 0];
          const greenColor = [0, 255, 0];

          if (parsedRating >= 80) {
            return `rgb(${greenColor[0]}, ${greenColor[1]},
${greenColor[2]})`;
          } else if (parsedRating >= minRating) {
            const ratio = (parsedRating - minRating) / (80 - minRating);
            const interpolatedColor = redColor.map((channel, index) => {
              const greenChannel = greenColor[index];
              return Math.round(channel + ratio * (greenChannel - channel));
            });
            return `rgb(${interpolatedColor[0]}, ${interpolatedColor[1]},
${interpolatedColor[2]})`;
          } else {
            return "rgb(190, 0, 0)";
          }
        };

        return (
            <>
                {loading && <Loader />}
                <MaterialTable
                    localization={{
                        body: {
                            emptyDataSourceMessage: noResultsMessage
                        }
```

```
                        }}
                        options={{
                            actionsColumnIndex: -1,
                            thirdSortClick: false,
                            headerStyle: {
                                position: "sticky",
                                top: "0",
                            },
                            maxBodyHeight: "85vh",
                            showTitle: false,
                            toolbar: true,
                            rowStyle: (rowData) => {
                              return {
                                fontFamily: "Mulish-Regular",
                                backgroundColor:
getRowBackgroundColors(rowData.overallCandidateRating),
                              };
                            },
                        }}
                        components={{
                            Toolbar: (props) => (
                                <div className='react-table-toolbar-custom'>
                                    <div>
                                        <div className='h3'>
                                            {resources.AlalisisTableTitle}
                                        </div>
                                    </div>
                                    <MTableToolbar {...props} />
                                </div>
                            ),
                        }}
                        columns={columns}
                        data={sortedCandidatesData}
                        actions={
                          [
                            {
                                icon:() => <PreviewIcon/>,
                                tooltip: 'Preview Resume',
                                onClick: async (event, rowData) => {
                                    navigate(`/resume-
viewer/${rowData.id}?historyFile=${rowData.fileKey}`);
                                }
                            },
                            {
                                icon: () => <DownloadIcon/>,
                                tooltip: 'Save Resume file',
                                onClick: async (event, rowData) => {
                                    try {
                                        const file = await
ResumesService.downloadFileById(rowData.fileKey)
                                        saveAs(file.file, file.filename);
                                    } catch (err) {
                                        notifyErrorMessage(err);
                                    }
                                }
                            }
                        ]}
                        detailPanel={[
                          {
```

```
                        tooltip: 'Show Justification',
                        render: rowData => {
                          return  <ResumeScoreJustificationComponent
cvName={rowData.cvName} color={rowData.backgroundColor}/>;
                        },
                      },
                      {
                        icon: () => <QuizIcon/>,
                        tooltip: 'Show questions',
                        render: rowData =>
                        <div>
                        {rowData.questions.length > 0 ? (
                          <ul>
                            {rowData.questions.map(keyWord => (
                              <li key={shortid.generate()} className='keyWord-
in-rectangle'>
                                {keyWord}
                              </li>
                            ))}
                          </ul>
                        ) : (
                          <p>-</p>
                        )}
                      </div>
                      },
                      {
                        icon: () => <SummarizeIcon/>,
                        tooltip: 'Show Recomandation',
                        render: rowData => {
                          return (
                            <div
                            style={{
                              fontSize: 16,
                              textAlign: 'center',
                              color: 'black',
                            }}>
                              {rowData.recommendations}
                          </div>
                          )
                        },
                      },
                    ]}
                    onRowClick={(event, rowData, togglePanel) =>
togglePanel()}
                  />
              </>
          );
      }
```

ResumeScoreJustificationComponent

```
      import React from 'react';
      import { useState } from 'react';
      import { useNavigate } from 'react-router-dom';
      import MaterialTable from 'material-table';
      import DownloadIcon from '@mui/icons-material/Download';
      import PreviewIcon from '@mui/icons-material/Preview';
      import { saveAs } from 'file-saver';
```

```
import { useSelector } from 'react-redux';

import ResumesService from "../../services/ResumesService";
import Loader from "../Loader/Loader";
import { notifyErrorMessage } from '../../utils/Helper';

import 'react-toastify/dist/ReactToastify.css';

const columns = [
    {
        title: 'Score Name',
        field: 'category',
    },
    {
        title: 'Score',
        field: 'score',
    },
    {
        title: 'Justification',
        field: 'justification',
        minWidth: 170
    },
];

export default function ResumeHistoryComponent({ cvName, color }) {
    const [loading, setLoading] = useState(false);
    const formData = useSelector((state) => state.formData);

    const scoreJustification = formData?.analyses?.find(candidate =>
candidate.cvName === cvName).analysis.scores ?? [];

    console.log(color);
    return (
        <>
            {loading && <Loader/>}
            <MaterialTable
                columns={columns}
                data={scoreJustification}
                options={{
                    actionsColumnIndex: -1,
                    toolbar: false,
                    search: false,
                    emptyRowsWhenPaging: false,
                    paging: false,
                    header: false,
                    headerStyle: {
                        position: "sticky",
                        top: "0"
                    },
                }}
            />
        </>
    );
}
```

RexumeAnalyzer.js

```
import { useCallback, useEffect, useState } from 'react'
```

```
import { useDropzone } from 'react-dropzone'
import { ArrowUpTrayIcon, XMarkIcon } from '@heroicons/react/24/solid'
import { notifyErrorMessage } from '../../utils/Helper';
import Loader from '../Loader/Loader'
import config from '../../config';
import { useDispatch } from 'react-redux';
import { saveFormData } from '../../storage/actions';
import { useNavigate } from 'react-router-dom';

import './RezumeAnalyzer.css';
const Dropzone = ({ className }) => {
  const [files, setFiles] = useState([])
  const [rejected, setRejected] = useState([])
  const [jobDescription, setJobDescription] = useState('');
  const [loading, setLoading] = useState(false);
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const onDrop = useCallback((acceptedFiles, rejectedFiles) => {
    if (acceptedFiles?.length) {
      setFiles(previousFiles => [
        ...previousFiles,
        ...acceptedFiles.map(file =>
          Object.assign(file, { preview: URL.createObjectURL(file) })
        )
      ])
    }

    if (rejectedFiles?.length) {
      setRejected(previousFiles => [...previousFiles, ...rejectedFiles])
    }
  }, [])

  const { getRootProps, getInputProps, isDragActive } = useDropzone({
    accept: {
      '.pdf': []
    },
    onDrop
  })

  useEffect(() => {
    // Revoke the data uris to avoid memory leaks
    return () => files.forEach(file => URL.revokeObjectURL(file.preview))
  }, [files])

  const removeFile = name => {
    setFiles(files => files.filter(file => file.name !== name))
  }

  const removeAll = () => {
    setFiles([])
    setRejected([])
  }

  const removeRejected = name => {
    setRejected(files => files.filter(({ file }) => file.name !== name))
  }

  const handleSubmit = async e => {
```

```
          e.preventDefault()

          if (!files?.length) return
          if (!jobDescription || jobDescription === '') return

          const formData = new FormData()
          files.forEach(file => formData.append('cv', file))
          formData.append('vacancyDescription', jobDescription)

          const URL = config.REZUME_ANALYZER_API_ADDRESS + config.ANALYZER +
config.RESUMES;
          setLoading(true);

          try {
              const response = await fetch(URL, {
                  mode: 'cors',
                  method: 'POST',
                  body: formData
              });

              if (!response.ok) {
                  throw new Error(`HTTP error! status: ${response.status}`);
              }
              const data = await response.json();
              dispatch(saveFormData(data));
              navigate('/analisis-result');
          } catch (error) {
              console.error("Error while fetching: ", error);
          } finally {
              setLoading(false);
          }
        }

        return (
          <div>
          {loading && <Loader />}
          <section className='section'>
            <div className='container'>
              <h1 className='title text-3xl font-bold'>Upload Resumes To
Analyze</h1>
              <form onSubmit={handleSubmit}>
                <div className='mt-10 overflow-hidden'>
                  <label for="message" class="block mb-2 text-sm font-medium
text-gray-900">Field for entering job description</label>
                  <textarea
                    id="message"
                    rows="10"
                    className="block p-2.5 w-full text-sm text-gray-900 bg-gray-
50 rounded-lg border border-gray-300 focus:ring-blue-500 focus:border-blue-500
resize: none;"
                    placeholder="Write your vacansy describe here..."
                    type="text"
                    value={jobDescription}
                    onChange={(e) => setJobDescription(e.target.value)}
                  ></textarea>
                </div>
                <div
                  {...getRootProps({
                    className: 'p-16 mt-10 border border-neutral-200'
```

```
                  })}
                >
                  <input {...getInputProps()} />
                  <div className='flex flex-col items-center justify-center gap-
4'>
                    <ArrowUpTrayIcon className='w-5 h-5 fill-current' />
                    {isDragActive ? (
                      <p>Drop the resumes here ...</p>
                    ) : (
                      <p>Drag & drop resumes here, or click to select
resumes</p>
                    )}
                  </div>
                </div>

                {/* Preview */}
                <section className='mt-10'>
                  <div className='flex gap-4'>
                    <button
                      type='button'
                      onClick={removeAll}
                      className='mt-1 text-[12px] uppercase tracking-wider font-
bold text-neutral-500 border border-secondary-400 rounded-md px-3 hover:bg-
secondary-400 hover:text-white transition-colors'
                    >
                      Remove all files
                    </button>
                    <button
                      type='submit'
                      className='ml-auto mt-1 text-[12px] uppercase tracking-
wider font-bold text-neutral-500 border border-purple-400 rounded-md px-3
hover:bg-purple-400 hover:text-white transition-colors'
                    >
                      Send resumes to analysis
                    </button>
                  </div>

                  {/* Accepted files */}
                  <h3 className='title text-lg font-semibold text-neutral-600
mt-10 border-b pb-3'>
                    Accepted Files
                  </h3>
                  <ul className='mt-6 grid grid-cols-1 sm:grid-cols-2 md:grid-
cols-3 lg:grid-cols-4 xl:grid-cols-6 gap-10'>
                    {files.map(file => (
                      <li key={file.name} className='relative h-20 w-40 rounded-
md shadow-lg'>
                                              <div>
                          <button
                            type='button'
                            className='w-7 h-7 border border-secondary-400 bg-
secondary-400 rounded-full flex justify-center items-center absolute -top-3 -
right-3 hover:bg-white transition-colors'
                            onClick={() => removeFile(file.name)}
                          >
                            <XMarkIcon className='w-5 h-5 fill-white hover:fill-
secondary-400 transition-colors' />
                          </button>
```

```
                        <p className='mt-2 text-neutral-500 text-[12px] font-
medium'>
                          {file.name}
                        </p>
                        </div>
                      </li>
                    ))}
                  </ul>

                  {/* Rejected Files */}
                  <h3 className='title text-lg font-semibold text-neutral-600
mt-24 border-b pb-3'>
                    Rejected Files
                  </h3>
                  <ul className='mt-6 flex flex-col'>
                    {rejected.map(({ file, errors }) => (
                      <li key={file.name} className='flex items-start justify-
between'>
                        <div>
                          <p className='mt-2 text-neutral-500 text-sm font-
medium'>
                            {file.name}
                          </p>
                          <ul className='text-[12px] text-red-400'>
                            {errors.map(error => (
                              <li key={error.code}>{error.message}</li>
                            ))}
                          </ul>
                        </div>
                        <button
                          type='button'
                          className='mt-1 py-1 text-[12px] uppercase tracking-
wider font-bold text-neutral-500 border border-secondary-400 rounded-md px-3
hover:bg-secondary-400 hover:text-white transition-colors'
                          onClick={() => removeRejected(file.name)}
                        >
                          Remove
                        </button>
                      </li>
                    ))}
                  </ul>
                </section>
              </form>
            </div>
          </section>
          </div>
        )
      }

      export default Dropzone
```

ResumeCreateUpdateModal

```
      import React, { useState, useEffect, useContext } from 'react';
      import { Modal, Button, Row, Col, Form } from 'react-bootstrap';
      import { toast } from 'react-toastify';
      import { useSelect, useInput, useFiles } from '../../hooks/cunstomHooks';
      import service from '../../services/Service';
```

```
      import {ResumesKeyWordsStateContext} from
'../Context/ResumesKeyWordsContext';
      import ResumesService from '../../services/ResumesService';
      import config from '../../config';
      import resources from '../../resources/Resource';
      import Loader from '../Loader/Loader'
      import ResumeNameInput from '../FormInputs/ResumeNameInput';
      import ResumeDescriptionInput from '../FormInputs/ResumeDescriptionInput';
      import ResumeKeyWordInput from '../FormInputs/ResumeKeyWordInput';
      import ResumeFileInput from '../FormInputs/ResumeFileInput';
      import AuthService from '../../services/AuthService';
      import { notifyErrorMessage } from '../../utils/Helper';

      import './ResumeCreateUpdateModal.css';
      import 'react-toastify/dist/ReactToastify.css';

      const notifySuccess = (mesage) => {
          toast.success(mesage, {
              position: toast.POSITION.TOP_RIGHT
          });
      };

      const toBase64 = file => new Promise((resolve, reject) => {
          const reader = new FileReader();
          reader.readAsDataURL(file);
          reader.onload = () => resolve(reader.result);
          reader.onerror = error => reject(error);
      });

      const createNewResumeAsync = async (resumeToSave) => {
          const saveFileToStorageResMessage =
              await ResumesService.saveFileInStorageAsync(
                  resumeToSave.file
              );

          resumeToSave.file = saveFileToStorageResMessage?.data?.fileID;
          const saveResumeResMessage = await
ResumesService.saveResumeAsync(resumeToSave);

          const keyWords = await ResumesService.getAllKeyWordsAsync();

          notifySuccess(resources.ResumeCreateSuccessful);
          return service.composeResumeKeyWords(saveResumeResMessage, keyWords);
      };

      const createNewKeyWordAsync = async (selectedOptions) => {
          const createKeyWordsId = [];

          for (const option of selectedOptions.value) {
              if (option?.__isNew__) {
                  const createNewKeyWordResult = await
ResumesService.createNewKeyWordAsync(
                      option?.label
                  );

                  createKeyWordsId.push(createNewKeyWordResult?.id);
              } else {
                  createKeyWordsId.push(option?.value);
              }
```

```
            }

            return createKeyWordsId;
        };

        const saveResumeToStorageOrReturnOldFilekey = async (resume, fileKey) => {
            if (typeof resume === 'object') {
                const saveFileToStorageResMessage =
                    await ResumesService.saveFileInStorageAsync(
                        resume
                    );
                return saveFileToStorageResMessage?.data?.fileID;
            } else {
                return fileKey;
            }
        }

        const editExistaceResumeAsync = async (resume, resumeToSave) => {
            try {
                resumeToSave.file = await
saveResumeToStorageOrReturnOldFilekey(resumeToSave.file, resume.fileKey)
            } catch (err) {
                throw err;
            }

            const editResumeResMessage = await ResumesService.editResumeAsync(
                resumeToSave,
                resume.id
            );

            const keyWords = await ResumesService.getAllKeyWordsAsync();

            notifySuccess(resources.ResumeEtitSuccessful);
            return service.composeResumeKeyWords(editResumeResMessage, keyWords);
        };

        const checkFieldsToInvalid = (resume, resumeToSave) => {
            return !resumeToSave.fullName.isValid
            || !resumeToSave.description.isValid
            || ((!resume.id && !resumeToSave.file.isValid)
                || (!resumeToSave.file.isEmpty && !resumeToSave.file.isValid))
        }

        const hideAndClearIfUndefined = (resume, contextResumesKeyWords, hideForm,
clearForm) => {
            if (resume !== undefined) {
                const newResumes = [...contextResumesKeyWords.resumes];
                newResumes.push(resume);
                contextResumesKeyWords.resumes = newResumes;
                hideForm();
                clearForm();
            }
        }

        const ifResumesContainsResumeSplice = (resume, composedResume,
contextResumesKeyWords) => {
            const index = contextResumesKeyWords.resumes.indexOf(resume);

            if (index !== -1 && composedResume !== undefined) {
```

```
            const newResumes = [...contextResumesKeyWords.resumes];
            newResumes.splice(index, 1);
            contextResumesKeyWords.resumes = newResumes;
        }
    }

    const createOrEditResume = async (resume, resumeToSave,
contextResumesKeyWords) => {
        let composedResume = null;

        try {
            if (JSON.stringify(resume) === JSON.stringify(config.EMPTY_MODEL))
{
                composedResume = await createNewResumeAsync(resumeToSave);
            } else {
                composedResume = await editExistaceResumeAsync(resume,
resumeToSave);
                ifResumesContainsResumeSplice(resume, composedResume,
contextResumesKeyWords);
            }
        } catch (err) {
            throw err;
        }

        return composedResume;
    }

    const setAllFieldsDurty = (resumeToSave, value) => {
        resumeToSave.file.setDirty(value)
        resumeToSave.fullName.setDirty(value);
        resumeToSave.description.setDirty(value);
        resumeToSave.keyWords.setDirty(value);
    }

    const clearAllFieldsDurty = (resumeToSave, options) => {
        resumeToSave.file.setValue('');
        resumeToSave.fullName.setValue(config.EMPTY_MODEL.fullName);
        resumeToSave.description.setValue(config.EMPTY_MODEL.description);
        resumeToSave.keyWords.setValue(service.generateOptionsFromSelectedKeyW
ord(options, config.EMPTY_MODEL.keyWords));
    }

    const replaceCustomHooksInResumeToSaveToValue = (resumeToSave) => {
        return {
            file: resumeToSave.file.value,
            fullName: resumeToSave.fullName.value,
            description: resumeToSave.description.value,
            keyWords:
service.generateKeyWordFromOptions(resumeToSave.keyWords.value),
        };
    }

    export default function ResumeCreateUpdateModal({ resume, show, onHide })
{
        const contextResumesKeyWords =
useContext(ResumesKeyWordsStateContext);
        const [loading, setLoading] = useState(false);
        const [options, setOptions] =
useState(service.generateOptionsFromKeyWord([]));
```

```
          const selectedOptions = useSelect(
              service.generateOptionsFromSelectedKeyWord(options,
resume.keyWords), []);
          const fullName = useInput(resume.fullName, {isEmpty: true, minLength:
4});
          const description = useInput(resume.description, {isEmpty: true,
minLength: 2});

          const file = useFiles(resume.fileKey, {isEmpty: true, isFile: true});
          const [resumeToSave, setResumeToSave] = useState({});

          const getAndSetKeyWordsAsync = async () => {
              const receivedKeyWords = await
ResumesService.getAllKeyWordsAsync();
              contextResumesKeyWords.keyWords = receivedKeyWords;
          }

          useEffect(() => {
              getAndSetKeyWordsAsync();
          }, []);

          useEffect(() => {
              setOptions(service.generateOptionsFromKeyWord(contextResumesKeyWor
ds.keyWords));
          }, [contextResumesKeyWords.keyWords]);

          useEffect(() => {
              setResumeToSave({
                  fullName: fullName,
                  description: description,
                  file: file,
                  keyWords: selectedOptions
              });
          }, [fullName.value, description.value, selectedOptions.value,
file.value,
                  fullName.isValid, description.isValid,
selectedOptions.isValid, file.isValid
              ])

          useEffect(() => {
              fullName.setValue(resume.fullName);
              description.setValue(resume.description);
              selectedOptions.setValue(service.generateOptionsFromSelectedKeyWor
d(options, resume.keyWords));

          }, [resume, resume.fullName, resume.description,
resume.previewBlobKey, resume.keyWords])

          const clearForm = () => {
              clearAllFieldsDurty(resumeToSave, options);
              setAllFieldsDurty(resumeToSave, false);
          }

          const isNotValid = () => {
              return checkFieldsToInvalid(resume, resumeToSave);
          }

          const handleSubmitAsync = async (evt) => {
```

```
            evt.preventDefault();
            const isValid = isNotValid();
            if (isValid) {
                setAllFieldsDurty(resumeToSave, true);
            } else {
                try {
                    setLoading(true);
                    const createNewKeyWordResult = await
createNewKeyWordAsync(selectedOptions);
                    await getAndSetKeyWordsAsync();

                    const replacedResumeToSave =
replaceCustomHooksInResumeToSaveToValue(resumeToSave);
                    replacedResumeToSave.keyWords = createNewKeyWordResult;
                    replacedResumeToSave.userId =
AuthService.getCurrentUser()?.userId;
                    const composedResume = await createOrEditResume(resume,
replacedResumeToSave, contextResumesKeyWords);

                    hideAndClearIfUndefined(composedResume,
contextResumesKeyWords, onHide, clearForm)

                } catch (err) {
                    notifyErrorMessage(err);
                } finally {
                    setLoading(false);
                }
            }
        }

        return (
            <div>
                {loading && <Loader />}
                <Modal
                    show={show}
                    onHide={onHide}
                    size="lg"
                    aria-labelledby="contained-modal-title-vcenter"
                    centered>
                    <Modal.Header closeButton>
                        <Modal.Title id='contained-modal-title-vcenter'>
                            {resume.id ? (<p>Updating resume</p>) :
(<p>Creating resume</p>)}
                        </Modal.Title>
                    </Modal.Header>
                    <Modal.Body>
                        <div className='container'>
                            <Row>
                                <Col>
                                    <Form onSubmit={handleSubmitAsync}>
                                        <div className='form-group'>
                                            <Form.Group
controlId='ResumeModal'>
                                                <Form.Group
controlId='ResumeModal.InputFields'>
                                                    <ResumeNameInput
fullName={fullName} />
                                                    <ResumeDescriptionInput
description={description} />
```

```
                                                    <ResumeKeyWordInput
options={options} selectedOptions={selectedOptions}  />
                                                </Form.Group>
                                                <Form.Group className="mb-3" >
                                                    <ResumeFileInput
file={file} />
                                                </Form.Group>
                                                <Form.Group className="mb-3
create-clear-form-block" >
                                                    <Button
                                                        className='button-
submit create-clear-btn'
                                                        data-testid="create-
or-update-btn"
                                                        id='CreateOrUpdateButt
on'
                                                        variant='primary'
                                                        type='submit'
                                                    >
                                                        {resume.id ?
("Update") : ("Create")}
                                                    </Button>
                                                    <Button
                                                        className='button
create-clear-btn'
                                                        data-testid="clear-
form-btn"
                                                        id='clearFormButton'
                                                        variant='primary'
                                                        type='reset'
                                                        onClick={event =>
clearForm()}
                                                    >
                                                        Clear
                                                    </Button>
                                                </Form.Group>
                                            </Form.Group>
                                        </div>
                                    </Form>
                                </Col>
                            </Row>
                        </div>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button data-testid="close-btn"
onClick={onHide}>Close</Button>
                    </Modal.Footer>
                </Modal>
            </div>
        );
    }
```

ДОДАТОК Б

Текст вакансії:

Similarweb is the leading digital intelligence platform used by over 3500 global customers. Our wide range of solutions power the digital strategies of companies like Google, eBay, and Adidas. We help our customers succeed in today's digital world by giving them access to data-driven insights, competitive benchmarks, strategic analysis, and more. In 2021, we went public on the New York Stock Exchange, and we haven't stopped growing since!

We're looking for a Full Stack Engineer (.Net/React) to design, code and manage Similarweb's large-scale and highly available services, using cutting-edge technologies & design. This role will report to Team Manager, R&D.

Why is this role so important at Similarweb?

We are a company that is driven by innovation, which means that we are constantly building and developing new features and products to meet customers' needs. As a Full-Stack developer, you will have the opportunity to work with the most cutting-edge technologies, to solve complex challenges, and create high-quality and scalable code. Your work will have a direct impact on customer satisfaction and our business objectives.

So, what will you be doing all day?

Designing, coding and managing large scale, highly available services using cutting edge technologies (.NET Core, React, TypeScript, Redis, Docker, etc.)

Leading decision-making & design of APIs, mechanisms and abstractions

Building new features and directly impact the future of our product from UI to DB

Writing (a lot of) high-quality, functional code — backend and frontend

Performing code reviews, evaluating implementations, and providing feedback

This is the perfect job for someone who:

Has a minimum of 3+ years of experience in software engineering

Has demonstrated experience in backend development (.NET)

Has experience in frontend development (React)

Knows how to architect and design scalable and high-performance complex distributed systems

Has previous experience with databases (MySQL, PostgreSQL, Redis, Elasticsearch)

Has excellent verbal and written communication skills

Is comfortable taking on challenges and learning new technologies

Is a team player who can also work independently

Would be a plus:

Experience working with message brokers (SQS/SNS, RabbitMQ, Kafka)

Why you'll love being a Similarwebber:

You'll actually love the product you work with: Our customers aren't our only raving fans. When we asked our employees why they chose to come work at Similarweb, 99% of them said "the product." Imagine how exciting your job is when you get to work with the most powerful digital intelligence platform in the world.

You'll find a home for your big ideas: We encourage an open dialogue and empower employees to bring their ideas to the table. You'll find the resources you need to take initiative and create meaningful change within the organization.

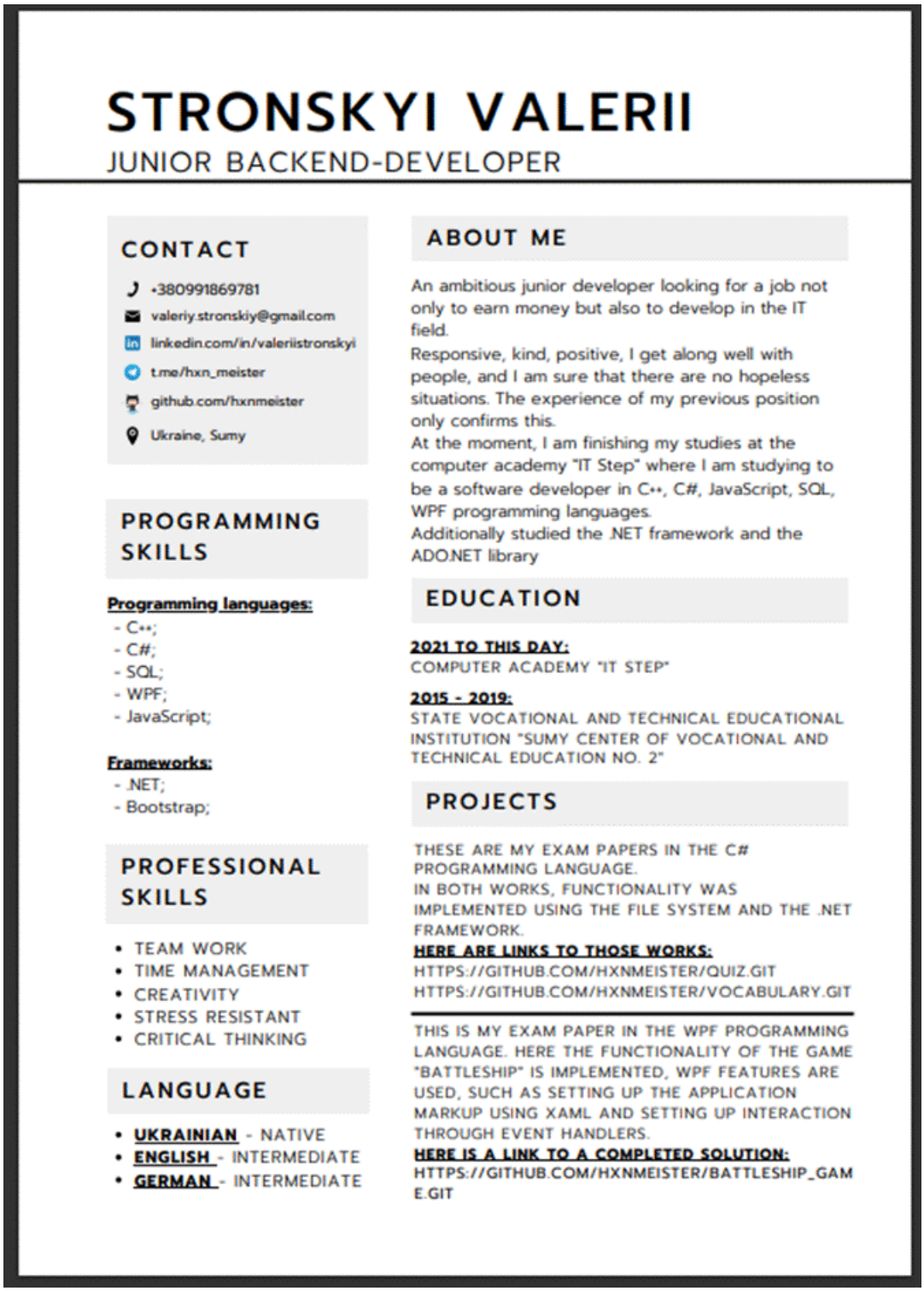
We offer competitive perks & benefits: We take your well-being seriously, and offer competitive compensation packages to all employees. We also put a strong emphasis on community, with regular team outings and happy hours.

You can grow your career in any direction you choose: Interested in becoming a VP or want to transition into a different department? Whether it's Career Week,

personalized coaching, or our ongoing learning solutions, you'll find all the tools and opportunities you need to develop your career right here.

Diversity isn't just a buzzword: People want to work in a place where they can be themselves. We strive to create a workplace that is reflective of the communities we serve, where everyone is empowered to bring their full, authentic selves to work. We are committed to inclusivity across race, gender, ethnicity, culture, sexual orientation, age, religion, spirituality, identity and experience. We believe our culture of equality and mutual respect also helps us better understand and serve our customers in a world that is becoming more global, more diverse, and more digital every day.

ДОДАТОК В



STRONSKYI VALERII

JUNIOR BACKEND-DEVELOPER

CONTACT

+380991869781
valeriy.stronskiy@gmail.com
linkedin.com/in/valeriistronskyi
t.me/hxn_meister
github.com/hxnmeister
Ukraine, Sumy

PROGRAMMING SKILLS

Programming languages:
- C++;
- C#;
- SQL;
- WPF;
- JavaScript;

Frameworks:
- .NET;
- Bootstrap;

PROFESSIONAL SKILLS

- TEAM WORK
- TIME MANAGEMENT
- CREATIVITY
- STRESS RESISTANT
- CRITICAL THINKING

LANGUAGE

- **UKRAINIAN** - NATIVE
- **ENGLISH** - INTERMEDIATE
- **GERMAN** - INTERMEDIATE

ABOUT ME

An ambitious junior developer looking for a job not only to earn money but also to develop in the IT field.
Responsive, kind, positive, I get along well with people, and I am sure that there are no hopeless situations. The experience of my previous position only confirms this.
At the moment, I am finishing my studies at the computer academy "IT Step" where I am studying to be a software developer in C++, C#, JavaScript, SQL, WPF programming languages.
Additionally studied the .NET framework and the ADO.NET library

EDUCATION

2021 TO THIS DAY:
COMPUTER ACADEMY "IT STEP"

2015 - 2019:
STATE VOCATIONAL AND TECHNICAL EDUCATIONAL INSTITUTION "SUMY CENTER OF VOCATIONAL AND TECHNICAL EDUCATION NO. 2"

PROJECTS

THESE ARE MY EXAM PAPERS IN THE C# PROGRAMMING LANGUAGE.
IN BOTH WORKS, FUNCTIONALITY WAS IMPLEMENTED USING THE FILE SYSTEM AND THE .NET FRAMEWORK.
HERE ARE LINKS TO THOSE WORKS:
HTTPS://GITHUB.COM/HXNMEISTER/QUIZ.GIT
HTTPS://GITHUB.COM/HXNMEISTER/VOCABULARY.GIT

THIS IS MY EXAM PAPER IN THE WPF PROGRAMMING LANGUAGE. HERE THE FUNCTIONALITY OF THE GAME "BATTLESHIP" IS IMPLEMENTED, WPF FEATURES ARE USED, SUCH AS SETTING UP THE APPLICATION MARKUP USING XAML AND SETTING UP INTERACTION THROUGH EVENT HANDLERS.
HERE IS A LINK TO A COMPLETED SOLUTION:
HTTPS://GITHUB.COM/HXNMEISTER/BATTLESHIP_GAME.GIT

Рисунок 0.1 - Резюме 1 - 1690721646239.pdf

Volodymyr Ivaniuk

React & React Native Developer

Experienced React & React Native developer with 1.5 years of commercial experience. Result-oriented developer, aimed at creating eye-catching and user-friendly solution with strong communication and teamwork skills.

ivaniuk.7531@gmail.com
+380501007198
Ivano-Frankivsk, Ukraine
linkedin.com/in/volodymyr-ivaniuk-573577263
github.com/SelfDestroyer
@ivaniuk7531

SKILLS

- JavaScript, TypeScript
- HTML, CSS/SCSS/SASS
- React, React Native, Next.js
- Redux, Redux Toolkit, RTK Query, Redux Saga, Redux Thunk
- Agile/Scrum
- Eslint, Prettier
- Jira
- Sentry
- Push notifications (Notifee, FMC)
- Rest API
- MySQL, PostgreSQL
- Styled Component
- Unit and Integration Tests

WORK EXEPERIENCE

React / React Native Developer
Expletech
Ivano-Frankivsk, Ukraine
Feb. 2022 – May 2023
Responsibilities: Develop web/mobile applications using React & React Native

React / React Native developer
Techlexity
Ukraine
May. 2023 – Sep. 2023
Responsibilities: Responsibilities: Develop mobile application from scratch using React Native, integration of third-party services (Google, Apple)

LANGUAGES

English – Intermediate
Ukrainian – Native

CERTIFICATES

Feb. 2023 – Mar. 2023
React Native - The Practical Guide [2023]
Credential ID UC-7dab99c5-a7ef-49cb-9f8b-6947d986f1cCredential

PET PROJECTS

Memory Game
Technologies: JavaScript, HTML, CSS.
https://selfdestroyer.github.io/memory-game/

Meals App
Technologies: TypeScript, React, React Native, Eslint, Prettier.

EDUCATION

EPAM University Program
Front-End Autumn 2021
Sep. 2021 – Nov. 2021

SoftServe React Marathon
React online marathon
Feb. 2022 – May. 2022

Internship
Softjourn: Software Development Company

Рисунок 0.2 - Резюме 2 - 1695117515643.pdf сторінка 1

https://github.com/SelfDestroyer/meals-app

Expence Tracker App
Technologies: TypeScript, React, React Native, Eslint, Prettier.
https://github.com/SelfDestroyer/expense-tracker-app

ToDo App
Technologies: React, JavaScript, HTML, CSS.
https://github.com/SelfDestroyer/todo-app

Jan. 2022 – Apr. 2022

Bachelor`s degree
Ivano-Frankivsk National Technical University Gas and Oil Computer Engineering
Sep. 2019 – Jun. 2023

Master's degree
Ivano-Frankivsk National Technical University Gas and Oil Computer Engineering
Sep. 2023 – Present

PROFECCIONAL EXPERIENCE

Graberly
Mar. 2022 – Sep. 2022
https://graberly-fe-prod.herokuapp.com/
Graberly - is a unique project where businesses like shops, cafes, and restaurants can generate QR codes for their customers to leave reviews. Technology stack includes: NextJS, Redux Toolkit, Styled Components, ExpressJS, Sequelize, and MySQL.

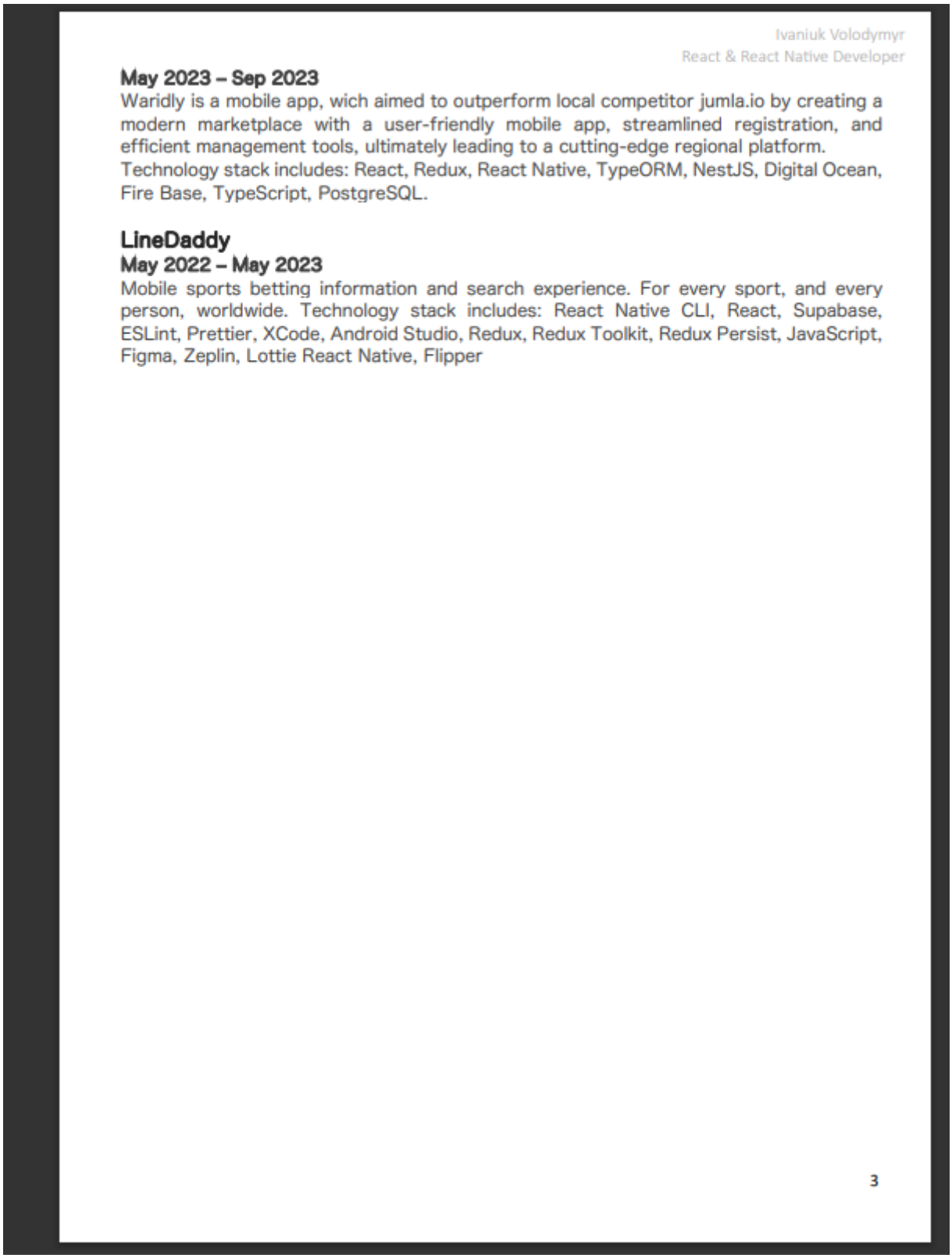
Roobi
Jan. 2023 – Mar. 2023
https://www.roobi.ae
Roobi represents a grocery delivery service operating in Dubai. It offers a range of products from various stores, allowing customers to select their preferred options. Technology stack includes: React, Redux Toolkit, RTK Query, SCSS, TypeScript, Nestjs, PostgreSQL, and TypeORM.

ReCheck
Mar. 2022 – Oct. 2022
https://recheck-candidate.com
I have created a system for sending candidates their individualized invitation links. This system also facilitates the display of user profiles and invitation forms, all managed by the administrator. Technology stack includes: React, Redux Toolkit, RTK Query, SCSS, TypeScript, PHP, Laravel.

Aroundish
May 2022 – Jan. 2023
https://play.google.com/store/apps/details? id=com.aroundish&hl=en_US&gl=US&pli=1
https://apps.apple.com/ua/app/aroundish/id1463795271
Aroundish is the innovative app to discover nearby vegan food. Join the community, connect with like-minded people, and share your favorite vegan dishes globally. Technology stack includes: React, Redux, React Native, PHP, Laravel, PostgreSQL.

Waridly

Рисунок 0.3 - Резюме 2 - 1695117515643.pdf сторінка 2



May 2023 – Sep 2023

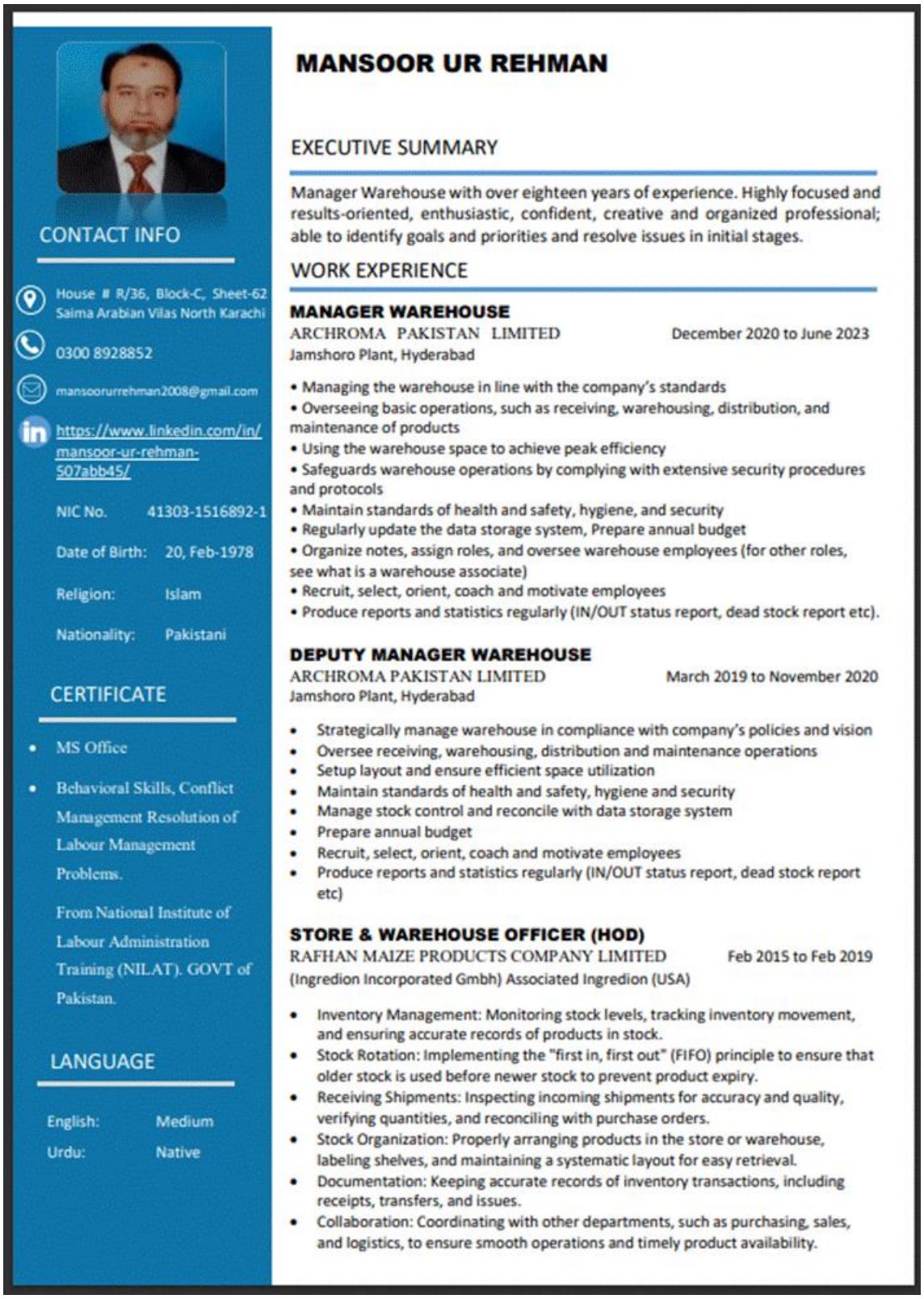
Waridly is a mobile app, wich aimed to outperform local competitor jumla.io by creating a modern marketplace with a user-friendly mobile app, streamlined registration, and efficient management tools, ultimately leading to a cutting-edge regional platform.
Technology stack includes: React, Redux, React Native, TypeORM, NestJS, Digital Ocean, Fire Base, TypeScript, PostgreSQL.

LineDaddy

May 2022 – May 2023

Mobile sports betting information and search experience. For every sport, and every person, worldwide. Technology stack includes: React Native CLI, React, Supabase, ESLint, Prettier, XCode, Android Studio, Redux, Redux Toolkit, Redux Persist, JavaScript, Figma, Zeplin, Lottie React Native, Flipper

Рисунок 0.4 - Резюме 2 - 1695117515643.pdf сторінка 3



MANSOOR UR REHMAN

CONTACT INFO

House # R/36, Block-C, Sheet-62
Saima Arabian Vilas North Karachi

0300 8928852

mansoorurrehman2008@gmail.com

https://www.linkedin.com/in/mansoor-ur-rehman-507abb45/

NIC No. 41303-1516892-1

Date of Birth: 20, Feb-1978

Religion: Islam

Nationality: Pakistani

CERTIFICATE

- MS Office
- Behavioral Skills, Conflict Management Resolution of Labour Management Problems.

  From National Institute of Labour Administration Training (NILAT). GOVT of Pakistan.

LANGUAGE

English: Medium

Urdu: Native

EXECUTIVE SUMMARY

Manager Warehouse with over eighteen years of experience. Highly focused and results-oriented, enthusiastic, confident, creative and organized professional; able to identify goals and priorities and resolve issues in initial stages.

WORK EXPERIENCE

MANAGER WAREHOUSE

ARCHROMA PAKISTAN LIMITED — December 2020 to June 2023
Jamshoro Plant, Hyderabad

- Managing the warehouse in line with the company's standards
- Overseeing basic operations, such as receiving, warehousing, distribution, and maintenance of products
- Using the warehouse space to achieve peak efficiency
- Safeguards warehouse operations by complying with extensive security procedures and protocols
- Maintain standards of health and safety, hygiene, and security
- Regularly update the data storage system, Prepare annual budget
- Organize notes, assign roles, and oversee warehouse employees (for other roles, see what is a warehouse associate)
- Recruit, select, orient, coach and motivate employees
- Produce reports and statistics regularly (IN/OUT status report, dead stock report etc).

DEPUTY MANAGER WAREHOUSE

ARCHROMA PAKISTAN LIMITED — March 2019 to November 2020
Jamshoro Plant, Hyderabad

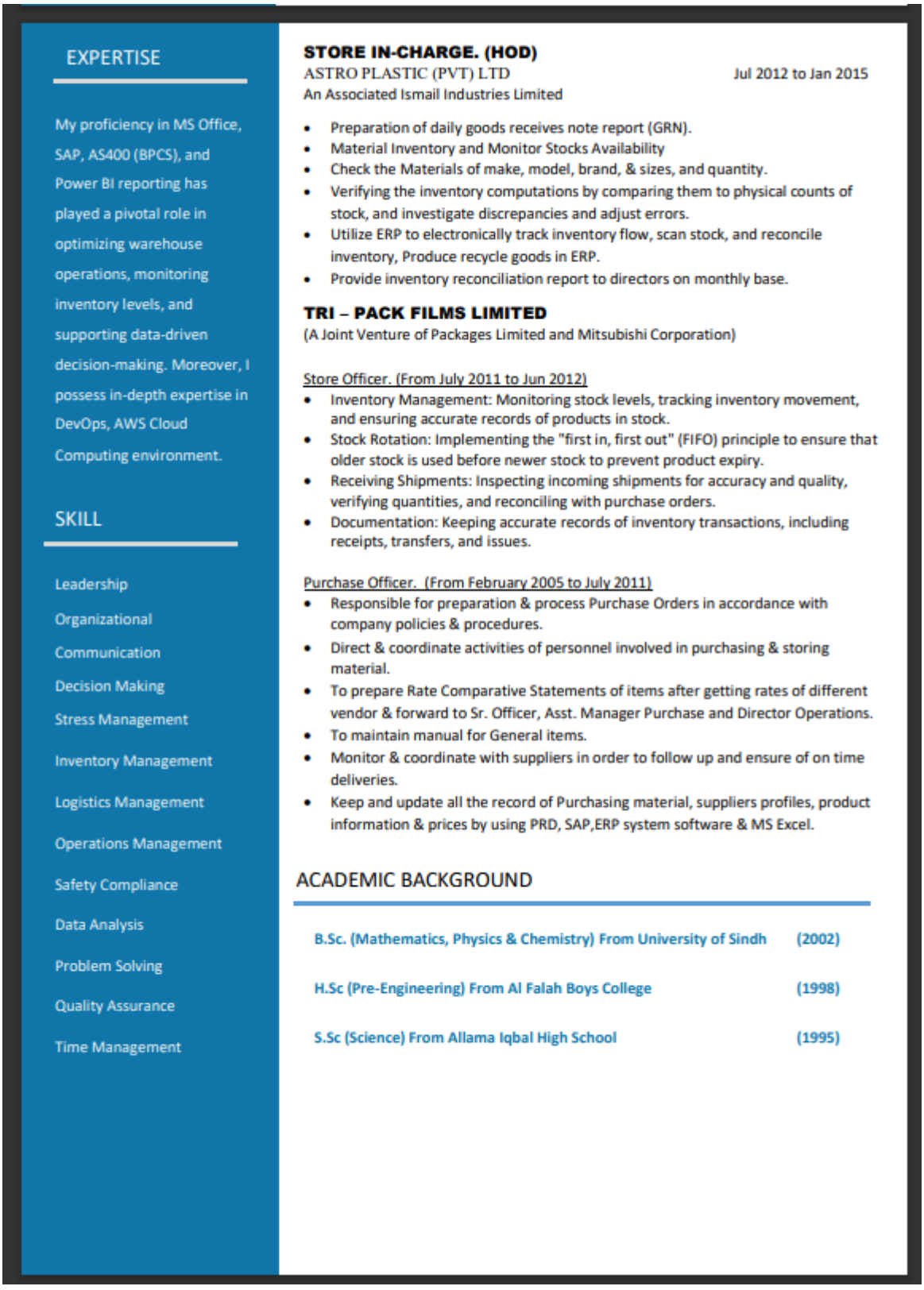
- Strategically manage warehouse in compliance with company's policies and vision
- Oversee receiving, warehousing, distribution and maintenance operations
- Setup layout and ensure efficient space utilization
- Maintain standards of health and safety, hygiene and security
- Manage stock control and reconcile with data storage system
- Prepare annual budget
- Recruit, select, orient, coach and motivate employees
- Produce reports and statistics regularly (IN/OUT status report, dead stock report etc)

STORE & WAREHOUSE OFFICER (HOD)

RAFHAN MAIZE PRODUCTS COMPANY LIMITED — Feb 2015 to Feb 2019
(Ingredion Incorporated Gmbh) Associated Ingredion (USA)

- Inventory Management: Monitoring stock levels, tracking inventory movement, and ensuring accurate records of products in stock.
- Stock Rotation: Implementing the "first in, first out" (FIFO) principle to ensure that older stock is used before newer stock to prevent product expiry.
- Receiving Shipments: Inspecting incoming shipments for accuracy and quality, verifying quantities, and reconciling with purchase orders.
- Stock Organization: Properly arranging products in the store or warehouse, labeling shelves, and maintaining a systematic layout for easy retrieval.
- Documentation: Keeping accurate records of inventory transactions, including receipts, transfers, and issues.
- Collaboration: Coordinating with other departments, such as purchasing, sales, and logistics, to ensure smooth operations and timely product availability.

Рисунок 0.5 - Резюме 3 - 1703229952270.pdf сторінка 1



EXPERTISE

My proficiency in MS Office, SAP, AS400 (BPCS), and Power BI reporting has played a pivotal role in optimizing warehouse operations, monitoring inventory levels, and supporting data-driven decision-making. Moreover, I possess in-depth expertise in DevOps, AWS Cloud Computing environment.

SKILL

Leadership
Organizational
Communication
Decision Making
Stress Management
Inventory Management
Logistics Management
Operations Management
Safety Compliance
Data Analysis
Problem Solving
Quality Assurance
Time Management

STORE IN-CHARGE. (HOD)
ASTRO PLASTIC (PVT) LTD — Jul 2012 to Jan 2015
An Associated Ismail Industries Limited

- Preparation of daily goods receives note report (GRN).
- Material Inventory and Monitor Stocks Availability
- Check the Materials of make, model, brand, & sizes, and quantity.
- Verifying the inventory computations by comparing them to physical counts of stock, and investigate discrepancies and adjust errors.
- Utilize ERP to electronically track inventory flow, scan stock, and reconcile inventory, Produce recycle goods in ERP.
- Provide inventory reconciliation report to directors on monthly base.

TRI – PACK FILMS LIMITED
(A Joint Venture of Packages Limited and Mitsubishi Corporation)

Store Officer. (From July 2011 to Jun 2012)

- Inventory Management: Monitoring stock levels, tracking inventory movement, and ensuring accurate records of products in stock.
- Stock Rotation: Implementing the "first in, first out" (FIFO) principle to ensure that older stock is used before newer stock to prevent product expiry.
- Receiving Shipments: Inspecting incoming shipments for accuracy and quality, verifying quantities, and reconciling with purchase orders.
- Documentation: Keeping accurate records of inventory transactions, including receipts, transfers, and issues.

Purchase Officer. (From February 2005 to July 2011)

- Responsible for preparation & process Purchase Orders in accordance with company policies & procedures.
- Direct & coordinate activities of personnel involved in purchasing & storing material.
- To prepare Rate Comparative Statements of items after getting rates of different vendor & forward to Sr. Officer, Asst. Manager Purchase and Director Operations.
- To maintain manual for General items.
- Monitor & coordinate with suppliers in order to follow up and ensure of on time deliveries.
- Keep and update all the record of Purchasing material, suppliers profiles, product information & prices by using PRD, SAP,ERP system software & MS Excel.

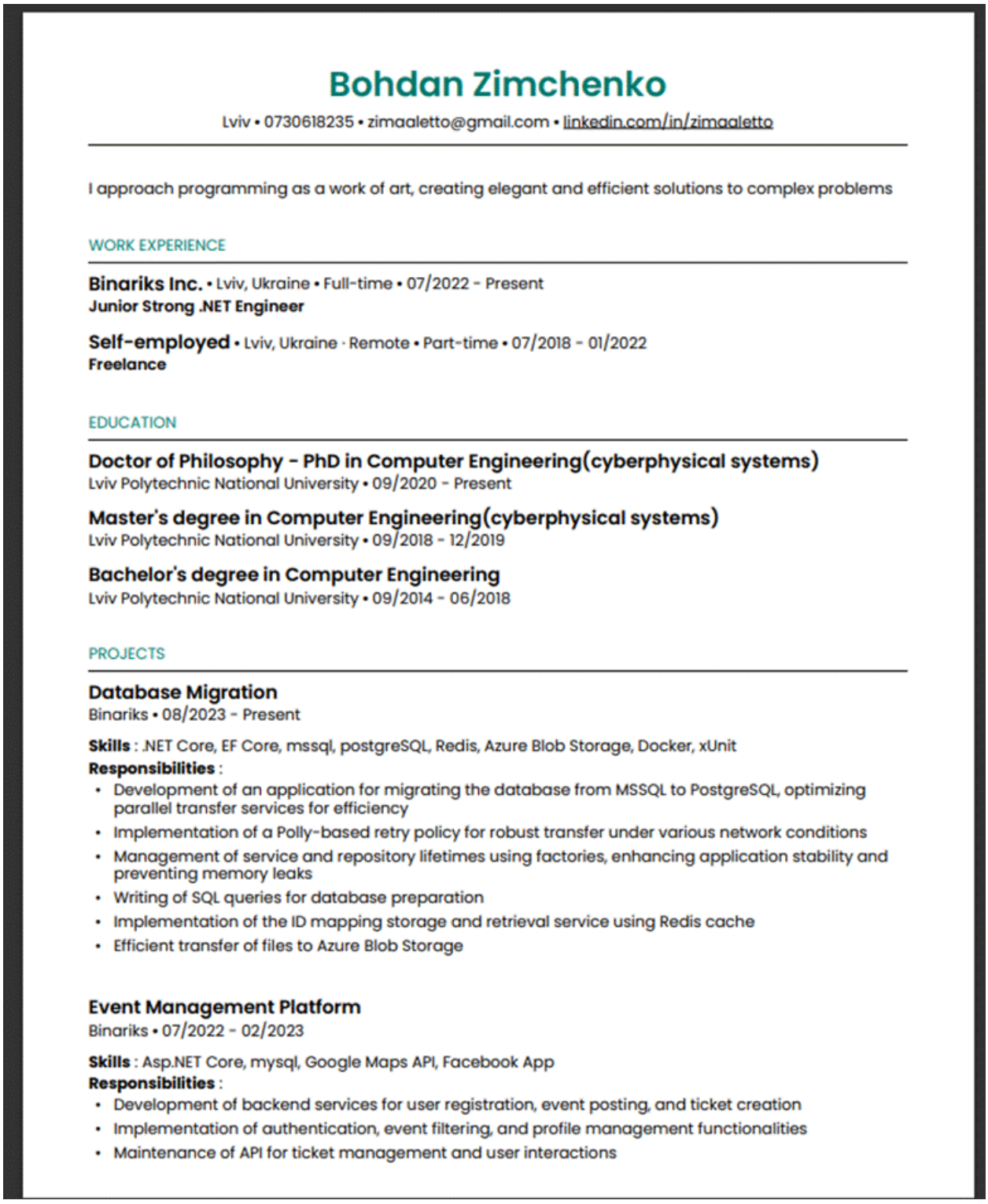
ACADEMIC BACKGROUND

B.Sc. (Mathematics, Physics & Chemistry) From University of Sindh (2002)

H.Sc (Pre-Engineering) From Al Falah Boys College (1998)

S.Sc (Science) From Allama Iqbal High School (1995)

Рисунок 0.6 - Резюме 3 - 1703229952270.pdf сторінка 2



Bohdan Zimchenko

Lviv • 0730618235 • zimaaletto@gmail.com • linkedin.com/in/zimaaletto

I approach programming as a work of art, creating elegant and efficient solutions to complex problems

WORK EXPERIENCE

Binariks Inc. • Lviv, Ukraine • Full-time • 07/2022 - Present
Junior Strong .NET Engineer

Self-employed • Lviv, Ukraine · Remote • Part-time • 07/2018 - 01/2022
Freelance

EDUCATION

Doctor of Philosophy - PhD in Computer Engineering(cyberphysical systems)
Lviv Polytechnic National University • 09/2020 - Present

Master's degree in Computer Engineering(cyberphysical systems)
Lviv Polytechnic National University • 09/2018 - 12/2019

Bachelor's degree in Computer Engineering
Lviv Polytechnic National University • 09/2014 - 06/2018

PROJECTS

Database Migration
Binariks • 08/2023 - Present

Skills : .NET Core, EF Core, mssql, postgreSQL, Redis, Azure Blob Storage, Docker, xUnit
Responsibilities :

- Development of an application for migrating the database from MSSQL to PostgreSQL, optimizing parallel transfer services for efficiency
- Implementation of a Polly-based retry policy for robust transfer under various network conditions
- Management of service and repository lifetimes using factories, enhancing application stability and preventing memory leaks
- Writing of SQL queries for database preparation
- Implementation of the ID mapping storage and retrieval service using Redis cache
- Efficient transfer of files to Azure Blob Storage

Event Management Platform
Binariks • 07/2022 - 02/2023

Skills : Asp.NET Core, mysql, Google Maps API, Facebook App
Responsibilities :

- Development of backend services for user registration, event posting, and ticket creation
- Implementation of authentication, event filtering, and profile management functionalities
- Maintenance of API for ticket management and user interactions

Рисунок 0.7 - Резюме 4 - 1703239748206.pdf сторінка 1

- Integration of social media logins via Google and Facebook
- Creation of an email notification system for event updates and ticket information

Employee Management Tool
Binariks • 03/2022 - 07/2022

Skills : Asp.NET Core, mssq, mysql, EF Core, Unit Testing
Responsibilities :
- Development of an employee management application to streamline operations for HR, accounting, and recruitment departments in a corporate setting
- Implemented backed API for employee registration, data entry, interview scheduling, working hours and vacation tracking
- Designing and managing the database

Automated Beer Pouring System
Freelance • 02/2020 - 12/2020

Skills : Python, OpenCV, Tensorflow, Anaconda, Linux
Responsibilities :
- Developed an automated beer pouring system for retail stores, leveraging machine learning and computer vision
- Integrated a TensorFlow model for accurate beer level detection, complemented by an OpenCV-based custom algorithm for real-time analysis
- Handled multiple controller connections to automate crane operations in the pouring process
- Managed IP camera streams for comprehensive real-time video analysis
- Wrote Python scripts for system operations, emphasizing error handling and system resilience
- Collaborated with the hardware team to ensure seamless integration of software with physical components

Spinner 360
Freelance • 08/2019 - 10/2019

Skills : C#, .NET Framework, WPF, Desktop Application, OpenCV, FFmpeg
Responsibilities :
- Developed a software solution interfacing with GoPro cameras via Wi-Fi, designed for remote control of camera and motor settings
- Implemented video processing features including effects, overlays, green screen processing, timeline editing, color correction, reverse playback, video concatenation, and more
- Integrated hardware binding functionality for software licensing

Automatic Logo Overlay Application for Clothes Designers
Freelance • 09/2019 - 10/2019

Skills : C++/CLI, OpenCV
Responsibilities :
- Developed a desktop application to automate logo overlay on clothing designs, enhancing fashion designers' workflow
- Implemented OpenCV-based ROI detection algorithms, identifying optimal logo placement areas on different clothing items

Рисунок 0.8 - Резюме 4 - 1703239748206.pdf сторінка 2

- Created a feature to export designs with logos into PDF format

LIDAR visualizer
Freelance • 04/2019 - 04/2019

Skills : C++, OpenGL
Responsibilities :
- Developed a 2D LIDAR visualization tool for the YDLIDAR X4 using the lidar SDK

3D Editor
Freelance • 11/2018 - 12/2018

Skills : C++, Microsoft Foundation Classes (MFC)
Responsibilities :
- Developed import/export functionalities for various 3D model formats

CERTIFICATIONS

Microsoft Certified: Azure Developer Associate
Microsoft • 05/2023 - Present

PUBLICATIONS

Microservices Infrastructure Architecture for the Cloud-Based Multi-Agent Group Decision Support Systems for Autonomous Cyberphysical Systems
CEUR Workshop Proceedings • 11/2023

An Autonomous Serverless Fuzzy Logic-Based Decision Support System for Evaluating the Reliability of a Country's Electric Power System
CEUR Workshop Proceedings • 09/2023

A Web-Based Application for Modeling and Deploying Fuzzy Logic-Based Decision Support Systems for Autonomous Cyberphysical Systems (keynote)
CEUR Workshop Proceedings • 06/2022

Рисунок 0.9 - Резюме 4 - 1703239748206.pdf сторінка 3

ДОДАТОК Г

Bohdan Kandela

.NET Developer

Backend developer. 5 years experience in IT. Main area of expertise is Rest API. Main experience with asp.net, also developed on java (android apps), node.js, php. I am looking for a company where I can develop my current skills and learn new technologies. I am highly motivated to work on projects and gain experience in microservice architecture. Unfortunately, I have not yet had the opportunity to work with that architecture.

Contact

Phone
+38(067)-210-36-89
Email
bohdan.kandela@gmail.com
Address
Ukraine, Dnipro
LinkedIn
https://www.linkedin.com/in/bohdan-kandela-014b8014a

Education

2015
Erasmus+ (Czech Republic, Prague)
Startup developing

2015 - 2016
2o EPAL Euosmou (Greece, Thessaloniki)
Informatics and Computer Engineering

2016 - 2019
IT Step (Ukraine, Dnipro)
Software development

Skills

- Asp.net (Currently working with 7.0)
- EF Core
- MSSQL, PostgreSQL
- MySQL, MongoDB (non-commercial experience)
- Memory cache, Redis
- Signal R
- REST API
- GRPC, Graph QL (non-commercial experience)
- Serilog
- Automapper
- Docker
- Sieve
- RabbitMQ (non-commercial experience)
- Fluent validation
- Swashbuckle

Language

Ukrainian (Native)
English (A2)
Greek (B2)

Experience

Oct 2017 - Jun 2018
Flagman
Junior system administrator
Part time job. Setting up systems in the offices. Setting up and working with billing system on Mikrotik (MikroBill). Technical work with networks.

Sep 2019 - Oct 2019
Metrans
Junior .NET developer
Full time job. First experience in .net development. Work consisted of writing parsers and converting data into different formats (xml, pdf, xls, etc.)

Oct 2020 - May 2022
Infovector
Middle .NET developer
Full time job. The main tasks are creating project architecture and rest api development. I used such technologies as asp.net, php, node.js, blazor, centrifugo, webrtc, docker. The company was small, and I had to take on many tasks involving different technologies, which helped me to develop the necessary skills to quickly learn new technologies. One of the interesting tasks at the company was developing a streaming service and working with cryptocurrency exchanges.

May 2022 - Now
Cloud Crew
Middle .NET developer
Freelance job. The main tasks are creating project architecture and rest api development. At the moment 3 projects have been developed. Working on demand. As a freelancer, I work on small projects such as online stores, cryptocurrency exchanges, and company information sites.

Sep 2022 - Apr 2023
Lumighost Ltd
Middle .NET developer
Part time job. The main tasks are project support and development of new functionality (rest api).

Рисунок 0.1 - Резюме 170323971234.pdf

ABOUT ME

I studied at a regular school, where I participated in olympiads in mathematics, physics and computer science. Then I entered the university with a degree in Integral Sensory Systems, where i currently studying. I love learning new things and going deep into technology that's why I chose .NET because it's deep. My favorite youtube channels are: Teddy Smith, DevJubgles, extremeCode I completed course TeachMeSkills and looking for a cool job

ARSENY STEFANENKO

.NET DEVELOPER(JUNIOR)

+375 (44) 709-51-74

Belarus, Minsk

stefisen@yandex.ru

• HARD SKILLS

- C#, ASP.NET Core
- Entity Framework Core
- MS SQL
- SignalR
- HTML, CSS, SASS, JavaScript
- SOLID
- Git/GitHub
- Docker
- AutoMapper

• LANGUAGES

- English - B1(in progress)
- Russian - native

• EDUCATION

- Belarusian National Technical University (2020-2024)
  INTEGRATED SENSOR SYSTEMS ENGINEER
- TeachMeSkills (08.22-01.23)
  .NET DEVELOPER

WORK EXPERIANCE (Since June 2022)

• SOCIAL WEB GAME github

Teach Stack: **(JANUARY 2023)**

ASP.NET CORE / Entity Framework Core / MS SQL / JWT Bearer / SignalR / React / SCSS / GitHub

This is a web app with elements of a social network and a pokemon game

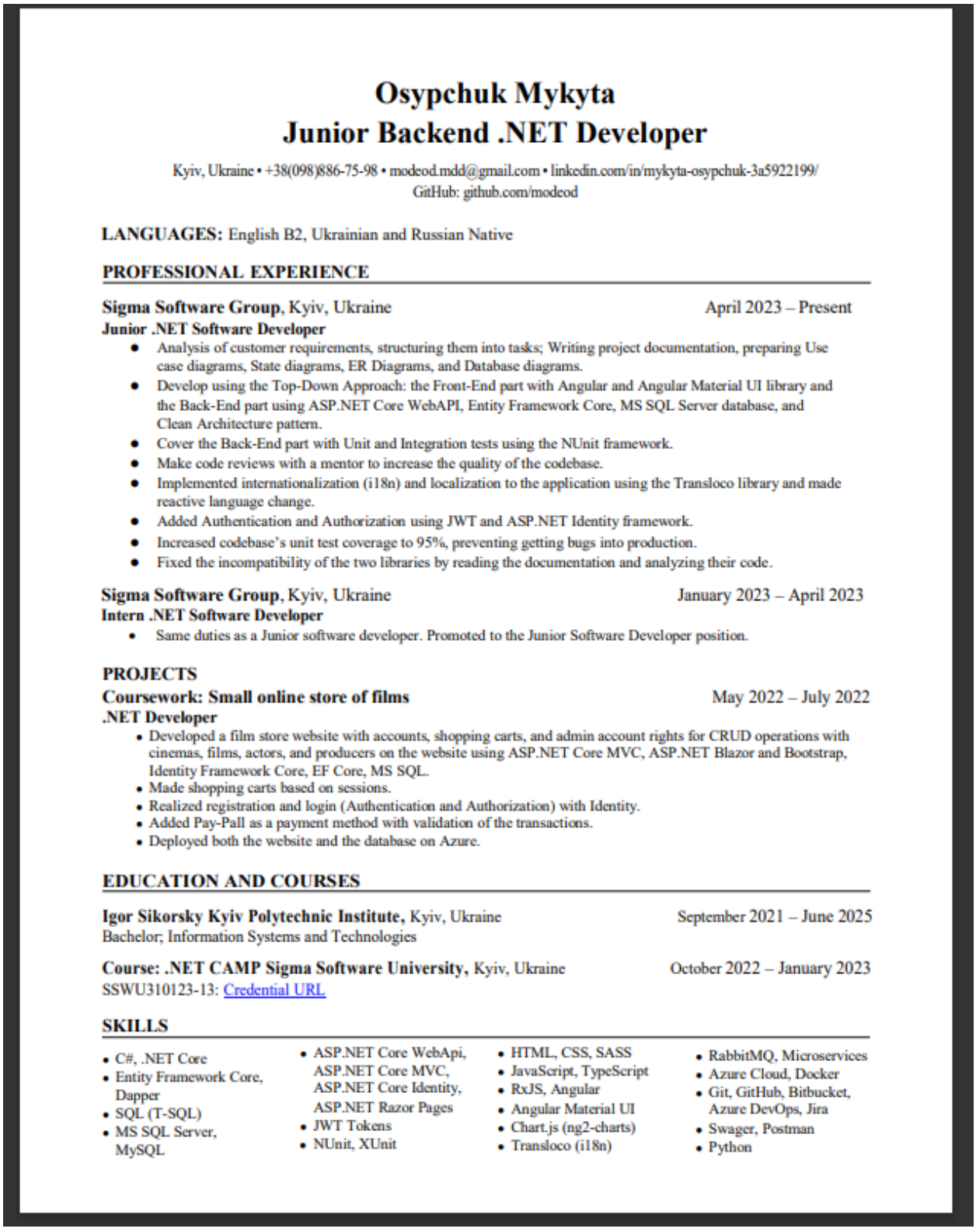
• SIMPLE BLOCKCHAIN github

Teach Stack: **(JUNE 2022)**

.NET CORE / LINQ / Serialization WinForms / GitHub

This is a computer application that found the hash of a block and gave the user who owns it.

Рисунок 0.2 - Резюме 1677415676205.pdf



Osypchuk Mykyta
Junior Backend .NET Developer

Kyiv, Ukraine • +38(098)886-75-98 • modeod.mdd@gmail.com • linkedin.com/in/mykyta-osypchuk-3a5922199/
GitHub: github.com/modeod

LANGUAGES: English B2, Ukrainian and Russian Native

PROFESSIONAL EXPERIENCE

Sigma Software Group, Kyiv, Ukraine — April 2023 – Present
Junior .NET Software Developer

- Analysis of customer requirements, structuring them into tasks; Writing project documentation, preparing Use case diagrams, State diagrams, ER Diagrams, and Database diagrams.
- Develop using the Top-Down Approach: the Front-End part with Angular and Angular Material UI library and the Back-End part using ASP.NET Core WebAPI, Entity Framework Core, MS SQL Server database, and Clean Architecture pattern.
- Cover the Back-End part with Unit and Integration tests using the NUnit framework.
- Make code reviews with a mentor to increase the quality of the codebase.
- Implemented internationalization (i18n) and localization to the application using the Transloco library and made reactive language change.
- Added Authentication and Authorization using JWT and ASP.NET Identity framework.
- Increased codebase's unit test coverage to 95%, preventing getting bugs into production.
- Fixed the incompatibility of the two libraries by reading the documentation and analyzing their code.

Sigma Software Group, Kyiv, Ukraine — January 2023 – April 2023
Intern .NET Software Developer

- Same duties as a Junior software developer. Promoted to the Junior Software Developer position.

PROJECTS

Coursework: Small online store of films — May 2022 – July 2022
.NET Developer

- Developed a film store website with accounts, shopping carts, and admin account rights for CRUD operations with cinemas, films, actors, and producers on the website using ASP.NET Core MVC, ASP.NET Blazor and Bootstrap, Identity Framework Core, EF Core, MS SQL.
- Made shopping carts based on sessions.
- Realized registration and login (Authentication and Authorization) with Identity.
- Added Pay-Pall as a payment method with validation of the transactions.
- Deployed both the website and the database on Azure.

EDUCATION AND COURSES

Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, Ukraine — September 2021 – June 2025
Bachelor; Information Systems and Technologies

Course: .NET CAMP Sigma Software University, Kyiv, Ukraine — October 2022 – January 2023
SSWU310123-13: Credential URL

SKILLS

- C#, .NET Core
- Entity Framework Core, Dapper
- SQL (T-SQL)
- MS SQL Server, MySQL
- ASP.NET Core WebApi, ASP.NET Core MVC, ASP.NET Core Identity, ASP.NET Razor Pages
- JWT Tokens
- NUnit, XUnit
- HTML, CSS, SASS
- JavaScript, TypeScript
- RxJS, Angular
- Angular Material UI
- Chart.js (ng2-charts)
- Transloco (i18n)
- RabbitMQ, Microservices
- Azure Cloud, Docker
- Git, GitHub, Bitbucket, Azure DevOps, Jira
- Swager, Postman
- Python

Рисунок 0.3 - Резюме 1683889430627.pdf

Stefanenko Arseny

Junior C#/.NET Developer

+375 (44) 7095174
stefisen@yandex.ru
LinkedIn: https://www.linkedin.com/in/arseny-stefanenko-376b531b5/m/
GitHub: https://github.com/Winststreloc

Work experience

March 2023 — May 2023
3 months

Orange Process

Minsk

Trainee Full-stack developer(JS + .Net)

Development of integrated reports in the CRM platform from Terrasoft. With glass language: MSSQL, C# (ASP.NET , .NET Framework), JavaScript (Ext.JS), CSS, jQuery

August 2022 - present 10 months

Pet-projects

Developer

During my self-study, I have developed several projects of my own. • Social web game about Pokemon.
Technology stack: ASP.NET , SignalR, JWT Beurer, Entity Framework Core, MS SQL, React
Web API for working in the cloud
Technology stack: ASP.NET , Docker, Heroku, Heroku Postgresql, JWT Beurer, Entity Framework Core Docker, Heroku

Education

Unfinished higher education

2024

Belarusian National Technical University, Minsk

2022

MBF, Integrated sensor systems,
TeachMeSkills

C# developer

Skills

Languages

Russian — native
English — B1

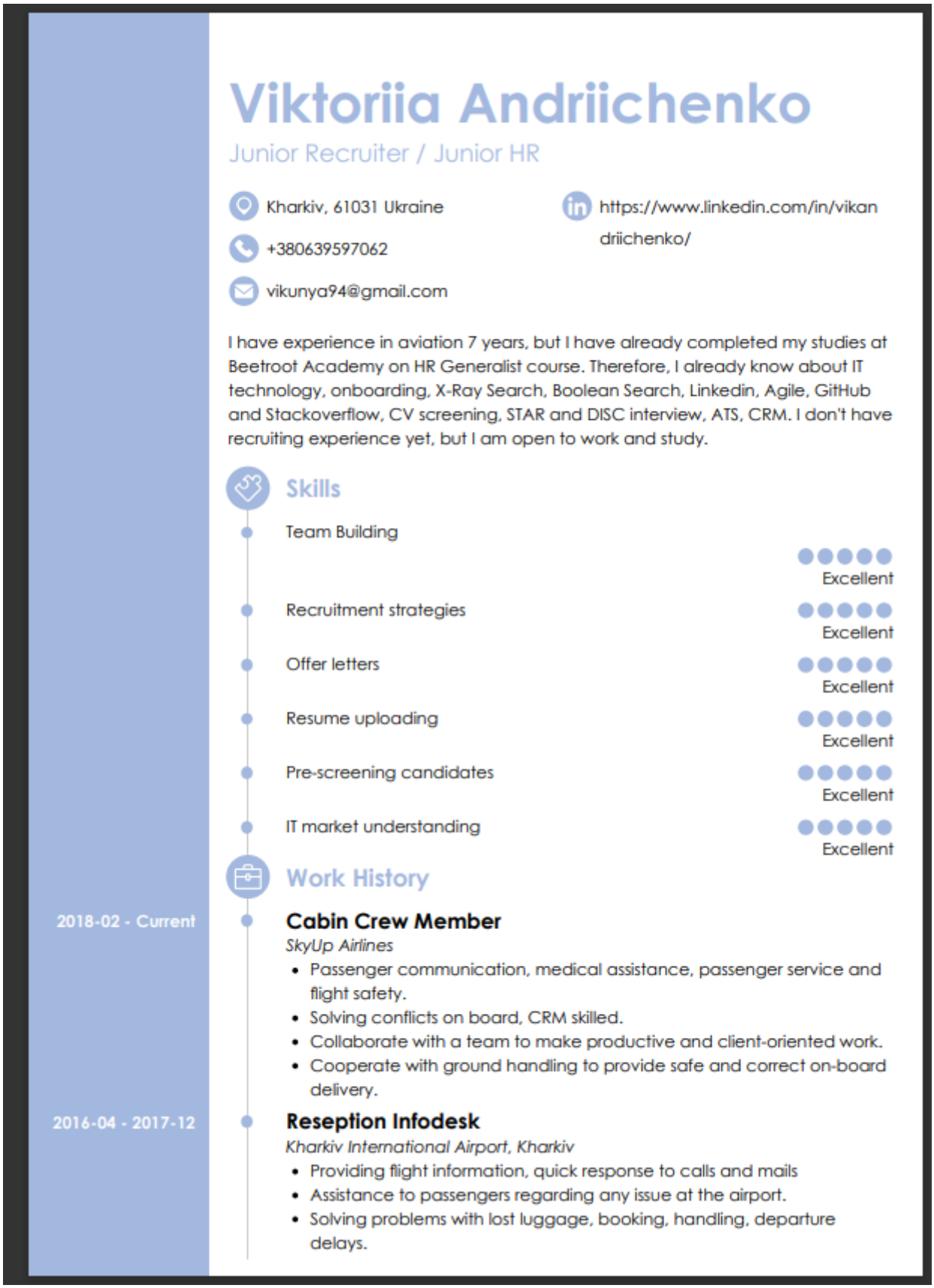
Hard Skills

ASP.NET, C#, MS SQL, Git, AutoMapper, SignalR, SOLID, JavaScript, HTML, Entity Framework, ASP.NET Core, Docker, REST API, PostgreSQL, ReactJS, CSS

About me

When I studied at school, I participated in olympiads at mathematics, physics and computer science. Then I entered the university with major in Integral Sensory Systems (where currently studying). I love learning new things and going deep into technology, that's why I chose .NET. My favorite youtube channels are: Teddy Smith, Devjubgles, extremeCode I completed course TeachMeSkills week.

Рисунок 0.4 - Резюме 1684165858680.pdf



Viktoriia Andriichenko

Junior Recruiter / Junior HR

Kharkiv, 61031 Ukraine
+380639597062
vikunya94@gmail.com
https://www.linkedin.com/in/vikandriichenko/

I have experience in aviation 7 years, but I have already completed my studies at Beetroot Academy on HR Generalist course. Therefore, I already know about IT technology, onboarding, X-Ray Search, Boolean Search, Linkedin, Agile, GitHub and Stackoverflow, CV screening, STAR and DISC interview, ATS, CRM. I don't have recruiting experience yet, but I am open to work and study.

Skills

- Team Building — Excellent
- Recruitment strategies — Excellent
- Offer letters — Excellent
- Resume uploading — Excellent
- Pre-screening candidates — Excellent
- IT market understanding — Excellent

Work History

2018-02 - Current

Cabin Crew Member

SkyUp Airlines

- Passenger communication, medical assistance, passenger service and flight safety.
- Solving conflicts on board, CRM skilled.
- Collaborate with a team to make productive and client-oriented work.
- Cooperate with ground handling to provide safe and correct on-board delivery.

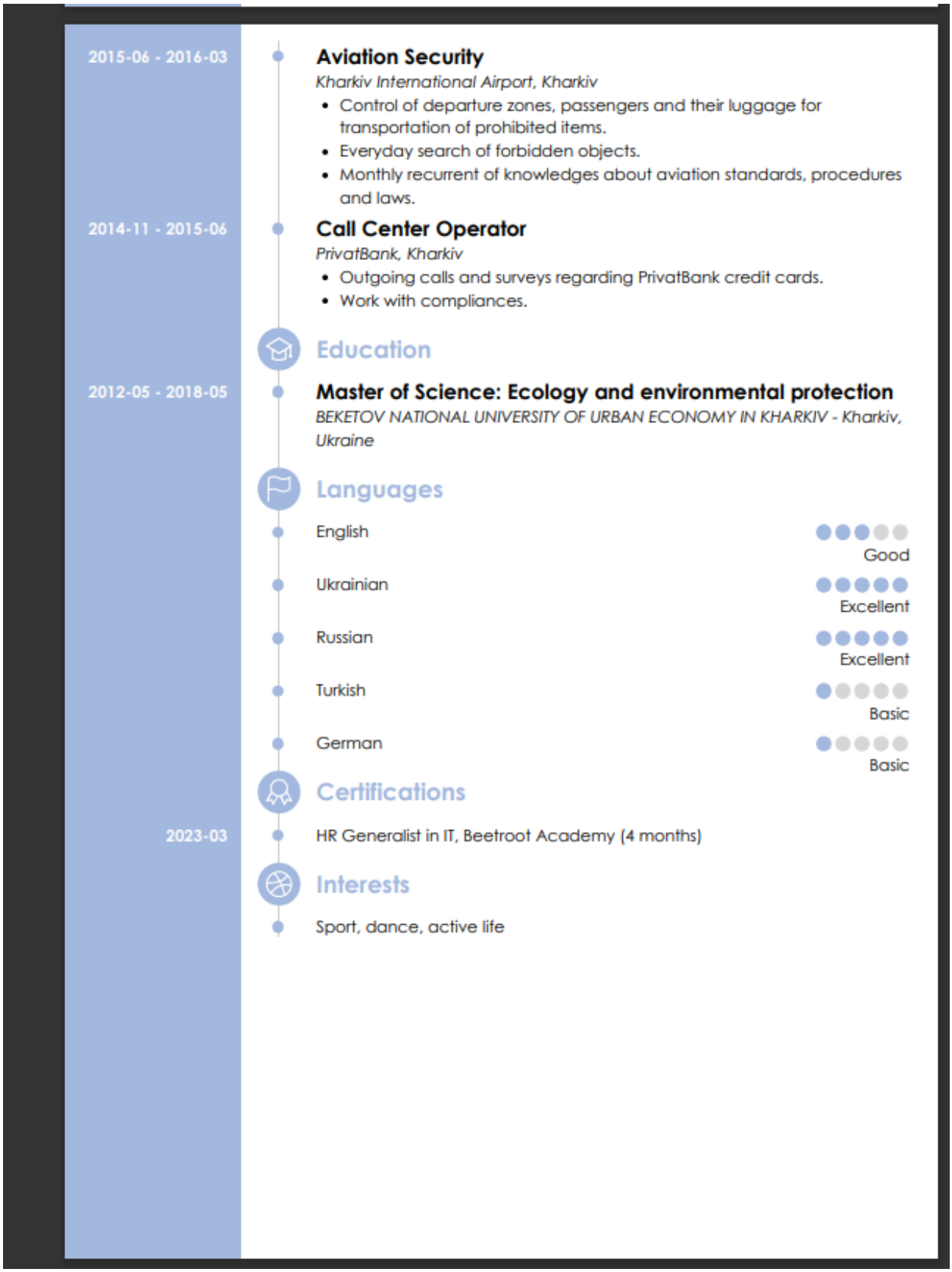
2016-04 - 2017-12

Reseption Infodesk

Kharkiv International Airport, Kharkiv

- Providing flight information, quick response to calls and mails
- Assistance to passengers regarding any issue at the airport.
- Solving problems with lost luggage, booking, handling, departure delays.

Рисунок 0.5 - Резюме 1689019534121.pdf - сторінка 1



2015-06 - 2016-03

Aviation Security

Kharkiv International Airport, Kharkiv

- Control of departure zones, passengers and their luggage for transportation of prohibited items.
- Everyday search of forbidden objects.
- Monthly recurrent of knowledges about aviation standards, procedures and laws.

2014-11 - 2015-06

Call Center Operator

PrivatBank, Kharkiv

- Outgoing calls and surveys regarding PrivatBank credit cards.
- Work with compliances.

Education

2012-05 - 2018-05

Master of Science: Ecology and environmental protection

BEKETOV NATIONAL UNIVERSITY OF URBAN ECONOMY IN KHARKIV - Kharkiv, Ukraine

Languages

English — Good

Ukrainian — Excellent

Russian — Excellent

Turkish — Basic

German — Basic

Certifications

2023-03

HR Generalist in IT, Beetroot Academy (4 months)

Interests

Sport, dance, active life

Рисунок 0.6 - Резюме 1689019534121.pdf - сторінка 2