Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка ATS інтегрованої з розширенням Google Chrome для оптимізації рекрутингу в LinkedIn

Виконав: студент IV курсу, групи СПзс-41 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Чикунов О.А.
(підпис) (прізвище та ініціали)

Керівник _____ Пастух О.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Приймак М.В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

АННОТАЦІЯ

Кваліфікаційна робота для здобуття ступеня «бакалавр» на тему: «Розробка ATS інтегрованої з розширенням Google Chrome для оптимізації рекрутингу в LinkedIn». Тернопільського національного технічного університету імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПзс-41.

Відомість про обсяг: 83 сторінок, 20 рисунків, 7 таблиць, 2 додатки, 14 джерел.

Об’єктом дослідження даної роботи є діяльність рекрутингових агенцій через їх взаємодію з LinkedIn для виконання роботи по підбору висококваліфікованого персоналу.

Метою роботи являється створення автоматизиваної системи керування кандидатами, для можливості додавати кандидатів з сайту LinkedIn і зберігати їх резюме в прив’язці до їх LinkedIn профіля, і відстежувати на які вакансії кандидат вже був поданий будь-яким з членів команди рекрутерів що працюють над закриттям вакансії.

Для дослідження була використана методика, що включає в себе аналіз предметної області та опрацювання інформації по поставленому завданню; проектування архітектури рішення, баз даних з використанням методів нормалізації; створення та тестування програми в середовищі Microsoft Visual Studio.

В результаті виконання поставленого завдання, було спроектовано та розроблено автоматизовану систему керування кандидатами інтегровану через розширення для Google Chrome на будь-яку сторінку LinkedIn.

Ключові слова: автоматизована система керування кандидатами, інтеграція з LinkedIn, розширення Google Chrome, рекрутингові процеси, інформаційна система, нормалізація бази даних, веб-архітектура, програмна інженерія, .NET технології, ReactJS, оптимізація процесів підбору персоналу.

ABSTRACT

Bachelor's qualification project on the topic: "Development of an ATS integrated with a Google Chrome extension for optimizing recruitment on LinkedIn." Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SPzS-41.

Summary of content: 83 pages, 20 figures, 7 tables, 2 appendices, 14 references.

The object of this research is the activity of recruitment agencies in their interaction with LinkedIn for the purpose of sourcing highly qualified personnel. The aim of the thesis is to develop an automated candidate tracking system that allows adding candidates directly from the LinkedIn website, saving their resumes linked to their LinkedIn profiles, and tracking which vacancies the candidate has already been submitted to by any member of the recruitment team working on the vacancy.

The research methodology includes analysis of the subject domain and study of the problem; system architecture and database design using normalization methods; as well as implementation and testing of the application in the Microsoft Visual Studio environment.

As a result of this work, an automated candidate tracking system was designed and developed, integrated via a Google Chrome extension into any LinkedIn profile page.

Keywords: automated candidate tracking system, LinkedIn integration, Google Chrome extension, recruitment processes, information system, database normalization, web architecture, software engineering, .NET technologies, ReactJS, recruitment process optimization.

ЗМІСТ

ВСТУП

В наш час, дедалі популярнішим стає пошук необхідної інформації на різноманітних веб-сайтах певного напрямку. З розвитком інформаційних технологій у мережі інтернет користувачі стали частіше використовувати послуги, які надаються за допомогою цієї мережі. А також на новий рівень вийшла можливість віддаленої роботи яку також можна знайти завдяки інтернету.

Серед таких спеціалізованих сайтів для пошуку роботу найбільшим є LinkedIn де було зареєстровано більше 1 млрд користувачів ще у 2023 році, а завдяки веріфікації профілів через мікрочіп з персональними даними, що вбудовані в паспорти – цей сайт має найбільш веріфіковану аудиторію на просторі інтернету. Але цей веб-сайт використовують не лише просто кандидати, для пошуку роботу, а і рекрутери і рекрутингові агенції які заробляють свої комісійні за пошук кандидатів в різноманітні клієнтські компанії.

На 2025 рік об’єм ринку прямих рекрутингових послуг у США складає 35,8 млрд$, без урахування ринку тимчасових співробітників, і показує зростання на протязі останніх п’яти років, а основний канал на якому ці агенції працюють і є саме Linkedin, який лише на підписках Talent Solutions для рекрутерів заробив 7 млрд$ в 2023, і після того показував зростання в 10% кожного року. Що свідчить про те – що інструменти для рекрутерів і рекрутерів користується попитом і цей попит зростає.

Проте є в Linkedin є обмеження, які не дозволяють працювати групі рекрутерів над однією вакансією таким чином, щоб бачити кому вже написали інші рекрутери з твоєї команди, а кому ні.

Це веде до репутаційних втрат якщо в компанії більше одної людини займається одною вакансією, але пришвидшити процес рекрутингу на даний проміжок часу можливо лише підключивши більшу кількість рекрутерів одночасно в роботу над вакансією, саме цю проблему і вирішує розробка надбудови над LinkedIn яка дозволить вирішити цю проблему роботи зі спільними списками кому

можна писати, а кому не можна. І фактично з використанням розробленого рішення можливо підключити хоч 100 хоч 1000 рекрутерів на одну вакансію одночасно і таким чином зменшити час на найм кандидатів в 100 або в 1000 раз в залежності від кількості долучених рекрутерів.

Наразі LinkedIn налічує 600000 активних користувачів-рекрутерів, які щодня пишуть сотні повідомлень на цій платформі кандидатам. І наше рішення розраховано саме на цю аудиторію. Кожний рекрутер має свою базу вакансій, свої шаблонні повідомлення в Excel, які має відправляти в випадковому порядку різним кандидатам коли вони додаються до рекрутера в LinkedIn. А ще рекрутер робить щоденно безліч одноманітних завдань які в рамках курсового проекту було вирішено автоматизувати.

Тобто аудиторія для якої розроблялось рішення є платоспроможною, і заробляє саме завдяки вдалому і швидкому пошуку потрібних висококваліфікованих кандидатів, здебільшого програмістів.

Тому було вирішено зробити окремий веб-сайт з базами даних який за допомогою розширення для Google Chrome можна було би підключити користувачам, які користуються LinkedIn і не змінюючи їх досвід користувачів запропонувати додаткові можливості, які не надає основна платформа LinkedIn, з якою вони працюють.

Наразі, на момент написання дипломної роботи, на проекті https://stoplist.ai/ вже є 35 користувачів-компаній, серед яких Sigma Software (має понад 2000 співробітників) і велику команду рекрутерів та інші.

Проект є комерційним, і таким, що не може розголосити весь код який було написано для його реалізації.

1. АНАЛІТИЧНА ЧАСТИНА

1.1 Опис об’єкта автоматизації

Предметна область роботи є діяльність рекрутерів під час процесу підбору персоналу використовуючи платформу LinkedIn, яка дозволяє заробляти гроші знаходячи там кандидатів на потрібні вакансії клієнтів, сплачуючи лише за підписні сервіси LinkedIn платну підписку.

Такий тип бізнесу існує і є прибутковим по всьому світу. Тобто потенційними користувачами платформи, що була розроблена в рамках дипломного проекту – є всі рекрутери, які рекрутують через LinkedIn – і це 600000 активних користувачів які користуються щоденно ним, деяка кількість із цих користувачів сплачують за користування або 50$ або 130$ на місяць в залежності від обраного пакету, що приносить компанії LinkedIn більше ніж 7 млрд$ чистої виручки в рік [1].

Одне з найважливіших і водночас найскладніших завдань рекрутера полягає в тому, щоб підібрати кандидата ідеально підходящого за технічними навичками під вакансію, а потім перевірити чи відповідають soft-skills кандидата цінностям компанії. В нашому проекті оцінка soft-skills лишається на рекрутерах, але рутинна робота має бути автоматизованою.

Ідея розробки такого рішення прийшла через те, що я як автор диплому маю великий досвід в рекрутингу і підборі персоналу в IT компанії, а також в керуванні командами рекрутерів, які займаються підбором персоналу, тобто усіма проблемами з якими стикається рекрутер, який виконує свою роботу вручну, та його керівник, який має оцінювати роботу рекрутера – я стикався особисто.

На вивчення предметної області мною було витрачено 7 років.

Тому була поставлена задача – зробити систему керування кандидатами (надалі ATS – Applicant Tracking System) таким чином, щоб кінцевому користувачу хотілось користуватись тою системою для спрощення своїх рутинних завдань.

LinkedIn має структуру кругів знайомств по принципу кругів знайомств в реальному житті, і перед тим як додатись до кандидата в контакти рекрутер має

зробити запит до кожного потенціального кандидата, з яким вже є якісь спільні контакти, і другим повідомленням після додавання таким кандидатом в сітку контактів рекрутер може вже надсилати повний опис вакансії. Через те якщо рекрутер водночас працює з більше ніж трьома вакансіями всі рекрутери тримають короткий опис вакансії і детальний опис вакансії для кожної вакансії в окремому файлі (частіше в Excel або в Google Docs). І чим більше вакансій у рекрутера в роботі – тим більша вірогідність помилки при відправці не того опису вакансії не тому кандидату, бо кандидати добавляються до рекрутерів в випадковому порядку.

Вирішення проблеми швидкого додавання потрібного опису вакансії і автоматизація відправки першого короткого повідомлення в LinkedIn ставиться як однією з задач виконання даної роботи.

А також ставиться задача відстежування статистики ефективності роботи рекрутерів по вакансії, з можливістю переглянути кількість відправлених повідомлень рекрутером кандидатам в LinkedIn, кількість прийнятих повідомлень кандидатами, кількість кандидатів з ким було проведено будь-які етапи рекрутингового процесу і кількість кандидатів кому було видано пропозиції на роботу для оцінки ефективності рекрутингового процесу окремими рекрутерами в команді, або ефективності всієї команди рекрутингового відділеня.

Також для рекрутерів є зручним відстеження кандидатів в kanban-дошках, то вже стала практика, до якої звикли користувачі, тому було вирішено написати свою kanban-дошку для керуванням кандидатами, що знаходяться в рекрутинговому процесі на вакансіях. Але відмінність від аналогічних рішень – в шляху потрапляння кандидатів в такі kanban-дошки – і зменшення кількості кліків для кінцевого користувача, що має призвести до більшої ефективності рекрутингових команд.

Рекрутери під час своєї роботи постійно мають переходити в додатковий розділ у кандидатів який називається Skills (навички), і вручну, або за допомогою окремих сторонніх додатків перевіряти наявність ключових слів в навичках кандидата, щоб оцінити можливість того, що він підходить на конкретну вакансію. Наприклад, для таких вакансій як .Net backend engineer – скоріш за все ключовими

навичками будуть .Net, і додаткові навички по базам даних таких як SQL, PostgressQL ітд, тому було вирішено зробити автоматичну перевірку цих навичок в профілі кандидата, для першої версії розробки не планувалось підключати штучний інтелект, але із зростанням кількості платних користувачів на платформі планується покращити рішення через оцінку профілів кандидатів штучним інтелектом який насправді буде нейронною сіткою, що буде натренована на досвіді користувачів системи. Тому рішення базується на доменному імені .ai, але поки немає достатньої кількості користувачів було вирішено не ускладняти систему.

Було вирішено розробляти додаток по моделі SaaS (Software as a Service), з платною підпискою для користувачів, але основна цільова аудиторія рішення знаходиться в Сполучених Штатах Америки, тому основна мова розробленого рішення є англійською і було вирішено не робити підтримку додаткових мов на платформі. Основна планована цільова аудиторія – компанії які займаються IT-рекрутингом – тобто пошуком висококваліфікованих програмістів.

1.2. Огляд подібних проектних рішень

Наразі існує більше 16 відомих рішень які заробляють в середньому 3-5 млн$/рік і дозволяють рекрутерам вести базу своїх кандидатів на своїх веб-сайтах, також вони мають розширення для Google Chrome, але ні одне з цих рішень не дозволяє керувати знаходженням кандидата на вакансії напряму з розширення Google Chrome не виходячи з LinkedIn. Всі конкуруючі сайти працюють таким чином, що рекрутер додає кандидата в базу, а потім заходить і розподіляє кандидата на окрему вакансію в середині бази кандидатів, що призводить до того, що рекрутер за цілий робочий день в такому режимі стає розфокусованим і може припускатись більших помилок в процесі рекрутинга а також в процесі спілкування з кандидатами, що може шкодити іміджу роботодавця, який дає йому вакансії в роботу.

Всі рішення конкурентів змушують рекрутера робити від 4 до 8 додаткових кліків в різних вікнах, щоб додати одного кандидата в свою базу і на кожному профілі кандидата який рекрутер хоче внести в базу – переходити на веб-сайт рішення і там налагоджувати куди саме поданий кандидат, на які умови ітд.

Я захотів змінити цей досвід користувачів, і дозволити робити все в 3 кліки. (при тому, що 2 з цих кліків – то абсолютно прозорий вибір назви вакансії в самому розширенні і етапу вакансії). Тим самим ми дозволяємо сфокусуватись рекрутерам на рекрутингу, а не на веденні баз даних, які по суті виступають як «мусорне звалище» з застарілими даними кандидатів.

Проблема імплементації конкуруючих рішень складається в тому – що дуже важко заставити всіх рекрутерів вести бази даних через те, що вони розуміють що то їм не потрібно. Я особисто стикався з проблемою організації рекрутинга на базі ATS Cleverstaff в команді з 7 рекрутерів, і хоча було витрачено багато часу, деякі рекрутери вели базу даних погано і потім було важко розбиратись з деякими кандидатами.

Існуючи системи – не допомагають рекрутерам в їх рутинній роботі по написанню кандидатам в LinkedIn, і хоча всі заявляють що підвищують ефективність – то лише через те, що кандидати стають розкладені по етапах на вакансіях, і рекрутинговий процес стає більше прозорим для керівників рекрутингових департаментів, але для того, рекрутери витрачають втричі більше часу на кожного кандидата, при тому додатково оплачують сторонні рішення які можуть автоматизувати відправку повідомлень кандидатам.

Тому моє рішення – базоване на бажанні знизити час обробки кандидата рекрутером, і спростити ручну роботу на добавлення в базу даних і на написання кандидату першого і другого повідомлення в LinkedIn, і зробити систему керування кандидатами і помічник по відправці повідомлень в одному додатку, щоб рекрутери не переплачували за користування сторонніми сервісами.

Оскільки на ринку наразі не існує програм аналогічної функціональності – то можна умовно сказати що аналогічна функціональність у нас лишилась від стандартних Applicant Tracking Systems – таких як:

1) https://www.manatal.com/ - засновники з Індії (виручка 7+ млн$/рік)
2) https://cleverstaff.net/ - засновники з України (виручка 3-5 млн$/рік)
3) https://hurma.work/en/ - засновники з України (виручка 10+ млн$/рік)
4) https://peopleforce.io/ - засновники ніби теж з України *(отримали 2 млн$ на розробку рішення для рекрутингу, бо воно в них дійсно погане).
5) https://www.workable.com/ - засновник з східної Європи (виручка 10+ млн$/рік)
6) https://recruitee.com/ - найжахливіша система з України, ціною 10000 грн за одне робоче місце на рік (але 8+ кліків, щоб додати одного кандидата в потрібну вакансію і 4+ екрани, на цьому моменті мої нерви не витримали і система була оцінена – як система яка зроблена людьми які не розуміються на задачах і проблемах рекрутерів, по принципу розробка для розробки).
7) https://recruiterflow.com/ - засновники з Індії (виручка 3+ млн $/рік)
8) https://textpie.io/ - засновник з України (молодий стартап який дозволяє вставляти описи вакансій в чат, і коштує 6,99 або 9,99$ в місяць, але не дозволяє зручно обирати потрібне повідомлення якщо кількість таких повідомлень більше 10, тобто вже з 5 вакансіями в одночасній роботі з цим рішенням працювати стає незручно, стартап вийшов на ринок восени 2024, даних про виручку немає)

Всі конкуренти були ретельно протестовані рекрутерами в роботі над реальними вакансіями. Але суть систем лишається незмінною, то зазвичай веб-сайт з розширенням під гугл хром. На кожну було створено окрему папку де ретельно продивившись всі екрани хотілось знайти зернятко корисності і зручності. Лідерами виявились Cleverstaff, Manatal і Recruiterflow по якості того, що вони роблять. Але кожна система мала свої мінуси.

Великим мінусом Recruiterflow - є те що система працює по індійському рекрутингу – і дозволяє автоматично написати всім підряд кандидатам без огляду на навички, досвід ітд, всім кого видало в пошуковому запити – всім автоматично надсилає повідомлення, що неодмінно призведе до блокування профілю з урахуванням того як працюють алгоритми LinkedIn.

Великим мінусом всіх – є кількість кліків для взаємодії з кандидатом і закріпленням його до визначеної вакансії, яка наразі перевищує 4-8 кліків і заставляє переходити на інші вікна, що призводить до розфокусування рекрутера з роботи, що він робить.

Багато з рішень, що вказані вище, дають доступ до користування своїми сервісами лише після спілкування телефоном чи гугл-мітингом. Ми хочемо зробити більш лояльну систему до користувачів, але обмеження системи в рамках платної підписки не планувалось розробляти в рамках цієї роботи.

Веб-сайти за своєю функціональністю подібні, бо всі вони оцифровують процес підбору кандидата на вакансію. Для огляду кожної системи я створив окремий простір де розмістив порівняльні скріншоти кожного з додатків, щоб можливо було повернутись в будь-який момент і переглянути, бо можливість користування системами частіш за все обмежена двома тижнями безкоштовного користування, або демо-дзвінком з представником компанії-розробника.

Нижче приклад як виглядає розширення для Google Chrome від Cleverstaff, це рішення дозволяє організувати рекрутинг лише в середині однієї компанії. І всі дані стають недоступні коли перестаєш платити платну підписку. На Рис.1.2.1. видно, що є можливість додати кандидата в базу або на окрему вакансію, але потім рекрутер має обов’язково зайти на веб-сайт і вказати на який саме етап вакансії було додано конкретного кандидата.

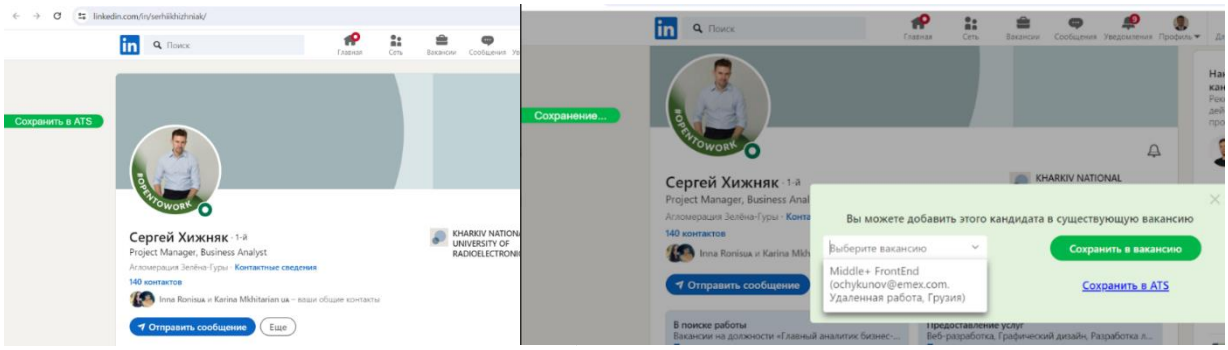

Рис. 1.2.1 – Приклад роботи веб-розширення Cleverstaff для LinkedIn, з лівої сторони пасивний плагін, з правої - активований. (5-6 кліків для додавання кандидата в вакансію і етап воронки вакансії)

Після цього треба перейти на сайт ATS Cleverstaff, і додати більше інформації про кандидата. Всі рішення платні, і наразі ними я вже не користуюсь. Тому є лише якість старі скріншоти на час користування платними підписками за 35$/місяць.

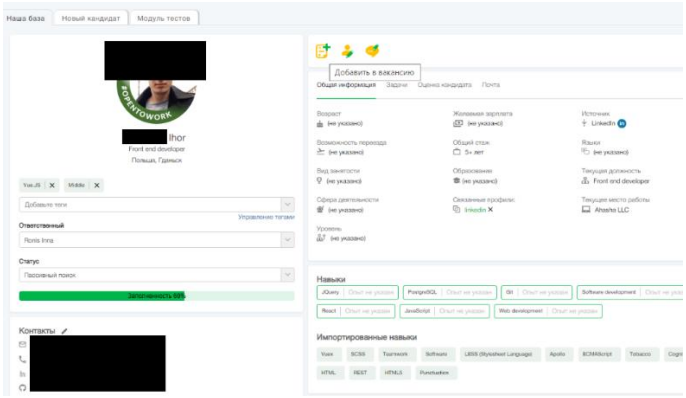

Рис. 1.2.2 – Вигляд профіля кандидата в ATS Cleverstaff.

На Рис.1.2.2 показано загальний вигляд профіля який далі потрібно додавати на вакансію, або як мінімум вказувати на який етап вакансії його треба додати, тобто сайт ініціює штучне накручування своєї відвідуваності через змушення користувачів заходити на їх веб-сайт для того щоб зробити якісь додаткові дії. І це незручне для користувачів рішення заробляє від 3 до 5 млн$ на рік.

На Рис.1.2.3 показаний принцип роботи іншого розширення – яке треба запускати зверху з панелі розширень, що вже не зручно.

Яке я б охарактеризував як приклад того як роботи не треба робити додатки. Я нарахував 8 кліків не враховуючи кліка на сторінку розширення, щоб лише додати кандидата в свою базу даних.

І це рішення, зображене на Рис. 1.2.3., що потребує 8 кліків, після кожного кліка йде додаткове завантаження, щоб додати кандидата в базу кандидата на потрібну вакансію коштує всього 10000 грн/рік за кожного користувача, що ним користується в команді рекрутерів.

Я не витримав перегляду презентацію, не те що працювати з таким рішенням. Але це рішення має якогось свого користувача мабуть, хто не контролює час рекрутерів на роботу, або платить зарплатню на рівні прожиткового мінімуму. Я би таким рішенням не користувався навіть якби мені за те доплатили.

Основна ціль будь-якого рішення – має бути спрощення життя користувача, а не ускладнити його додатковою роботою.

Інші розглянуті схожі рішення також мають свої недоліки. Деякі інтегровані лише з платними підписками Recruiter Lite (170$/міс) або Recruiter (5000$/рік) згідно тарифної сітки LinkedIn і регіональним цінам для України.

Жодне з рішень не дозволяє технічно будувати рекрутинг між партнерами, або без значних додаткових витрат мати можливість підключати фрілансерів для роботи над закриттям вакансії, поєднуючи бази даних про кандидатів між партнерами, що працюють над однією вакансію, при наданні обмеженого доступу з баз даних партнерів (без персональної та іншої чутливої інформації). І навіть LinkedIn не дозволяє так працювати на пакеті за 5000$/рік.

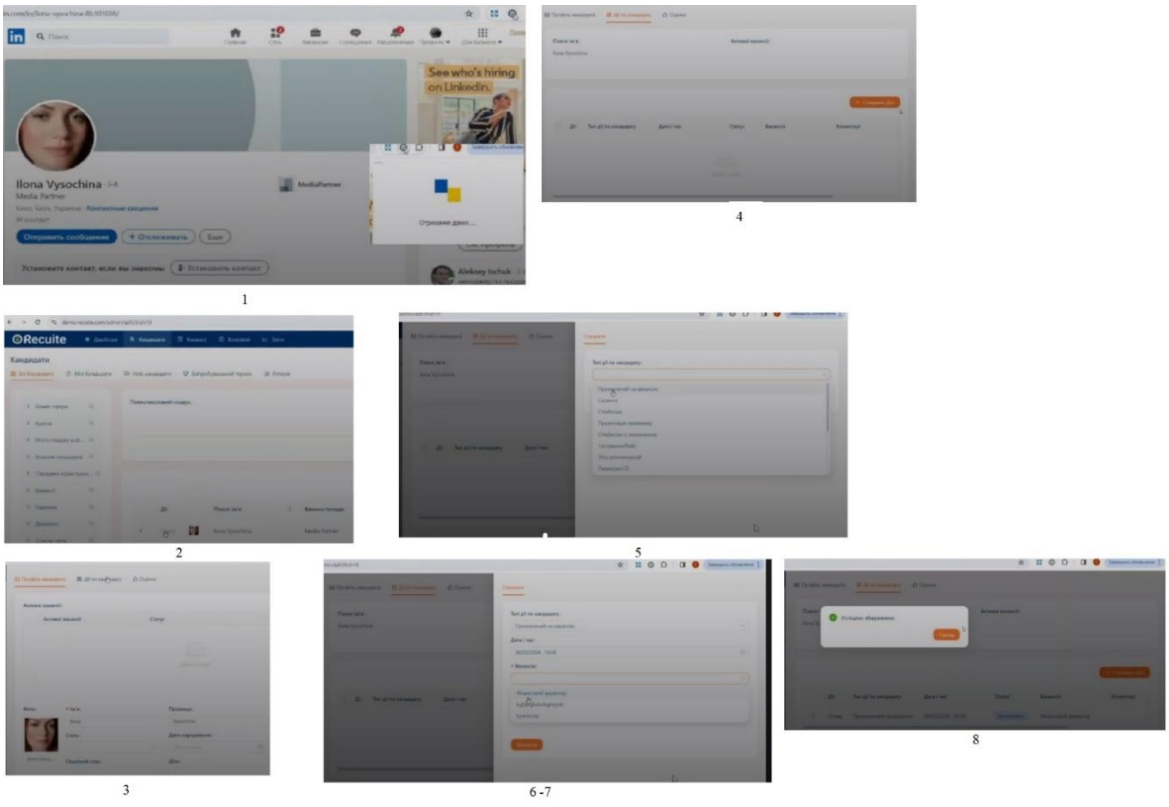

Рис. 1.2.3. Приклад роботи розширення від Recruitee, 8 кліків щоб додати кандидата із LinkedIn на вакансію в ATS.

2. ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдання

Головним завданням дипломної роботи являється створення для системи управління кандидатами, та рекрутерами, яке спростить процес рекрутинга при використанні LinkedIn, а також автоматизує відправку першого і другого повідомлення кандидатам через сайт LinkedIn, не виходячи з нього на сторонні ресурси.

Для цього основний код був написаний на .Net, побудована база даних на PostgressQL, деякі частини коду для швидкості відпрацювання були написані на Node.JS (то було архітектурне рішення).

Функціонал проектного рішення було описано задачами в Jira, і хоча цілі і задачі дипломної роботи вже виконані, проект ще знаходиться в стадії доробки до основної ідеї, над ATS наразі будується більша платформа.

На момент написання диплому було створено 7 епік задач, в яких було 491 окремих підзавдань на виконання по бекенду і фронтенду, з яких лишилось 237 до закриття, які не стосуються цілей і задач цієї дипломної роботи, а також проведена оцінка роботи [3].

Було сформовано на початку розробки окремий документ на 21 сторінку який описував мінімальну функціональність платформи від створення користувачем аккаунта на платформі до моменту процесу роботи системи.

Нижче наведено окремі задачі, що ставились як задачі для цієї роботи. Веб-сайт який працює у зв’язці з Google Chrome Extension – надалі Stoplist.ai – буде надаватиме користувачам можливість:

- мати кнопку на лендінгу Stoplist.ai – для реєстрації користувача на платформі, і збирання його персональних даних (email, телефон, фамілія, ім’я, назва компанії)

- додавати співробітників своєї команди в окремий простір який в робочому варіанті називається Агенція.

- створити форму реєстрації рекрутера через запрошення на емейл, адміністратора, чи рекрутера+адміністратора по посиланню-запрошенню, що може створювати перший зареєстрований користувач, і додавати всіх цих користувачів з їх ролями в спільне середовище

- користувач має можливість створити вакансію на платформі і додати на неї відповідальних за закриття рекрутерів або рекрутерів+адміністраторів.

- переглядати інформацію про те на якому етапі воронки закриття вакансії знаходиться кандидат

- відображати дані про те на якому етапі на кожній вакансії знаходиться яка кількість кандидатів

- загрузити розширення для Google Chrome з веб-інтерфейсу сайта Stoplist.ai

- додавати користуючись розширенням для Google Chrome кандидатів в обрану вакансію на потрібний етап вакансії, не виходячи з LinkedIn в 3 кліки

- додати шаблон повідомлень для швидкої відправки кандидатам в полі додавання кандидатів на пакеті LinkedIn Business (50 $/міс) і вище.

- додати користувачу можливість швидко вставляти в діалоги потрібні повні та детального описи вакансій в залежності від обраної в розширенні вакансії

- створити додаткове відображення на сторінці повідомлень в LinkedIn для зручності зміни статусів по кандидатам не виходячи з листування з кандидатом.

- здійснити обфускацію коду розширення для Google Chrome для запобігання несанкціонованим копіюванням програмного коду продукту (бо така спроба вже була), а також з метою уникання блокування LinkedIn розробленого розширення.

З метою обходу блокування зі сторони LinkedIn код розширення для Google Chrome не може буде викладений в додатки Google Chrome і має встановлюватись користувачами лише в ручному режимі.

2.2 Проектування архітектури

Архітектура розробленої ATS-системи (ATS — Applicant Tracking System) ґрунтується на мікросервісному підході й розділена на два незалежні предметні домени.

Перший, рекрутинговий домен (Recruitment Context), охоплює всі процеси, пов’язані з вакансіями та кандидатами: створення позицій, внесення змін до етапів підбору, зберігання й оновлення профілів, а також логіку зв’язування «кандидат — вакансія». За ці функції відповідають окремі служби-контейнери — `vacancy-service, candidate-service` і `application-service`; їхню асинхронну діяльність доповнюють фонова компонента `recruiting-worker`, що обробляє події, та `vacancy-worker`, яка виконує періодичні завдання, наприклад архівацію закритих позицій.

Другий домен Actors Context (зосереджений на керуванні акторами: їхніми обліковими записами, ролями, агентствами та компаніями-замовниками. Тут працюють `user-service, company-service i client-service`, а також додатковий `actors-worker`, що реагує на події реєстрації чи зміни прав доступу. Кожен сервіс розгорнуто у власному контейнері .NET 7 і має власну схему в PostgreSQL, тому оновлення однієї частини системи не впливає на роботу інших. Зв’язки між сервісами мінімізовано: замість прямих REST-запитів вони обмінюються повідомленнями через Kafka, що гарантує доставку подій, дозволяє обробляти їх повторно та спрощує горизонтальне масштабування.

На клієнтському боці передбачено три точки взаємодії. Рекрутери працюють у односторінковому веб-порталі на React, де в канбан-інтерфейсі відображаються всі етапи підбору. Адміністратори заходять у окремий React-інтерфейс, з якого створюють агентства, додають користувачів, налаштовують ролі, переглядають журнали змін і системні параметри. Третім клієнтським компонентом є розширення Chrome: відкривши профіль кандидата в LinkedIn, рекрутер натискає кнопку, і

розширення зчитує вміст сторінки, формує подію candidate.created, після чого опублікований запис миттєво потрапляє у топік Kafka «recruitment».

Усі клієнтські запити прямують не безпосередньо до мікросервісів, а через єдиний Gateway, реалізований як Backend-for-Frontend на Node.js. Шлюз перевіряє підпис токена, виданого IdentityServer-ом, накладає правила RBAC, обмежує частоту звернень і, якщо потрібно, паралельно викликає кілька бекенд-служб, об'єднуючи їхні відповіді у зручний для фронтенда формат. Така неблокуюча модель вводу-виводу дає можливість обробляти тисячі одночасних запитів із мінімальною затримкою та, завдяки WebSocket-каналу, одразу повідомляти користувача про зміну статусу кандидата чи вакансії.

Безпеку системи забезпечує зв'язка OAuth 2.0 з розширенням PKCE. Під час автентифікації користувач одержує короткоживучий код, який обмінюється на JWT. Токен містить `claims` із ідентифікатором агентства, набором ролей і `tenant`-ідентифікатором; таким чином, контролер чи шлюз можуть одразу визначити, до чого має доступ запитувач. Відмова від довготривалих рефреш-токенів і вимога постійної перевірки підпису значно зменшують ризик компрометації.

У реляційній моделі даних центральне місце посідають п'ять сутностей. Таблиця Agency зберігає рекрутингові агентства й пов'язана з Users відношенням «один-до-багатьох»: усі співробітники агентства мають спільний `tenant`-ідентифікатор. Кожен користувач водночас може належати до кількох ролей (адмін, рекрутер, клієнт) через проміжну many-to-many-таблицю. Вакансії, відкриті компаніями-замовниками, містяться окремо; списки претендентів розташовані у Candidates. Для фіксації взаємодії «кандидат — вакансія» введено перехресну таблицю Assignments, де зберігається не лише ідентифікатор пари, а й оперативні дані — поточний етап, дати інтерв'ю, комісія рекрутера та його нотатки. Такий підхід дозволяє одному кандидатові брати участь у кількох процесах одночасно, а одній вакансії утримувати безліч кандидатів, при цьому швидко діставати потрібний зріз даних. Для великих агентств система підтримує мультитенантний режим «schema-per-tenant», коли в одній інсталяції PostgreSQL кожен великий клієнт отримує власну схему, ізольовану від інших.

Робочий цикл виглядає так. Коли розширення надсилає відомості про нового кандидата, `candidate-service` створює відповідний запис і відразу сповіщає `application-service`, який зв'язує кандидата з потрібною вакансією. У керованій подією моделі `recruiting-worker` підписується на топік Kafka й виконує супровідні дії — додає запис у журнал, надсилає листа команді тощо. Якщо рекрутер переміщує картку на наступний етап у канбані, `vacancy-service` фіксує новий статус, генерує подію `assignment.stage_changed`, і WebSocket-потік у браузері оновлює інтерфейс для всіх користувачів, що працюють над тією самою вакансією. Завдяки подієвій узгодженості сервісам не потрібні ланцюжки синхронних REST-викликів, а втрата одного компонента не блокує роботу інших.

Уся інфраструктура описана декларативно в Helm-чартах і розгортається у Kubernetes-кластері. GitHub Actions збирає Docker-образи, запускає тестові збірки, після чого Helm-реліз автоматично оновлює сервіс у GKE. Збирання логів виконує Fluent Bit, який пересилає записи в Loki, а метрики в Prometheus; Grafana показує дашборди із завантаженням мікросервісів, затримками та швидкістю обробки подій Kafka. Завдяки впровадженій OpenTelemetry кожен HTTP-запит і кожна Kafka-подія мають глобальний Trace-ID, що спрощує діагностику.

Поєднання мікросервісів, подієвої шини, швидкісного Gateway на Node.js і суворої RBAC-моделі дозволяє системі одночасно бути масштабованою, продуктивною та безпечною. Розділення предметних областей спрощує командну розробку, а підтримка мультитенантності робить платформу готовою до залучення великих агентств без ризику змішування їхніх даних. У підсумку отримано цілісне рішення, здатне автоматизувати весь рекрутинговий цикл — від імпорту кандидатів до фінального етапу рекрутингу — з мінімальною затримкою та максимальним контролем доступу.

Схему архітектури наведено на Рис. 2.2.1 – Архітектура ATS для платформи з рекрутингу.

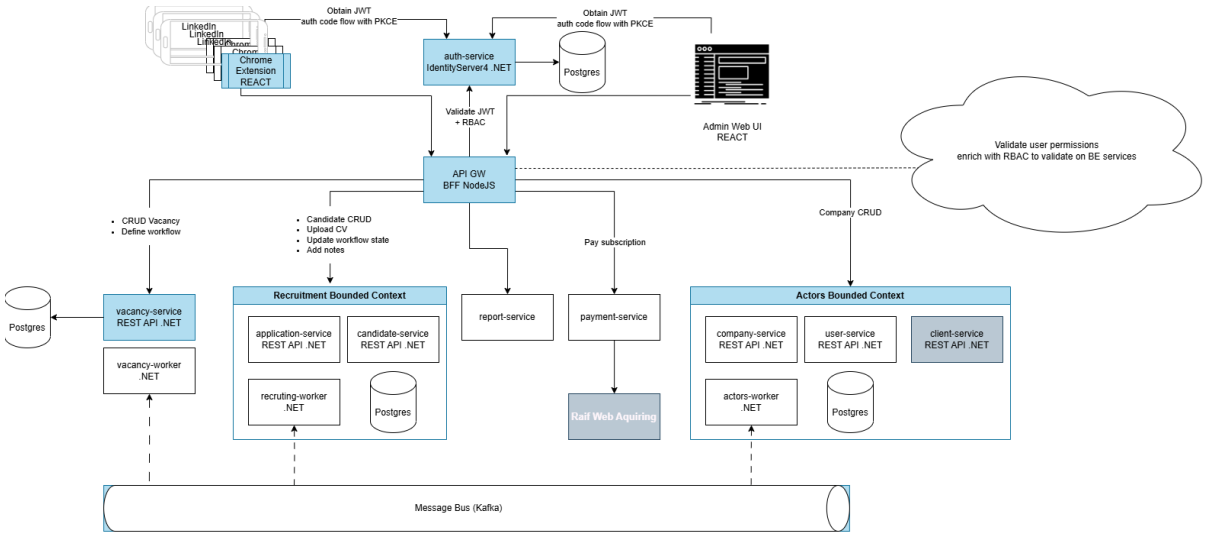

Рис. 2.2.1 – Архітектура ATS платформи для рекрутингу.

В рамках даної роботи поки не було реалізовано payment-service – і всі оплати здійснюються в ручному режимі, а також створення Admin Web UI було винесено за цілі цієї роботи, і взаємодія з видалення користувачів, отримання списку користувачів, блокування користувачів (така можливість була знайдена в процесі тестування через усунення окремих багів) наразі робиться через запити Swagger для токена з спеціальними правами рівня SuperAdmin.

2.3 Проектування та моделювання системи

З метою забезпечення масштабованості та зручності подальшого супроводу система побудована за мікросервісною архітектурою. Нижче описано головні компоненти, що формують логічне ядро розробки, а також їхню роль у межах архітектури:

1. Основна база даних – яка зберігає всю необхідну інформацію про кандидатів, рекрутерів, вакансії, а також взаємодію між ними. База даних складається з кількох таблиць, таких як Candidate, Education, Employment,

Candidate_Skill, Agency_Candidate тощо. Ці таблиці пов'язані через зовнішні ключі, що забезпечує цілісність та узгодженість даних.

2. Інтерфейс користувача (UI) – для забезпечення зручності роботи користувача, інтерфейс побудований на основі ReactJS. Це дозволяє створити інтерактивні та динамічні веб-сторінки, де рекрутери можуть додавати кандидатів, переглядати їхні дані, а також взаємодіяти з різними елементами системи, такими як Вакансії, Резюме, Профілі кандидатів ітд.
3. Розширення для Google Chrome – є однією з ключових частин архітектури, яке дозволяє рекрутерам взаємодіяти з профілями кандидатів без необхідності переходити на веб-сайт ATS, і через те економить час користувачам. Розширення сканує сторінки LinkedIn, перевіряючи наявність певних ключових слів, і дозволяє швидко додавати кандидатів до бази даних.
4. Обробка даних – включає взаємодію між клієнтським інтерфейсом і сервером, де використовуються Node.js для обробки запитів від користувачів, а також .Net для реалізації логіки роботи сервера. База даних обробляє запити на додавання, оновлення і вилучення даних.
5. Безпека та захист даних – є одним з важливих аспектів є забезпечення безпеки інформації. Для цього використовується обфускація коду розширення для Google Chrome, щоб запобігти можливості копіювання ідеї та несанкціонованому доступу до даних.
6. Інтеграція з зовнішніми платформами – є можливість інтеграції з такими зовнішніми платформами, як LinkedIn, через API для збору публічних профілів кандидатів, що дозволяє автоматизувати процес підбору кандидатів, а також з HR-системами користувачів.

Щоб краще проілюструвати взаємодію ролей (Кандидат, Рекрутер, Адмін, Суперадмін) із переліченими компонентами, далі подано діаграму варіантів використання. Вона відображає, які саме дії може виконувати кожен актор. Саме з цієї точки зору стає зрозуміло, як описані вище складові поєднуються у єдиному робочому процесі (Рис.2.3.1 – Діаграма використання).

Діаграма випадків використання (див. рис. 2.3.1) демонструє, як чотири основні ролі — Рекрутер, Адміністратор, Суперадміністратор і допоміжний сервіс - Chrome-розширення — взаємодіють із підсистемами ATS. Роль *Кандидат* поки що пасивна: самостійної реєстрації чи подання резюме з автоматичною реєстрацію такої ролі передбачено, але в подальшому, поки ця роль пасивна, тому стрілки до неї відсутні. Але сутності кандидатів створюються на платформі рекрутерами.

Усі користувачі починають із сценарію «Login / Authenticate»: система здійснює автентифікацію через OAuth 2.0 із PKCE та повертає короткоживучий JWT-токен, з фіксованим строком терміну дії такого токену і автоматичним оновленням його. Після цього відкривається доступ до дій, дозволених конкретній ролі.

Для Адміністратора центральним є варіант «Create Vacancy»: саме він формує нову позицію, заповнює опис, вимоги й бюджет, а система створює відповідний запис і розповсюджує подію `vacancy.created`. Крім того, адміністратор має доступ до сценаріїв «Manage Users & Roles» (додає або блокує рекрутерів, призначає ролі) та «Generate Reports» (отримує зведення про кількість доданих кандидатів по вакансії окремим рекрутером що важливо для розрахунку «time-to-hire» та інших метрик рекрутингового процесу, а також дозволяє оцінити активність і ефективність роботи рекрутера).

Суперадміністратор має розширені повноваження: він може видаляти будь-які облікові записи, блокувати користувачів (через видалення емейлу із БД користувачів, і залишенні емейлу в Gateway – відбувається неможливість отримати новий JWT-токен, що працює як блокування доступу до платформи), і запускати генерацію запити в базу даних про кількість зареєстрованих користувачів і отримувати їх персональні дані, що були вказані користувачами при реєстрації.

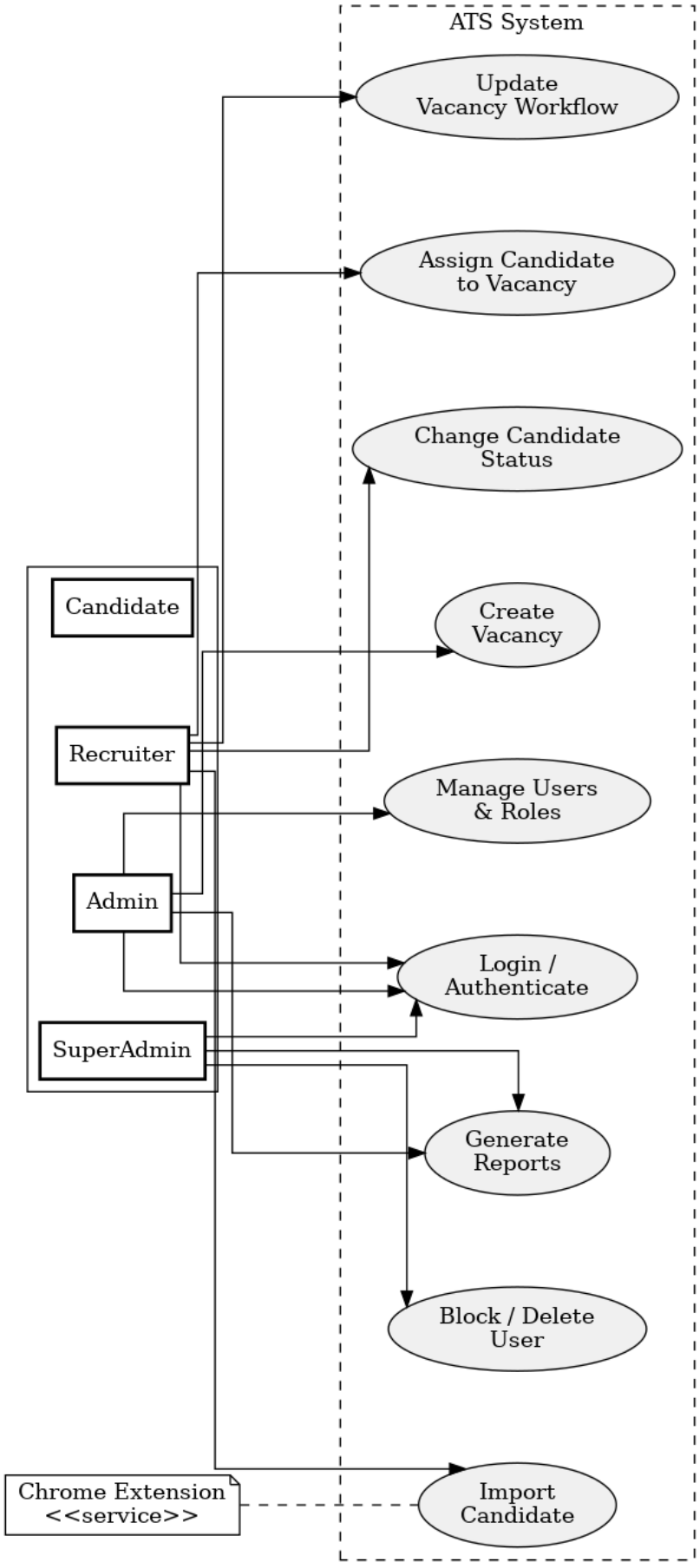

Рис.2.3.1 – Діаграма використання

Основна робота Рекрутера починається зі сценарію «Import Candidate». Під час перегляду LinkedIn-профілю рекрутер натискає кнопку в Chrome-розширенні; модуль зчитує публічні дані сторінки й миттєво надсилає їх у Kafka, звідки бекенд формує картку кандидата. Далі рекрутер за допомогою варіанта «Assign Candidate to Vacancy» прив’язує імпортованого фахівця до потрібної позиції, після чого може змінювати його етапи: «Update Vacancy Workflow» оновлює перелік стадій самої вакансії, а «Change Candidate Status» переміщує кандидата між етапами (наприклад, із Screening у Interview). Кожна зміна породжує подію assignment.stage_changed, яку одразу отримують усі зацікавлені користувачі через WebSocket-канал.

Таким чином, діаграма відображає чіткий розподіл обов’язків: створення й адміністрування вакансій — за адмінами, повсякденна рекрутингова робота — за рекрутерами, а глобальний контроль за користувачами та звітністю — за суперадміністратором. При цьому користувачі можуть робити більш чіткі налаштування і адмін може одночасно бути рекрутером, для отримання повної функціональності на платформі. Chrome-розширення виступає сервісом, що автоматизує імпорт даних, мінімізуючи ручну працю рекрутера та прискорюючи наповнення бази системи.

Також була спроектована база даних, для того щоб вона задовольняла потребам системи та виконанню поставлених в задач. (Рис.2.3.2 – Діаграма баз даних). У системі використовується реляційна база даних з кількома зв'язками між таблицями. На діаграмі видно таблиці, які містять різну інформацію про кандидатів, їх освіту, досвід роботи, навички, а також про рекрутерів і агентства.

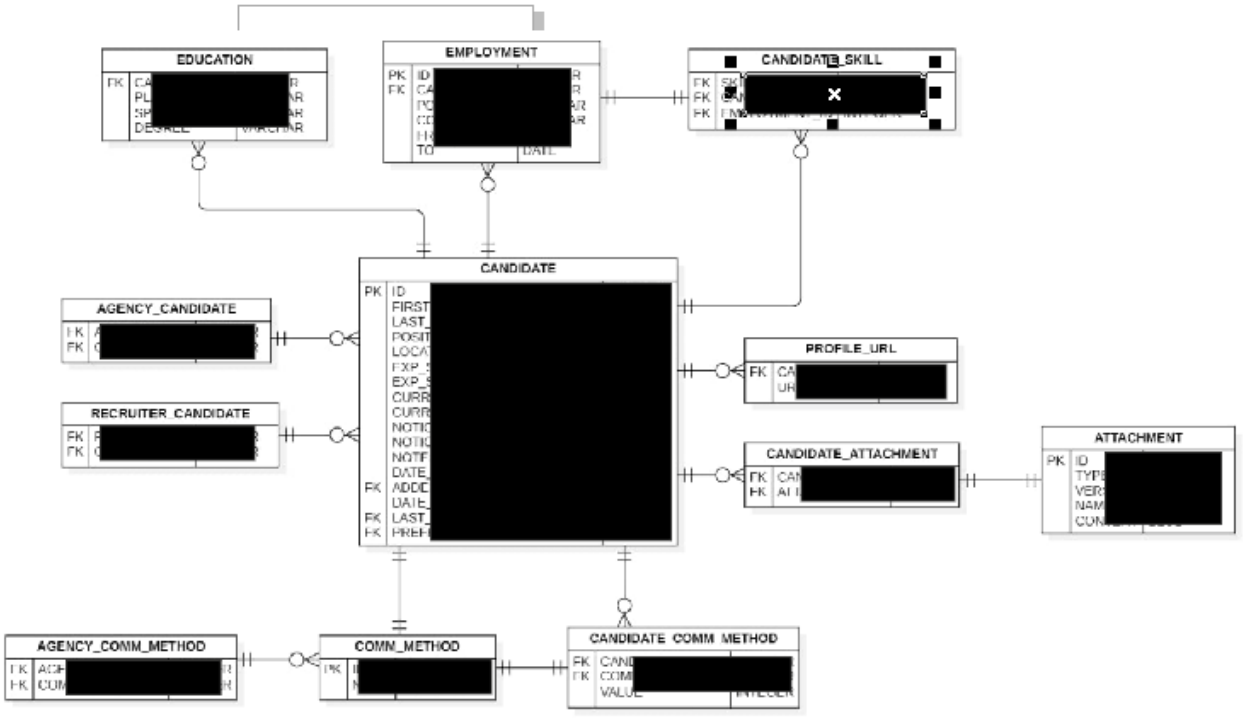

Рис.2.3.2 – Діаграма баз даних

Базовою сутністю моделі даних є таблиця CANDIDATE. У ній фіксується унікальний ідентифікатор, прізвище-ім’я-по батькові, сукупний професійний досвід, поточний статус і, що принципово, атрибут e-mail. Оскільки електронна адреса використовується для авторизації, надсилання системних сповіщень і швидкого пошуку, поле e-mail визначене як обов’язкове (NOT NULL) для всіх записів без винятку. Якщо під час імпорту з LinkedIn реальної поштової скриньки немає, платформа автоматично формує технічну адресу у форматі

```
{linkedin_uid}@bogus.stoplist.ai
```

де `linkedin_uid` — це числовий або буквено-цифровий ідентифікатор сторінки в LinkedIn. Такий e-mail зберігається у полі e-mail таблиці CANDIDATE і використовується як ключ для зіставлення записів, якщо той самий кандидат потрапляє до різних агентських баз. Такий «синтетичний» e-mail не містить персональних даних, тому не порушує вимог GDPR, але дозволяє однозначно

ідентифікувати профіль і запобігти дублюванню кандидатів усередині конкретної агенції. У різних tenant-схемах одна і та сама технічна адреса може повторюватися — це не критично, оскільки сховища ізольовані.

З кандидатом пов’язані три історичні довідники. EMPLOYMENT занотовує кожне місце роботи із зазначенням посади та діапазону дат; EDUCATION зберігає інформацію про навчальні заклади, ступені й рік закінчення; CANDIDATE_SKILL формує перелік навичок із можливістю задавати рівень володіння.

Щоб один кандидат міг співпрацювати відразу з кількома рекрутерами чи агентствами, зв’язки реалізовано через проміжні таблиці RECRUITER_CANDIDATE та AGENCY_CANDIDATE. У першій фіксується, який рекрутер додав фахівця та на якому етапі той перебуває; у другій — належність кандидата до певної агенції, що дає змогу правильно обчислювати комісії та SLA.

Контакти винесені в довідник COMM_METHOD, де стандартизовано назви каналів (e-mail, Telegram, WhatsApp тощо). Реальні значення зберігаються у зв’язувальній таблиці CANDIDATE_COMM_METHOD. Запис сюди потрапляє лише тоді, коли кандидат надав згоду відкрити конкретний канал саме цій агенції; так система виконує принцип «need-to-know» [2] і захищає приватність користувача. Додаткові допоміжні сутності PROFILE_URL (посилання на портфоліо, GitHub) і ATTACHMENT (резюме, сертифікати, мотиваційний лист) дозволяють зберігати до 4 файлів, розміром до 5Мб кожний, що відповідають кожному кандидаті та URL-адрес без дублювання основної таблиці.

Через суворі вимоги до конфіденційності даних реалізовано мультитенантну схему *schema-per-tenant*: кожна агенція працює у власному просторі PostgreSQL, ізольованому від усіх інших клієнтів. Така архітектура не лише захищає персональні дані, а й робить систему гнучкою для подальших розширень — можна безболісно додавати нові канали зв’язку або атрибути, не впливаючи на існуючі схеми інших агентств. У підсумку модель забезпечує цілісність і несуперечність інформації, підтримує багатокористувацькі бізнес-процеси рекрутингу й відповідає нормативним обмеженням щодо захисту персональних даних.

3. ПРАКТИЧНА ЧАСТИНА

3.1 Проектування інтерфейсу користувача

Для забезпечення необхідної функціональності системи були реалізовані наступні ключові функції:

1. Реєстрація та авторизація користувачів:
   - Підтримується реєстрація через електронну пошту з верифікацією через OTP-код.
   - Авторизація дозволяє користувачам отримати доступ до особистого кабінету. На загальнодоступному в інтернеті веб-сайті платформи
   - Авторизація зроблена шляхом верифікації через e-mail і надсилання автоматичного OTP-коду для входу в систему, через e-mail користувача

Рис.3.1.1 – Головна сторінка веб-сайту Stoplist.ai

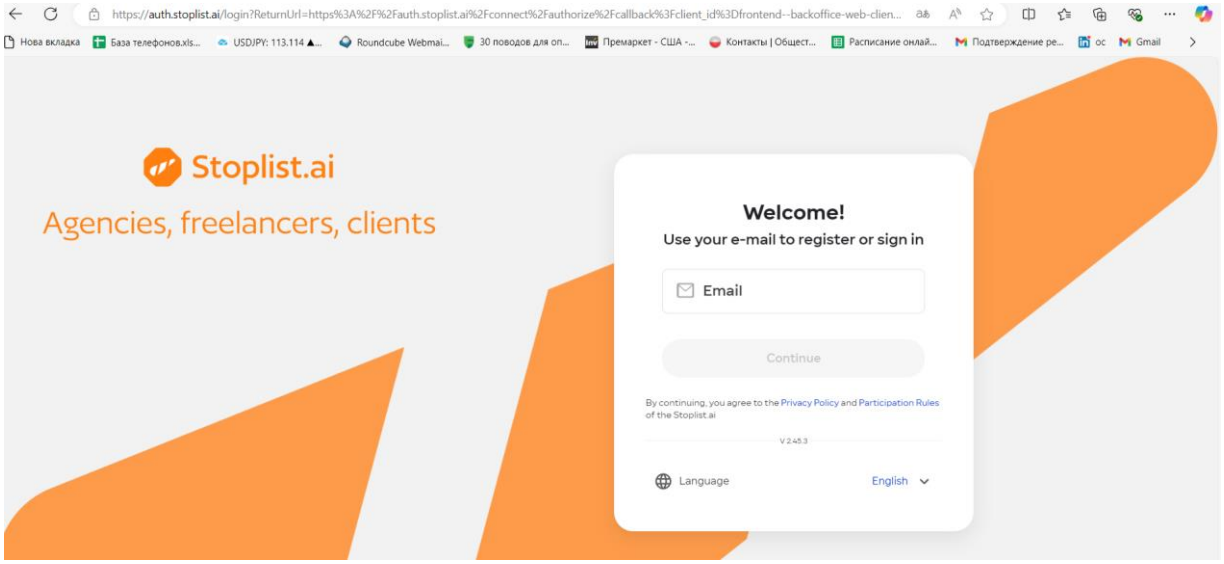

Рис. 3.1.2 – Форма авторизації через верифікацію e-mail

2. Управління вакансіями:
   - Користувачі можуть створювати, редагувати та видаляти вакансії.
   - Підтримується призначення рекрутерів до вакансій для спільної роботи.

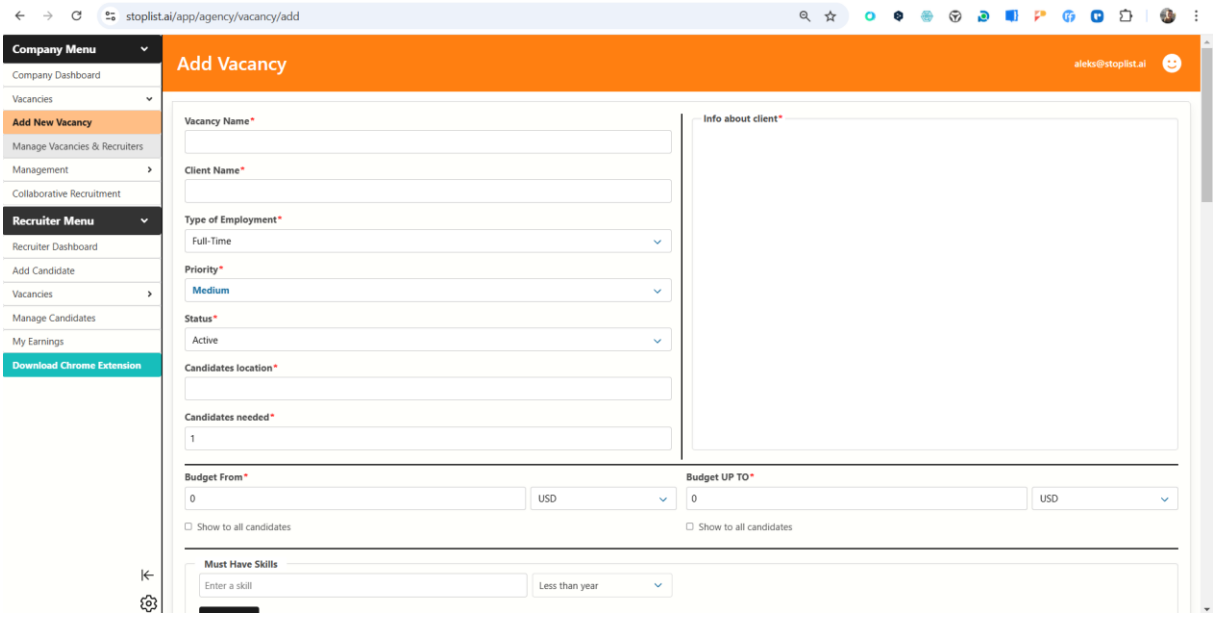

Рис.3.1.3 – Модуль управління вакансіями

Також на Рис.3.1.3 зліва розташована кнопка для завантаження користувачами розширення для Google Chrome

3. Інтеграція з LinkedIn через розширення Google Chrome:
   - Автоматизований збір даних про кандидатів з публічних профілів LinkedIn.
   - Швидке додавання кандидатів у базу даних з можливістю написання повідомлень.
   - Автоматична перевірка наявності навичок в кандидата в публічному профілі

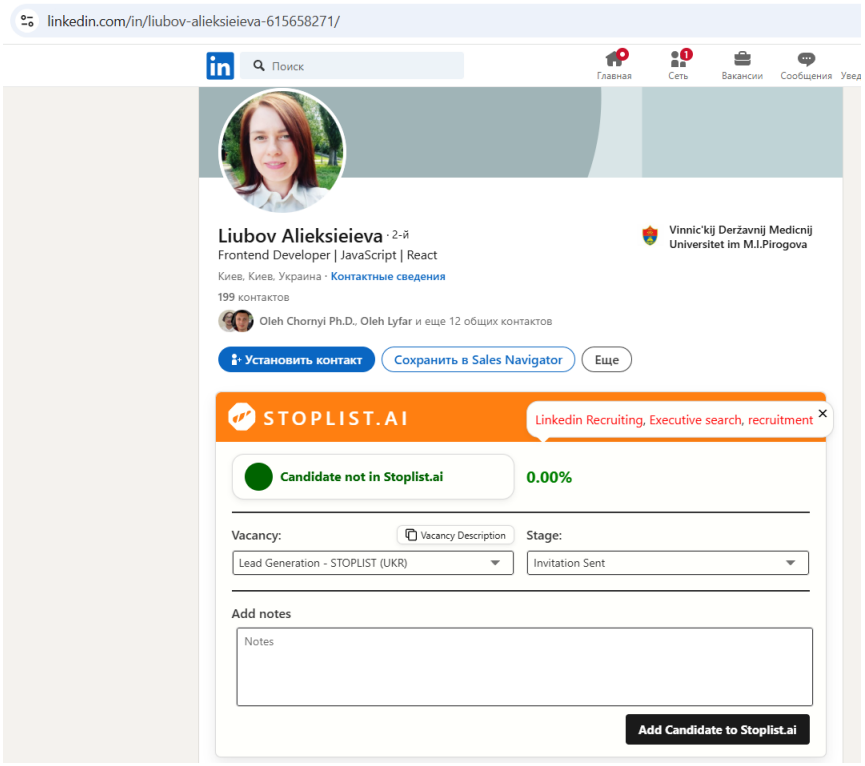

Рис. 3.1.4 – Розширення для Google Chrome в роботі з головної сторінки профіля LinkedIn

На рис. 3.1.4 показане встановлене розширення для LinkedIn в структуру DOM веб-сайту LinkedIn, та показана робота розширення в реальному часі. Розширення перевіряє навички кандидата вказані в профілі LinkedIn в автоматичному режимі, якщо ті навички були вписані при створенні вакансії, поки в системі є технічне обмеження і не можливо редагувати навички для перевірки після створення вакансії.

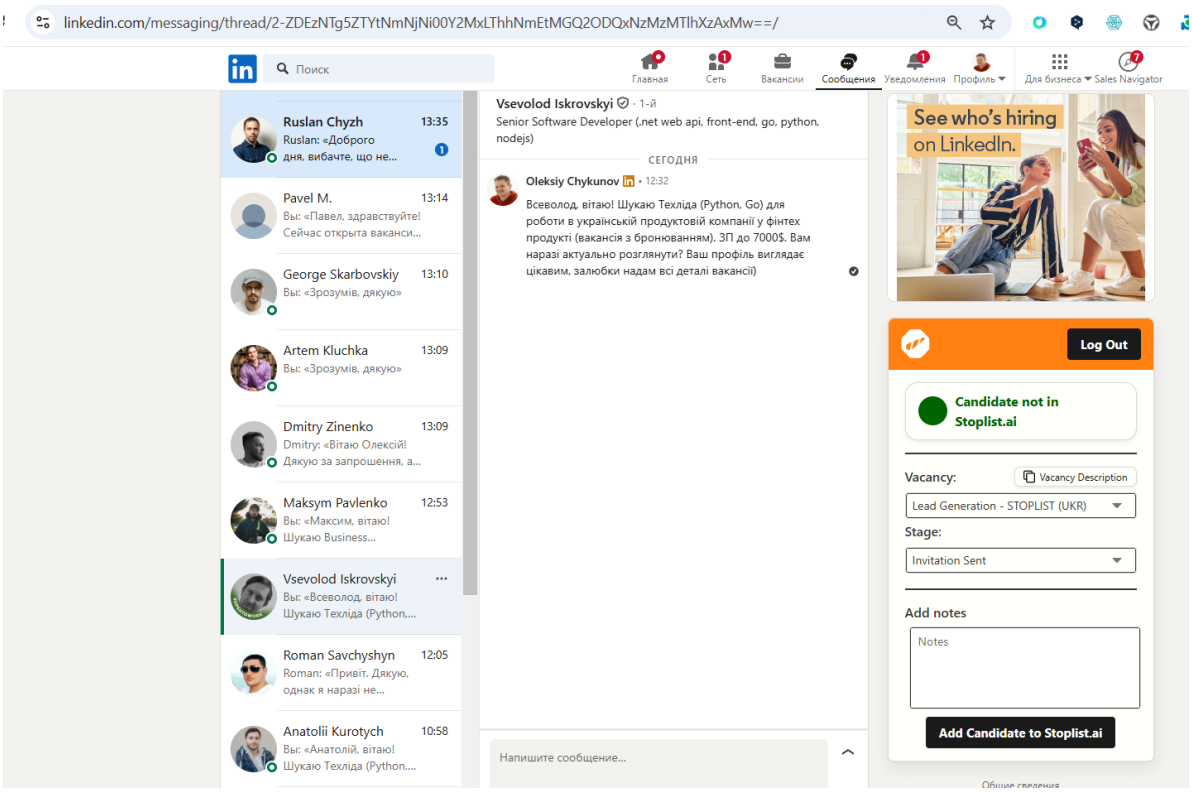

Рис. 3.1.5 – Розширення для хром в – з кнопкою Copy «Vacancy Description» для автоматичної вставки потрібного тексту опису вакансії

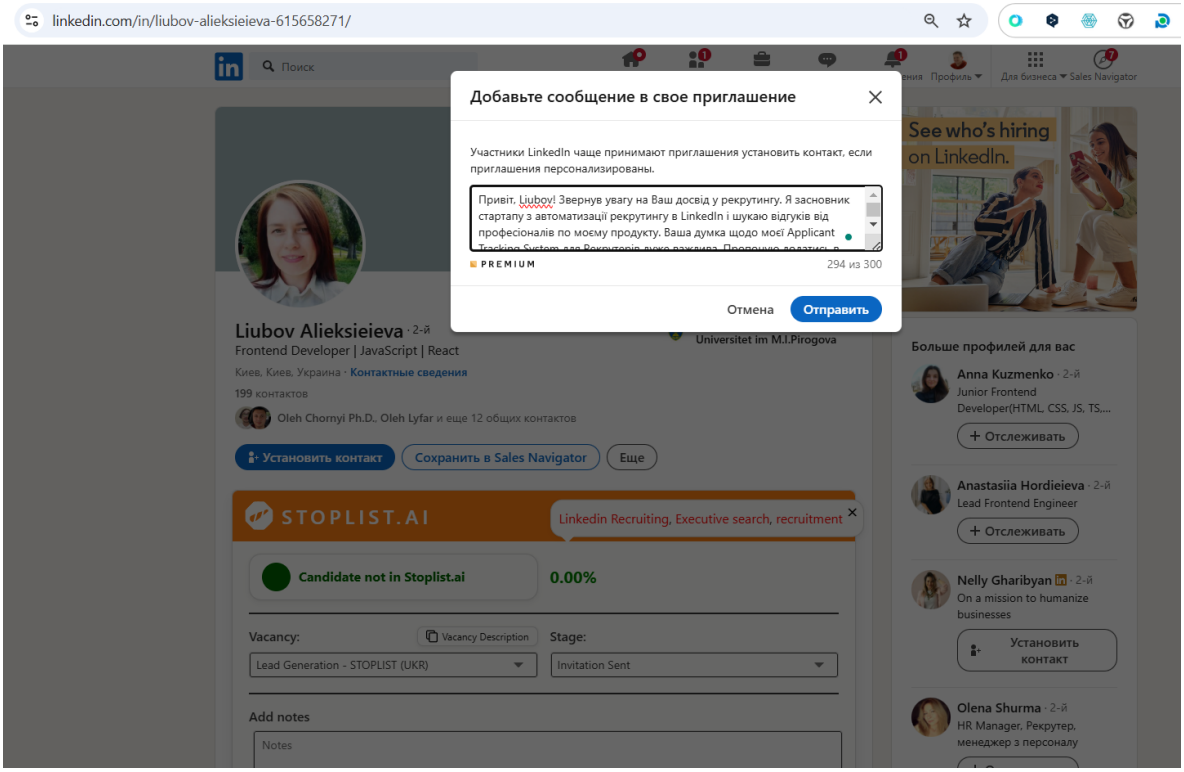

Рис. 3.1.6 – Автоматичне написання повідомлень в LinkedIn в залежності від обраної вакансії в розширенні для Google Chrome

4. Управління базою кандидатів:
   - Можливість перегляду, редагування та оновлення інформації про кандидатів.
   - Візуалізація етапів воронки закриття вакансії для кожної вакансії по кандидатам.

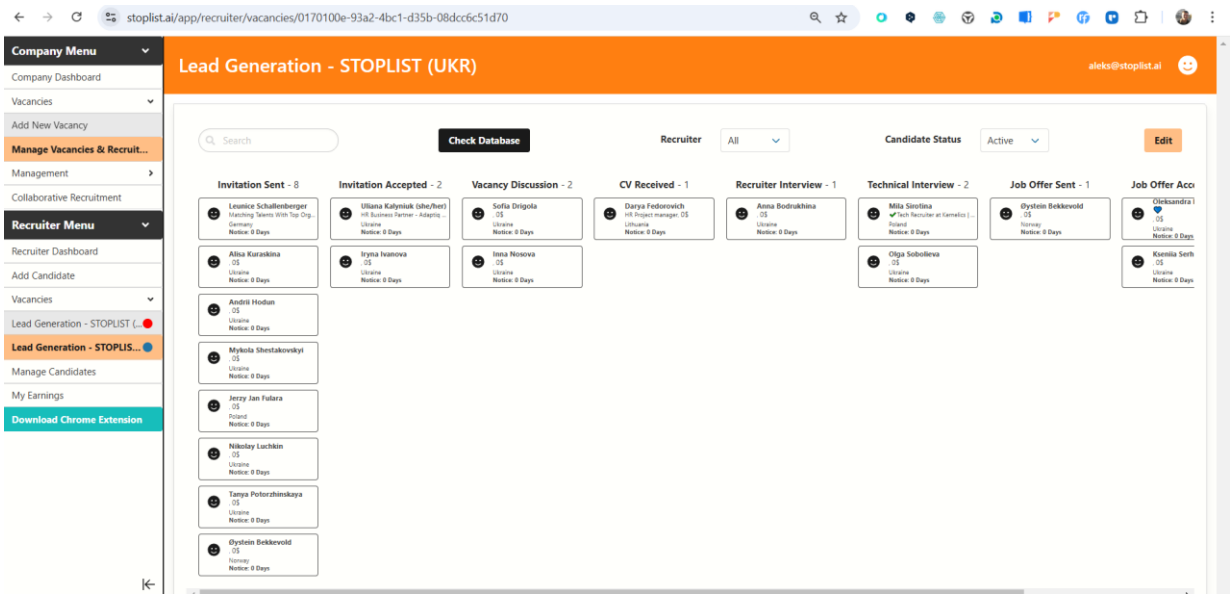

Рис. 3.1.7 – Візуалізація етапів воронки закриття вакансії

Код візуалізації етапів воронки закриття вакансій (Kanban dashboard – робоча назва), наведено в Додатку А.

5. Управління користувачами платформи для командної роботи над вакансіями
   - Розроблено інтерфейс додавання нових користувачів з різними рівнями доступу – Admin, Recruiter, Admin+Recruiter
   - Розроблено можливість включення і вимкнення користувачів юзерами з ролями Admin і Owner, Owner – є таким користувачем якого неможливо видалити
   - Зроблений неможливим для видалення і виключення профіль з рівнем доступу Owner

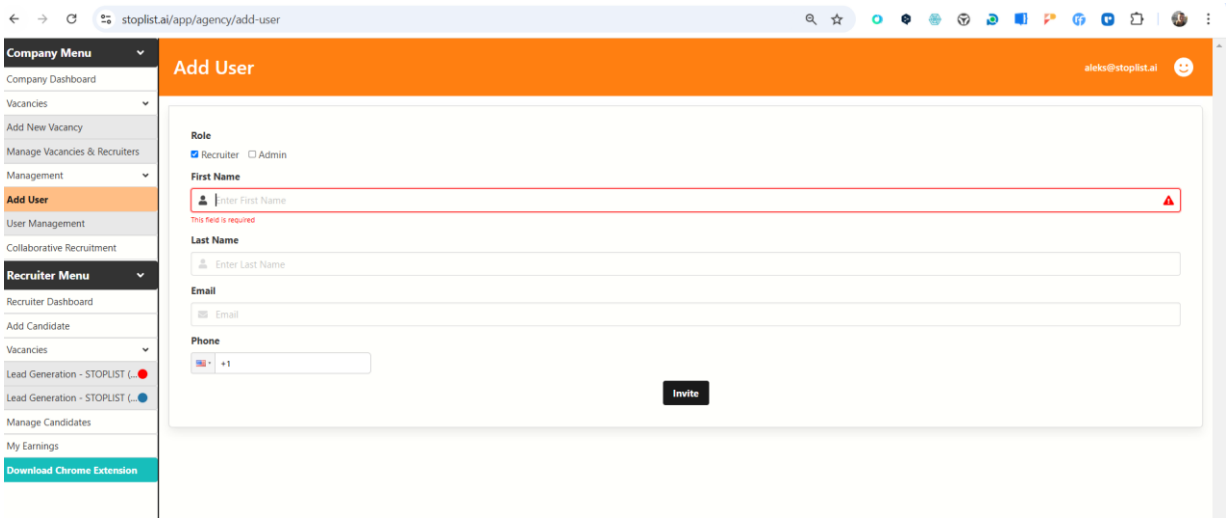

Рис. 3.1.8 – Інтерфейс для запрошення нових користувачів в свій робочий простір всереднині системи

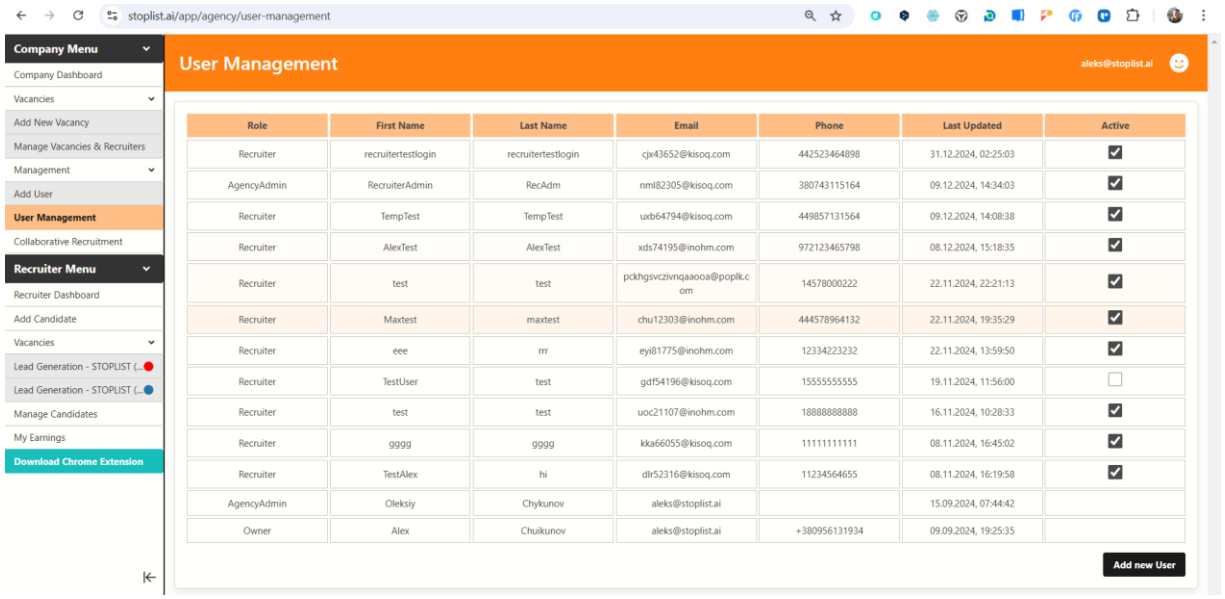

Рис. 3.1.9 – Інтерфейс керуванням включенням і вимкненням профілів користувачів платформи всередині простору роботу однієї компанії

У наступних пунктах було наведено детальний опис графічного інтерфейсу та ілюстрації роботи програмного продукту.

Додаткові ілюстрації роботи створеного програмного забезпечення приведені нижче.

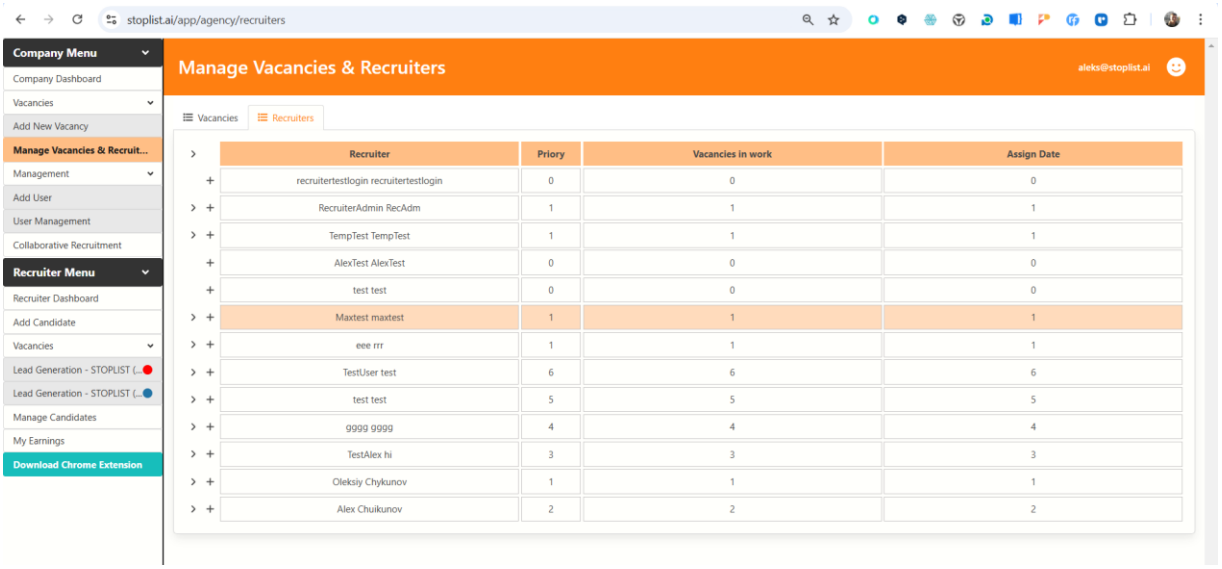

Рис. 3.1.10 – Інтерфейс керування навантаженням рекрутерів

На рис.3.1.10 зображено інтерфейс керування навантаженням рекрутерів, який дозволяє дуже швидко оцінити скільки вакансій знаходиться в роботі у кожного рекрутера і відрегулювати навантаження і або назначити додаткові вакансії в роботу, або перевести якісь вакансії з одного на іншого рекрутера.

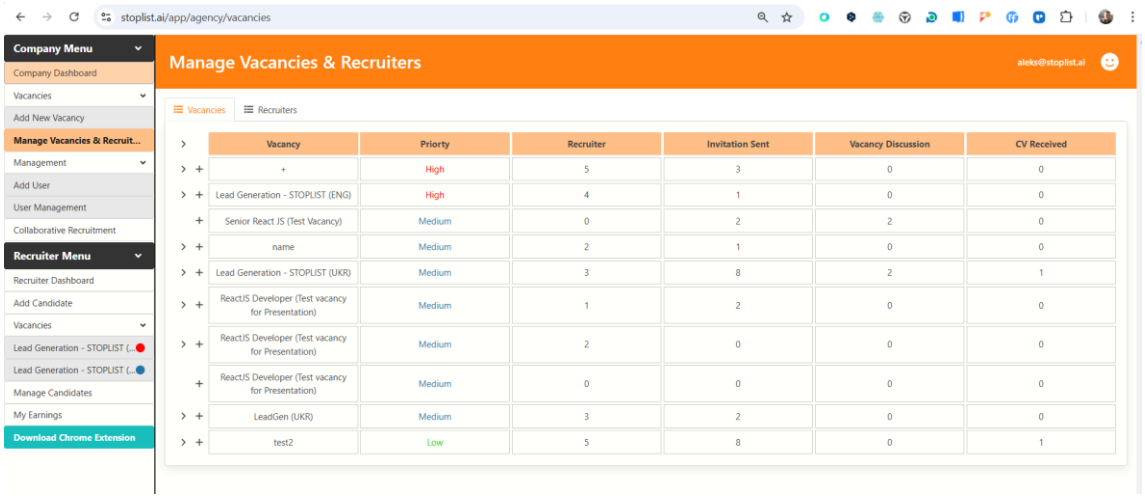

Рис.3.1.11 – Інтерфейс керування вакансіями компанії

На Рис. 3.1.11 – зображено інтерфейс керування вакансіями де видно з доступом Admin, Admin/Recruiter, Owner – всі вакансії що наявні в роботі в агенції. Натомість в рекрутер-меню відображені лише вакансії за якими закріплено самого Owner-а або Admin/Recruiter-а. У просто Admin-а – рекрутерського меню не

відображається і не доступний з його ID використання розширення для гугл хрому. Те було зроблено для того, щоб мій продукт не брав оплату в користувачів за ролі рівня Admin, які не здійснюють рекрутерську діяльність.

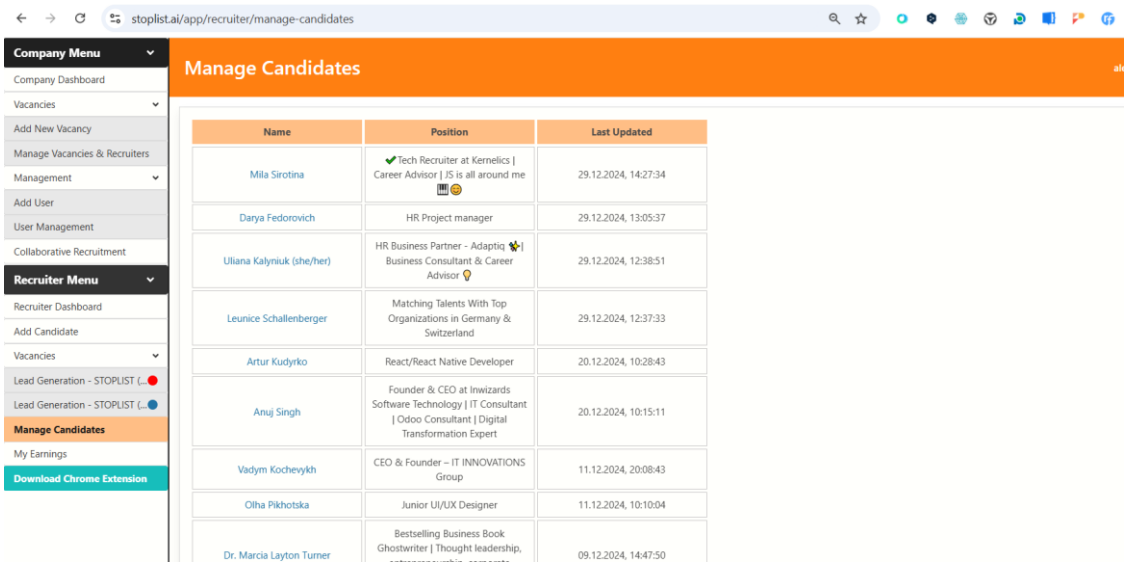

Рис.3.1.12 – Керування своїми кандидатами

На Рис. 3.1.12 – форма яка дозволяє побачити всіх кандидатів яких додав користувач сам як рекрутер. На цій формі ми тимчасово відключили пошук, який було реалізовано на фронтенді але він працював доволі швидко тому було вирішено не переводити його на бекенд. Наразі відбуваються роботи над дублюванням цього пункту в меню компанії – для того щоб користувачі з адмінськими правами доступу могли шукати всіх кандидатів, що були додані всіма рекрутерами в базу даних компанії

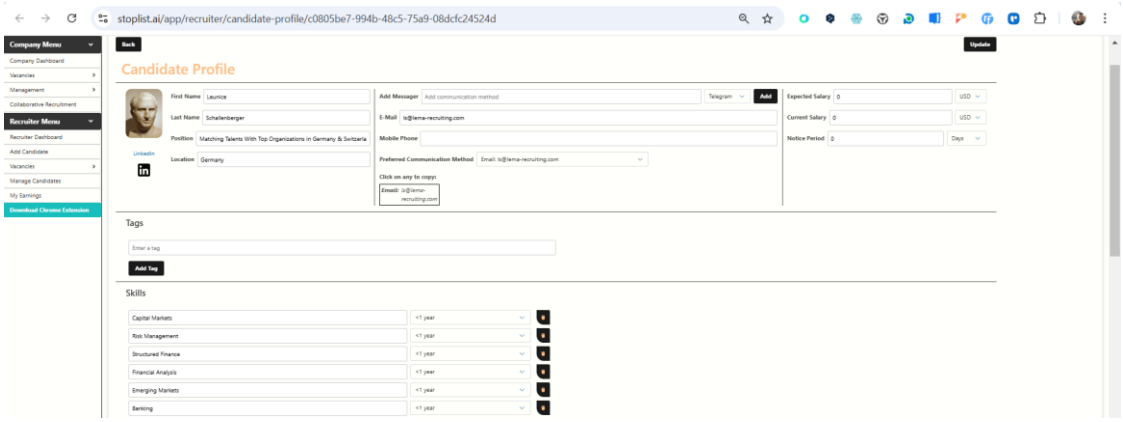

Рис.3.1.13 – Детальний профіль кандидата

На рис.3.1.13- детальний профіль кандидата, що автоматично заповнюється через парсинг загальнодоступної сторінки в LinkedIn при використанні розширення. Але в LinkedIn є система захисту, яку ми обійшли для того, щоб наші користувачі могли спокійно користуватись нашим розширенням і автоматично заповнювати всі ці данні при тому, не DDoS-ити запитами від нашого розширення LinkedIn.API

3.2 Опис програмних модулів

Програмне забезпечення Stoplist.ai реалізовано за мікросервісною архітектурою з доменно-орієнтованим поділом. Кожен мікросервіс охоплює окремий предметний сегмент (вакансії, кандидати, користувачі, сповіщення) та має власну схему даних у СУБД PostgreSQL. Подібна ізоляція мінімізує міжсервісні залежності, спрощує супровід і дає змогу масштабувати окремі компоненти незалежно від решти системи.

Вхідні HTTP-запити та WebSocket-підключення надходять до єдиного шлюзу Gateway, розгорнутого на платформі Node.js. Gateway виконує автентифікацію за допомогою JWT, перевіряє повноваження користувача згідно з RBAC-моделлю, а далі маршрутизує запити до відповідних бек-енд-служб. Внутрішня взаємодія мікросервісів відбувається як синхронно (REST-інтерфейси), так і асинхронно через шину подій Apache Kafka [14]; це забезпечує гарантовану доставку повідомлень і нечутливість до короткочасних відмов окремих вузлів.

Процес безперервної інтеграції та розгортання (CI/CD) реалізовано засобами GitHub Actions: після кожного коміту виконується збірка Docker-образів, прогін автоматизованих тестів і деплой у керований кластер Google Kubernetes Engine. Керування секретами здійснюється через HashiCorp Vault, а централізоване логування й моніторинг — за допомогою стеку Prometheus + Grafana та Fluent Bit.

Наведена нижче таблиця 3.2.1 - Програмні модулі системи Stoplist.ai, систематизує програмні модулі платформи: подає їх призначення, основні канали взаємодії та застосовані технології, відображаючи послідовність обробки типового користувацького запиту.

Таблиця 3.2.1 – Програмні модулі системи Stoplist.ai

№	Назва модуля	Призначення й функціональність	Взаємодія	Технології
1	Gateway (Node.js)	Єдина точка входу для REST- і WebSocket-запитів; валідація JWT, RBAC-фільтр, агрегування відповідей мікросервісів, трансляція push-подій.	HTTPS до `vacancy-`, `candidate-`, `user-service`; Kafka “`notification`” → WebSocket. `/ws/notify`	Node.js 18, Express 4, Socket.IO
2	vacancy-service (.NET)	CRUD вакансій, керування етапами (pipeline), публікація подій `vacancy.created/updated`.	Запити від Gateway; пише/читає PostgreSQL (`Vacancies`, `VacancyStage`) “`recruitment`”.	ASP.NET Core EF Core
3	candidate-service (.NET)	Зберігання карток кандидатів; імпорт із LinkedIn; API `/candidates/*`.	JSON із Gateway; Kafka “`candidate.created`”	ASP.NET Core, PostgreSQL (`Candidates`, `CandidateSkill`)

Продовження Таблиці 3.2.1

4	application-service (.NET)	Пов’язує кандидатів із вакансіями; оновлює статуси у таблиці `Assignments`; застосовує правила «світлофора».	Слухає Kafka-подію `candidate.created`; пише у БД і публікує `assignment.stage_changed`.	Worker Service .NET
5	user-service (.NET)	CRUD користувачів, OTP-входи, генерація JWT; таблиці `Users`, `Roles`, `OtpTokens`.	Відповідає на `/auth/otp`, `/auth/login`; пише Redis TTL-код OTP.	ASP.NET Core, Redis
6	notification-worker	Надсилання e-mail та `push`; слухає `assignment.stage_changed`.	Kafka → SendGrid / WebPush.	.NET Worker, Confluent.Kafka, SendGrid SDK
7	Chrome Extension	Парсинг DOM профілю LinkedIn, формування JSON-payload; надсилання POST `/api/v1/import`; UI світлофора.	content-script → Gateway; WebSocket.	Manifest v3, Fetch API
8	React SPA	TopBar, Hero, Dashboard, Kanban, модуль запрошень користувачів.	REST через Gateway; WebSocket push.	React 18, Vite, Tailwind
9	analytics-service	Агрегація метрик time-to-hire; конверсія етапів (розробка).	Планується cron-job `report-worker`; REST `/reports/*`	.NET 7, TimescaleDB

Продовження Таблиці 3.2.1

10	CI/CD pipeline	Збирання Docker-образів, тестування, деплой у Google Kubernetes Engine.	GitHub Actions → Helm Charts.	Docker, Helm, Kubernetes

3.3 Опис результатів тестування

З метою перевірки працездатності, логічної цілісності та відповідності функціональних можливостей програмного забезпечення Stoplist.ai поставленим вимогам було проведено комплексне ручне тестування ключових модулів системи. Тестування виконувалося без використання спеціалізованих фреймворків автоматизації, безпосередньо в середовищі розробки та у тестовій версії інтерфейсу користувача.

У процесі перевірки особливу увагу було приділено таким компонентам системи, як процедура реєстрації та авторизації користувачів, створення вакансій, додавання та розподіл ролей між рекрутерами й адміністраторами, а також логіки роботи розширення для Google Chrome, що відповідає за імпорт кандидатів із LinkedIn. Було здійснено перевірку основних сценаріїв взаємодії користувачів із системою, включаючи відображення вакансій, призначення виконавців, а також доступність функцій згідно з ролями (Admin, Recruiter, SuperAdmin).

Під час тестування було підтверджено правильну реалізацію обмежень доступу: користувачі не мали можливості переглядати інформацію поза межами призначених їм ролей або організаційної структури. Також було успішно протестовано механізм генерації OTP-кодів для верифікації електронної пошти, і підтверджено стабільну роботу при додаванні нових користувачів до компанії.

Окремо перевірялась поведінка розширення в браузері Chrome. Тестування підтвердило коректне зчитування публічних профілів кандидатів на LinkedIn, правильне заповнення полів у внутрішній базі даних, а також обмеження на дублювання кандидатів у межах однієї вакансії.

Результати тестування засвідчили стабільну роботу системи за основними сценаріями, відсутність критичних помилок, а також достатню готовність до подальшого практичного використання та розширення функціональності. Виявлені незначні візуальні недоліки інтерфейсу були оперативно усунуті в рамках завершального етапу розробки.

Для перевірки надійності механізму входу до системи Stoplist.ai було проведено тестування основних сценаріїв, пов’язаних з автентифікацією через електронну пошту та OTP-код. Особливу увагу приділено обробці помилкових даних, повторному входу та захисту сесій користувача. Нижче наведено результати тестування:

Таблиця 3.3.1. – Результати тестування функціоналу авторизації

№	Опис тесту	Очікуваний результат	Фактичний результат
1	Введення коректного email та OTP	Успішна авторизація	Успішна авторизація
2	Введення некоректного OTP	Повідомлення про помилку	Помилка відображена
3	Спроба входу без OTP	Запит на OTP	Запит надіслано
4	Повторний вхід з тим самим браузером	Вхід без повторної верифікації	Успішно виконано
5	Введення неіснуючого email	Відмова у вході	Доступ заборонено
6	Спроба ввести код після завершення часу дії	Повідомлення про прострочення	Отримано помилку часу
7	Скидання OTP після кількох невдалих спроб	Надсилання нового коду	Новий код отримано

Усі перевірені сценарії авторизації виконувалися вірно. Система стабільно реагує на помилкові або невалідні дані, а також забезпечує плавний вхід повторного користувача без втрати сесії. Це дозволяє зробити висновок про надійність реалізованої логіки автентифікації.

Другою критично важливою частиною системи є інтерфейс управління вакансіями, що дозволяє створювати, редагувати, видаляти вакансії та призначати відповідальних рекрутерів. Було проведено тестування як позитивних сценаріїв, так і помилки взаємодії з інтерфейсом. Результати наведені в таблиці нижче.

Таблиця 3.3.2. – Результати тестування функціоналу управління вакансіями

№	Опис тесту	Очікуваний результат	Фактичний результат
1	Створення вакансії з усіма полями	Вакансію створено	Успішне створення
2	Редагування існуючої вакансії	Зміни збережено	Дані оновлено
3	Призначення рекрутера до вакансії	Рекрутера призначено	Додано в інтерфейсі
4	Спроба створити вакансію без назви	Помилка валідації	Помилка показана
5	Видимість вакансії для призначених рекрутерів	Вакансія відображається	Доступ є

Усі функції модуля вакансій функціонують як очікується. Інтерфейс дозволяє адміністраторам гнучко взаємодіяти з вакансіями, не допускаючи некоректного введення чи дублювання записів. Система демонструє стабільність та відповідність очікуванням користувача.

Chrome-розширення є ключовим інструментом рекрутера, що дозволяє автоматизувати збір профілів кандидатів з LinkedIn. Під час тестування перевірялися як основні дії (імпорт даних), так і коректність візуалізації стану

кандидатів за допомогою «світлофора». Таблиця нижче узагальнює результати перевірки.

Таблиця 3.3.2. – Результати тестування функціоналу додатку для Google Chrome

№	Опис тесту	Очікуваний результат	Фактичний результат
1	Імпорт кандидата з LinkedIn	Кандидата додано	Імпорт успішний
2	Перевірка стану світлофору	Стан показано вірно	Колір відповідає статусу
3	Заповнення шаблону повідомлення	Повідомлення сформовано	Шаблон працює
4	Авторизація в розширенні	OTP авторизація	Користувача впізнано
5	Блокування дублювання повідомлень	Попередження про повтор	Функція спрацювала
6	Показати статус кандидата іншим рекрутерам	Статус синхронізовано	Стан видно
7	Спроба відправки без вибраної вакансії	Повідомлення про помилку	Помилка показана

Розширення стабільно виконує свою роль автоматизації, значно зменшуючи ручну роботу рекрутера. Механізм світлофору забезпечує прозору взаємодію в межах команди та дозволяє уникати дублювання контактів з кандидатами. Реалізований функціонал відповідає очікуванням та доводить ефективність архітектурного рішення щодо інтеграції з браузером.

На Рис. 3.3.1. представлено фрагмент робочого середовища з результатами тестування окремих компонентів системи Stoplist.ai. Зліва показано виконання unit-тестів, які перевіряють базову логіку на рівні окремих функцій і модулів, включаючи обробку рекрутерських запитів, відповідність JWT-токенів, та обробку некоректних відповідей сервера.

Справа зображено приклад ручного API-тестування через інструмент Postman, де здійснюється запит авторизації користувача через OTP-код. У

відповідності до архітектури системи, пароль користувача не зберігається і не передається — замість цього відбувається вхід шляхом генерації одноразового 6-значного коду, який надсилається на пошту. В тестовому середовищі було використано тимчасовий email через сервіс «10 Minute Mail» для перевірки швидкості доставки та стабільності обробки токена. У відповіді API видно, що запит на отримання токену пройшов успішно, й система повертає JWT у форматі, придатному для подальшої авторизації на захищених ендпоінтах.

Тестування дозволило переконатись у правильній реалізації логіки авторизації, відповідності відповіді API очікуванням та стабільності взаємодії між frontend, backend і email-сервісом навіть в умовах штучного таймауту. Таким чином, зображення підтверджує як функціональну, так і інтеграційну цілісність реалізованої системи.

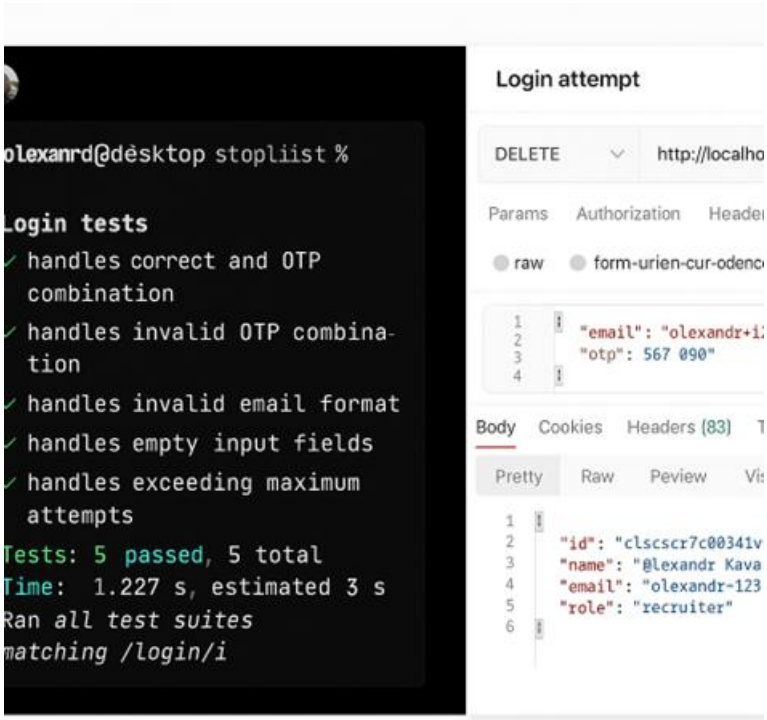

Рис. 3.3.1 – Приклад перевірки API-запитів та unit-тестів у процесі тестування

4. БЕЗПЕКА ЖИТТЄДІЯЛЬПОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні чинники безпеки життєдіяльності

Ергономіка як міждисциплінарна наукова галузь досліджує закономірності взаємодії «людина – машина – середовище» (ЛМС) та спрямована на гармонізацію технічних і організаційних рішень із можливостями й обмеженнями оператора ПК [4]. Для фахівців цифрової економіки (розробників, рекрутерів, проєктних менеджерів) саме робота за комп'ютером залишається ядром професійної діяльності (6-8 год за зміну), тож вичерпний аналіз ергономічних ризиків є обов'язковим компонентом системи управління охороною праці (СУОП).

Таблиця 4.1.1. Ключові чинники ергономічної небезпеки

Група факторів	Механізм формування ризику	Потенційні наслідки
Статичні та динамічні навантаження	Фіксована поза сидячи з нахилом тулуба > 20°, горизонтальне розміщення передпліч, відсутність поперекової опори та мікрорухів → підвищений внутрішньодисковий тиск на L4-L5, ішемія паравертебральних м'язів [6]	Остеохондроз, міофасціальний больовий синдром, карпальний тунель
Зорові навантаження	Яскравість екрана перевищує яскравість оточення > 3 рази, мерехтіння > 90 Гц, відстань око-екран < 45 см [7]	Спазм акомодації, синдром «сухого ока», астенопія
Когнітивно-емоційна напруга	Високий темп оброблення вхідної інформації (до 2000 подій/зміну), мультизадачність, відповідальність за якість даних [4]	Digital fatigue, когнітивні помилки, вигорання, тривожно-депресивні реакції

Продовження Таблиці 4.1.1.

Небажані мікрокліматичні впливи	Перевищення нормативів CO_2 (> 1000 ppm), сухість повітря < 40 %, шум > 50 дБА [6]	Зниження уваги на 12-15 %, головний біль, респіраторні подразнення

Нормативно-правові рамки в Україні є:

- Закон України «Про охорону праці» (ст. 13 та 17) покладає на роботодавця обов’язок створювати безпечні та нешкідливі умови праці, проводити попередні та періодичні оцінки ризиків, а також навчати персонал безпечним методам роботи [5].
- ДСанПіН 3.3.2.007-98 - установлює гранично-допустиму тривалість безперервної роботи з ВДТ (не більше 2 год поспіль), режими праці / відпочинку, параметри мікроклімату (22 ± 2 °C, 40-60 % RH) та освітленості (300-500 лк) [6].
- ДСТУ ISO 9241-5:2001 - регламентує геометрію робочого місця: кут огляду −15…-20°, висоту плечової опори 200-250 мм, відстань від очей до екрана 500-700 мм [7].
- ДСТУ ISO 26800:2019 - закладає концепцію сумісності робочої системи з антропометричними, когнітивними та сенсорно-моторними характеристиками оператора [9].

Дотримання цих вимог підлягає державному нагляду; невиконання спричиняє адміністративні штрафи (ст. 41-1 КУпАП) і може трактуватися як грубе порушення трудового законодавства.

Фізіологічні й психофізіологічні наслідки.

За офіційною статистикою МОЗ, 61 % усіх профзахворювань працівників ІТ-галузі пов’язані з опорно-руховим апаратом, 24 % — із зоровими дисфункціями, решта — з психоемоційними розладами [8]. Метаболічна вартість сидячої роботи (≈ 1,5 МЕТ) у поєднанні з гіподинамією сприяє ожирінню та інсулінорезистентності, а тривала когнітивна напруга активує вісь «гіпоталамус –

гіпофіз – наднирники», підвищуючи рівень кортизолу, що пришвидшує вичерпання ресурсів уваги.

Методи оцінювання ризиків та профілактики

Моделювання небезпечних ситуацій та профілактика була зроблена для прогнозування травматизму у системі ЛМС застосовують логічні схемні моделі (НУ → НД → НС) [4], що враховують статистично залежні/незалежні події . Ймовірність виникнення небезпечної ситуації $P_{НС}$ обчислюють за формулами множинної чи сукупної ймовірності, що дозволяє ранжувати ризики і планувати пріоритетні заходи:

$$P_{\text{НС}} = \prod_{i=1}^{n} P(\text{Подія}_i).$$

На практиці профілактика реалізується через:

- чек-лист ергономічного аудиту робочих місць (щокварталу);
- установлення регульованих столів-консолей «sit-stand»;
- гібридний графік (2–3 дні/тиждень дистанційно) для чергування видів діяльності;
- інструктажі з охорони праці згідно з Типовим положенням (Держпраці, постанова № 359, 2018 р.).

Інженерно-технічні заходи:

- Регульовані столи-консолі (діапазон 650-1200 мм) для чергування поз «сидячи / стоячи», що скорочує статичне напруження м’язів trapezius на 32 %.
- Крісла з динамічним сидінням (тривісною підвіскою) та поперековою підтримкою глибиною 40-60 мм.
- Екрани з фільтром blue-light < 450 нм та частотою ≥ 100 Гц зменшують візуальну втому на 18-22 %.

Організаційні та програмні рішення:

- Режим 50/10 (50 хв роботи + 10 хв активної перерви) з автоматизованим нагадувачем.
- Мікропаузи «20-20-20» — кожні 20 хв глянути 20 сек на об’єкт за 6 м — доведено знижують спазм акомодації.

- Вбудовані трекери активності (плагіни Time Out, Stretchly) та інтеграція з standing-desk полегшують дотримання режиму.

Навчання та залучення персоналу та підхід «participatory ergonomics» передбачає:

1. залучення працівників до вибору меблів;
2. регулярні тренінги з профілактичної гімнастики;
3. зворотний зв’язок через анонімні опитувальники й коригувальні заходи за результатами кожного кварталу.

Ергономічні ризики при роботі з ПК — це поліфакторна проблема, яка об’єднує фізіологічний (МСР, зорові порушення), психофізіологічний (вигорання) та соціально-економічний (втрати продуктивності, лікарняні) виміри. Ефективна профілактика спирається на четверо принципів:

1. Комплаєнс із нормативами (Закон № 2694-XII [5], ДСанПіН, ISO).
2. Інженерно-технічне усунення факторів ризику (ергономічні меблі, оптимізовані ВДТ).
3. Організаційне керування режимами праці (цикл 50/10, standing desk, мікропаузи).
4. Безперервне навчання персоналу та залучення його до процесу «участь + відповідальність».

Такий інтегрований підхід забезпечує збереження здоров’я працівників, покращує якість продукції (послуг) та зменшує сукупні витрати бізнесу на компенсацію професійних ризиків.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Заходи з охорони праці користувачів ПК розглядають в трьох аспектах: соціальному, психологічному та медичному. У соціальному плані розв'язання цих проблем пов'язане з оптимізацією умов життя, праці, відпочинку, харчування,

побуту, розвитком культури, транспорту. Значне місце у профілактиці розладів здоров'я належить психології праці. Тому заходи, пов'язані з формуванням раціональних виробничих колективів, у яких відсутня психологічна несумісність, сприяють зменшенню нервово-психічного перенапруження, підвищенню працездатності та ефективності праці [6]. Особливої значущості у користувачів відеодисплейних терміналів набуває психоемоційний стрес, який більшою або меншою мірою проявляється у кожного з них. Оскільки цю проблему відразу вирішити неможливо, доцільно на рівні підприємства, організації послідовно усувати такі виробничі умови, які є сприятливими для розвитку емоційного стресу. Значна роль у профілактиці захворювань користувачів ПК відводиться медицині. Існує перелік профілактичних заходів для користувачів ПК, що включає як складові первинної профілактики здоров'я (професійний відбір), так і вторинної, яка направлена на зниження ймовірності розвитку перевтоми та перенапруження. Ці комплексні заходи спрямовані на відновлення функціонального стану зорового та опорно-рухового апарату. [10]

Як було встановлено у п. 4.1.1, ключовою передумовою зниження професійних ризиків ІТ-персоналу є приведення просторових, технічних і організаційних характеристик робочої зони у відповідність до вітчизняних нормативів охорони праці й сучасних стандартів ергономіки. Нижче наведено практичний регламент облаштування місця оператора ПК, деталізований з урахуванням вимог Закону України «Про охорону праці» [5] та ДСанПіН 3.3.2.007-98 [6], ДСТУ ISO 9241-5:2001 [7].

Примітка. Регулювання здійснюється за зростом у межах *5-го–95-го* процентилів працівників організації згідно з Методрекомендаціями Держпраці № 73/2020.

Відповідно до встановлених гігієнічно-санітарних вимог (ГОСТ 12.1.005-88, СН 4088-86) роботодавець зобов’язаний забезпечити в приміщеннях з ВДТ оптимальні параметри виробничого середовища [10].

Основні вимоги до виробничого приміщення для експлуатації ВДТ

– воно не може бути розміщено у підвалах та цокольних поверхах;

– площа на одне робоче місце в такому приміщенні повинна становити не менше 6,0м2, а об’єм не менше 20,0 м3;

– воно повинно мати природне та штучне освітлення відповідно до СНіПП-4-79;

– в ньому мають бути шафи для зберігання документів, магнітних дисків, полиці, стелажі, тумби тощо, з урахуванням вимог до площі приміщення;

– щоденно проводити вологе прибирання;

– поруч з приміщенням для роботи з ВДТ мають бути обладнані:

– побутова кімната для відпочинку під час роботи;

– кімната психологічного розвантаження.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним ВДТ, має здійснюватись системою загального рівномірного освітлення.

Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ.

Тривалість регламентованих перерв під час роботи з ЕОМ за 8-годинної денної робочої зміни залежно від характеру праці: 15 хвилин через кожну годину роботи – для розробників програм зі застосуванням ЕОМ; 15 хвилин через кожні дві години – операторів із застосуванням ЕОМ; 10 хвилин після кожної години роботи за ВДТ для операторів комп’ютерного набору.

У випадках, коли виробничі обставини не дозволяють застосовувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 годин.

Для зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу і запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які передбачені ДСанПіН 3.3.2.007-98, в тому числі і для сеансів психологічного розвантаження у кімнаті з відповідним інтер’єром та кольоровим оформленням

Таблиця 4.2.1 – Антропометрична сумісність «користувач-обладнання»

Параметр	Рекомендоване значення	Нормативне джерело
Висота сидіння	420–550 мм, регулювання ≥120 мм	ISO 9241-5 §5.2 [7]
Глибина сидіння	370–430 мм, з зазором 25–50 мм від краю до підколінної ямки	ДСанПіН п. 3.2 [6]
Підлокітники	Висота 200–250 мм, регулювання кутів ≥30°	[7]
Висота робочої поверхні	680–760 мм (сидячи); 950–1150 мм (стоячи) або електропривід 650–1250 мм	ДБН В.2.2-3:2014, додаток Г

Розміщення візуальних та керувальних пристроїв

- Монітор
  - центр екрана → на 0–100 мм нижче горизонталі ока;
  - кут нахилу – від -5° до -15°;
  - відстань «око–екран» – *500–700 мм* при діагоналі 24" (коефіцієнт масштабування шрифту ≥125 %).
- Клавіатура – на рівні ліктів (±20 мм); кут тильної поверхні кисті ≤10°.
- Маніпулятор – у проєкції плечового суглоба, без зміщення у фронтальній площині.

Досить важливим є вимоги до освітлення приміщень, оскільки відомо, що тривала робота за комп’ютером та з документами при недостатньому рівні освітленості може призвести до значного перенапруження зору. Природне освітлення має забезпечувати коефіцієнт природної освітленості (КПО) не нижче ніж 1,5%. Для регулювання рівня освітлення природним світлом бажано застосовувати жалюзі. Робоче місце, обладнане ПК повинно бути розташоване так, щоб уникнути попадання в очі прямого сонячного світла. Штучне освітлення приміщення має бути обладнане системою загального рівномірного освітлення. Застосування світильників без розсіювачів та екрануючих сіток забороняється.

Світлотехнічні вимоги, до робочого місця мають бути наступними освітленість робочої поверхні в приміщеннях для роботи з дисплеями і відеотерміналами, дисплейних залах має складати 300–500 лк (LED 4000 K, Ra ≥ 80), при комбінованому освітленні, або 400лк при загальному освітленні, при цьому коефіцієнт пульсації освітлення не може перевищувати 10%, а показник дискомфорту – не більше 15% (ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ ДБН В.2.5-28:2018, Додаток Д [13]).

Блокування відблисків досягається перпендикулярним розташуванням робочого столу до лінії світловидіння вікна та використанням тканинних рулонних штор із коефіцієнтом пропускання < 10 %.

Мікроклімат та акустика:

- приміщення для роботи з персональними комп’ютерами мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією,
- у приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температура повітря повинна становити 22–25°С,
- відносна вологість повітря — 40–60%,
- швидкість руху повітря — не більше 0,1 м/с.

При недотриманні вказаних показників мікроклімату в офісних приміщеннях робочий день для робітників повинен бути скорочений мінімум на 10%.-в офісних приміщеннях нормуються також еквівалентні рівні звуку (для програмістів – 50 дБА, а для операторів в залах обробки інформації на ПК та операторів комп’ютерного набору – 65 дБА), [12], але Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин були скасовані розпорядженням Кабінета Міністрів України від 08.04.2025р. №317-р.

Вимоги щодо рівня неіонізуючих електромагнітних випромінювань, електростатичних і магнітних полів, а також інтенсивність потоків інфрачервоного

та ультрафіолетового випромінювань встановлюються відповідно до ДСанПіН 3.3.2.007-98 [6] і ДСанПіН 3.3.6.096-2002

Електробезпека та організація кабель-менеджменту

- Живлення ПЕОМ – через автоматичний вимикач *С-2 А* із вбудованим УЗО 30 мА [11].
- Кабелі прокладаються у пластикових кабель-каналах із негорючого полімеру, радіус згину ≥8 діаметрів, що мінімізує напруження жил.

Організаційні заходи

1. Система «sit-stand»: не менше 20 % робочого часу – у вертикальному положенні; перехід здійснюється автоматизованим таймером-нагадувачем.
2. Метод 20-5-20 (адаптація ДСанПіН) [6]: кожні 20 хв – 5 с фокус на об’єкт, віддалений на 20 м, для відновлення акомодації.
3. Паспорт робочого місця (ст. 13 ЗУ № 2694-XII) [5]: вноситься розділ «Ергономічні параметри» з відміткою про калібрування крісел і столів.

Таблиця 4.2.2 – Показники ефективності (KPI) впровадження

Показник	Базове значення	Цільове через 12 міс.
Частка скарг на МСР, % від персоналу	37 %	≤20 %
Відсоток робочих місць з ПКЕ ≥ 1,5 %	54 %	≥85 %
Коефіцієнт відповідності меблів ISO 9241-5, [7]	0,62	≥0,9

Також у приміщеннях, де здійснюється робота з комп’ютерами, щодня має проводитися вологе прибирання з метою недопущення запиленості підлоги та меблів.

Ігнорування санітарних правил і норм роботи з ВДТ може викликати у осіб, які з ними професійно працюють, загальну втому, зорову втому, болі та відчуття піску в очах, відчуття засміченості та свербіння очей, болі в хребті, закам’янілість

та оніміння м’язів шиї та плечового поясу, пошкодження дисків хребта, порушення постави, судоми м’язів ніг, синдром RSI хронічний розтяг зв’язок, синдром тунелю Карпаля, головні болі, поганий сон, депресивні стани тощо .

Зазначені технічні та організаційні вимоги формують системний підхід до проектування ПК-робочого місця, що доповнює загальні положення безпеки життєдіяльності, викладені у п. 4.1.1. Їх реалізація забезпечує демонстративне виконання обов’язків роботодавця, визначених ст. 13 і 17 Закону України «Про охорону праці» [5] та адаптує робочу систему до фізичних, сенсорних і когнітивних можливостей операторів, підвищуючи продуктивність і знижуючи ризики професійних захворювань.

ВИСНОВКИ

У кваліфікаційній роботі здійснено повний цикл досліджень — від аналізу існуючих ATS-рішень до проєктування, реалізації та апробації системи, інтегрованої з LinkedIn через розширення Google Chrome.

Запропонована архітектура базується на мікросервісах, об’єднаних шиною подій Kafka; це дало змогу забезпечити горизонтальне масштабування, відмовостійкість і безпечний мультитенантний режим з ізоляцією даних кожного агентства. Практичним результатом роботи став SaaS-продукт, що дозволяє додати кандидата у базу та закріпити його за вакансією лише у три кліки без переходів на зовнішні сторінки ATS системи, навідміну від інших наявних рішень на ринку ATS для рекрутингу.

У порівнянні з конкурентами час виконання операції скоротився у 4–8 разів, а пілотне впровадження, і наразі має 35 рекрутерів або команд рекрутерів.

Реалізований інтерфейс на React оперує подіями в реальному часі, що підвищує прозорість роботи команди рекрутерів, а модуль аналітики дозволяє відстежувати конверсію етапів і продуктивність окремих рекрутерів.

Питання безпеки й ергономіки розглянуто у контексті вимог національних нормативів: попереджено надмірні статичні та зорові навантаження, запропоновано режими роботи з мікропаузами й активною гімнастикою. Це забезпечує дотримання законодавства та зберігає працездатність операторів-рекрутерів, не впливаючи на основну функціональність системи.

Отримані результати мають наукову новизну (безшовна інтеграція ATS у LinkedIn) і відчутну прикладну цінність. Подальші напрями розвитку — впровадження модуля billing, розширення аналітики за рахунок LLM-моделей для скорингу профілів кандидатів з подальшим розвитком в проект AI-рекрутер для LinkedIn по підписці, що трохи суперечить правилам використання LinkedIn, але є можливість зробити AI-рекрутера на базі даного проекта який буде вдало обходити блокування від LinkedIn.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ключові бізнес-показники LinkedIn [Електронний ресурс] – Режим доступу до ресурсу: https://news.linkedin.com/2023/july/linkedin-business-highlights-from-microsoft-s-fy23-q4-earnings (дата звернення 14.06.2025)

2. On Commission Implementing Regulation (EU) 2024/2690 of 14.10.2024 laying down rules for the application of Directive (EU) 2022/2555 as regards technical and methodological requirements of cybersecurity risk-management measures, p.119 [Електронний ресурс] – Режим доступу до ресурсу: https://www.enisa.europa.eu/sites/default/files/2024-11/Implementation%20guidance%20on%20security%20measures_FOR%20PUBLIC%20CONSULTATION.pdf (дата звернення 17.06.2025)

3. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329 (дата звернення 17.06.2025)

4. Я. І. Бедрій. Безпека життєдіяльності: навч. посібник. – Київ, 2009. – С. 45-70. [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/10349524/page:8/ (дата звернення 14.06.2025)

5. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII (чинна редакція 2024 р.). [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/go/2694-12 (дата звернення 14.06.2025)

6. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми при роботі з відеодисплейними терміналами персональних ЕОМ». [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/go/v0007282-98 (дата звернення 14.06.2025)

7. ДСТУ ISO 9241-5:2001 «Вимоги до безпечного і здорового робочого місця з відеодисплейними терміналами». [Електронний ресурс] – Режим доступу до ресурсу:

https://cdn.standards.iteh.ai/samples/16877/c63fa5bf2d9c4b05a08e9432912c8751/ISO-9241-5-1998.pdf (дата звернення 14.06.2025)

8. МОЗ України. Методичні рекомендації щодо організації робочого місця користувача ПЕОМ (01.12.2006 р.). [Електронний ресурс] – Режим доступу до ресурсу: https://zpto.nmcdon.org.ua/images/files/metodrekomendazii/metod_recomendacii.pdf

9. ДСТУ ISO 26800:2019 «Ергономіка. Загальні підходи та принципи проєктування». [Електронний ресурс] – Режим доступу до ресурсу: https://cdn.standards.iteh.ai/samples/42885/27d65cdc5d7a437da1680589fa7ba671/ISO-26800-2011.pdf (дата звернення 14.06.2025)

10. УДК 163.5 Вінницький національний технічний університет, Організація робочого місця користувача ПК, А.І. Деркач С.В. Королевська, [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/itpf/2016/paper/download/1447/1141 (дата звернення 14.06.2025)

11. НПАОП 40.1-1.32-01. Правила будови електроустановок. Розд. 1.7. Захист від ураження електричним аструмом. – Чинна редакція (із доп.). [Електронний ресурс] – Режим доступу до ресурсу: https://misksvitlo.if.ua/wp-content/uploads/2015/09/Правила-улаштування-електроустановок.pdf

12. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text

13. ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ ДБН В.2.5-28:2018, Таблиця Д.1 – Нормовані показники освітлення основних приміщень цивільних будівель, стр 74 [Електронний ресурс] – Режим доступу до ресурсу: https://e-construction.gov.ua/laws_detail/3074958732556240833?doc_type=2

14. Event-Driven Microservices with Apache Kafka: Making Services Resilient to Failure // Confluent Blog. – [Електронний ресурс] – Режим доступу до ресурсу: https://www.confluent.io/blog/event-driven-microservices-apache-kafka/ (дата звернення 17.06.2025)

ДОДАТКИ

Додаток А. Лістинг коду

```
import React, { ReactNode, useContext, useEffect, useState } from 'react'

import { DragStart, DragUpdate, DraggableLocation, DropResult } from '@hello-pangea/dnd'

import { v4 } from 'uuid'

import { ColumnType, Stage, TaskType } from '../ReactDnD/assets'


type DragDropProps = (source: DraggableLocation, destination: DraggableLocation) => void


// handle the manipulation of placeholder for row

// eslint-disable-next-line

type RowDropshadowProps = (event: any, destinationIndex: number, sourceIndex: number) => void


// handle the manipulation of placeholder for column

// eslint-disable-next-line

type ColumnDropshadowProps = (event: any, destinationIndex: number, sourceIndex: number) => void


type RowDropshadow = { margintop: number; height: number }

type ColDropshadow = { marginLeft: number; height: number }


type DragDropContextProps = {

  onSubmit: (

    id: string,

    candidateId: string,

    fullname: string,

    position: string,

    salaryExp: number,

    location: string,

    notice: string,

    image: string,

    colIndex: number

  ) => void

  handleNewColumn: (newName: string) => void

  handleRemoveTask: (rowIndex: string, colIndex: number) => void

  handleDeleteColumn: (colIndex: number) => void
```

```
  handleDragEnd: (result: DropResult) => void
  // eslint-disable-next-line
  handleDragStart: (event: any) => void
  // eslint-disable-next-line
  handleDragUpdate: (event: any) => void
  rowDropshadowProps: RowDropshadow
  colDropshadowProps: ColDropshadow
  columns: ColumnType[]
  setColumns: React.Dispatch<React.SetStateAction<ColumnType[]>>
  handleAddNewItem: () => void
}

const DragDropContext = React.createContext<DragDropContextProps | undefined>(undefined)

// grabbing element currently being dragged from the dom
const getDraggedElement = (draggableId: string) => {
  const queryAttr = 'data-rfd-drag-handle-draggable-id'
  const domQuery = `[${queryAttr}='${draggableId}']`
  const draggedElement = document.querySelector(domQuery)
  return draggedElement
}

// updating the array of the placeholder by switching out the source and destination colIndex
const getUpdatedChildrenArray = (draggedElement: Element, destinationIndex: number, sourceIndex: number) => {
  const child: Element[] = [...draggedElement!.parentNode!.children]

  // if the indexes are the same (onDragStart) just return the dom array
  if (destinationIndex === sourceIndex) return child

  const draggedItem = child[sourceIndex]
  child.splice(sourceIndex, 1)

  return child.splice(0, destinationIndex, draggedItem)
}

// isolate the number of style desired to pass as props
```

```
const getStyle = (
  updatedChildrenArray: Element[],
  destinationIndex: number,
  property: string,
  clientDirection: 'clientHeight' | 'clientWidth'
) =>
  updatedChildrenArray.slice(0, destinationIndex).reduce((total, curr) => {
    const style = window.getComputedStyle(curr)

    const getProp = style.getPropertyValue(property)
    const match = parseFloat(getProp.slice(0, -2))

    const prop = match
    return total + curr[clientDirection] + prop
  }, 0)

interface DragDropProviderProps {
  children: ReactNode
  data: ColumnType[]
  onRemove?: (value: string) => void
}

const DragDropProvider: React.FC<DragDropProviderProps> = ({ children, data,
onRemove }) => {
  const [columns, setColumns] = useState<ColumnType[]>(data)
  const [colDropshadowProps, setColDropshadowProps] = useState<ColDropshadow>({
    marginLeft: 0,
    height: 0
  })
  const [rowDropshadowProps, setRowDropshadowProps] = useState<RowDropshadow>({
    margintop: 0,
    height: 0
  })

  useEffect(() => {
    if (data) {
      setColumns(data)
    }
```

```
  }, [data])

  // handling movement of row in the same column
  // [[],[]],[]
  const moveRowSameColumn: DragDropProps = (source, destination) => {
    // moving tasks in same column
    setColumns((prev) => {
      const updated = [...prev]
      const column = updated.find(({ id }) => id === source.droppableId)
      if (!column) return prev

      const [removed] = column.items.splice(source.index, 1)
      column.items.splice(destination.index, 0, removed)
      return updated
    })
  }

  // handling movement of row between columns
  const moveRowDifferentColumn: DragDropProps = (source, destination) => {
    // moving tasks between columns
    setColumns((prev) => {
      // filter out which column is the source and which is the destination
      const updated = [...prev]
      const sourceColumn = updated.find(({ id }) => id === source.droppableId)
      const destinationColumn = updated.find(({ id }) => id === destination.droppableId)
      if (!sourceColumn || !destinationColumn) return prev

      // extract the tasks from the column
      const [removed] = sourceColumn.items.splice(source.index, 1)
      // insert the source item at the new colIndex
      destinationColumn.items.splice(destination.index, 0, removed)

      return updated
    })
  }

  const handleRowMove: DragDropProps = (source, destination) => {
```

```
    if (source.droppableId !== destination.droppableId) {
      moveRowDifferentColumn(source, destination)
    } else {
      moveRowSameColumn(source, destination)
    }
  }

  // move columns
  const handleColumnMove: DragDropProps = (source, destination) =>
    setColumns((prev) => {
      const updated = [...prev]
      const [removed] = updated.splice(source.index, 1)
      updated.splice(destination.index, 0, removed)
      return updated
    })

  const handleDropshadowRow: RowDropshadowProps = (event, destinationIndex, sourceIndex) => {
    const draggedElement = getDraggedElement(event.draggableId)
    if (!draggedElement) return
    const { clientHeight } = draggedElement as Element

    const updatedChildrenArray: Element[] = getUpdatedChildrenArray(
      draggedElement as Element,
      destinationIndex,
      sourceIndex
    )

    const margintop = getStyle(updatedChildrenArray, destinationIndex, 'margin-bottom', 'clientHeight')

    setRowDropshadowProps({
      height: clientHeight + 2,
      margintop: margintop + 2 * destinationIndex
    })
  }

  const handleDropshadowColumn: ColumnDropshadowProps = (event, destinationIndex, sourceIndex) => {
```

```
    const draggedElement: Element | Node | null = getDraggedElement(event.draggableId)!.parentNode!.parentNode

    if (!draggedElement) return
    const { clientHeight } = draggedElement as Element

    const updatedChildrenArray: Element[] = getUpdatedChildrenArray(
      draggedElement as Element,
      destinationIndex,
      sourceIndex
    )

    const marginLeft = getStyle(updatedChildrenArray, destinationIndex, 'margin-right', 'clientWidth')

    setColDropshadowProps({
      height: clientHeight,
      marginLeft
    })
  }

  const handleDragUpdate = (event: DragUpdate) => {
    const { source, destination } = event
    if (!destination) return
    if (event.type === 'column') {
      handleDropshadowColumn(event, destination.index, source.index)
    } else {
      handleDropshadowRow(event, destination.index, source.index)
    }
  }

  const handleDragStart = (event: DragStart) => {
    const { index } = event.source
    if (event.type === 'column') {
      handleDropshadowColumn(event, index, index)
    } else {
      handleDropshadowRow(event, index, index)
    }
```

```
}

const handleDragEnd = (result: DropResult) => {
  if (!result.destination) return

  const { source, destination } = result

  if (source.droppableId === 'all-columns') {
    handleColumnMove(source, destination)
  } else {
    handleRowMove(source, destination)
  }
}

const handleDeleteColumn = (colIndex: number) =>
  setColumns((prev) => {
    const updated = [...prev]
    return updated.filter((_col, index) => index !== colIndex)
  })

const onSubmit = (
  id: string,
  candidateId: string,
  fullname: string,
  position: string,
  salaryExp: number,
  location: string,
  notice: string,
  image: string,
  colIndex: number
) => {
  setColumns((prev) => {
    const updated = [...prev]
    const newItem: TaskType = {
      id,
      candidateId,
      fullname,
      position,
```

```
        salaryExp,
        location,
        notice,
        image
      }
      if (updated[colIndex].items[0] && 'available' in updated[colIndex].items[0])
{
        updated[colIndex].items = [...updated[colIndex].items, newItem] as Stage[]
      } else {
        updated[colIndex].items   =   [...updated[colIndex].items,   newItem]   as
TaskType[]
      }
      return updated
    })
  }

  const handleAddNewItem = () => {
    const newColumns = [...columns]
    const newTaskId = Math.random().toString(36).substr(2, 9)

    const newItem: Stage = {
      id: newTaskId,
      title: '',
      available: true
    }

    newColumns[0].items.unshift(newItem)

    setColumns(newColumns)
  }

  const handleRemoveTask = (rowIndex: string, colIndex: number) => {
    setColumns((prev) => {
      const updated = [...prev]
      updated[colIndex].items  =  updated[colIndex].items.filter((item)  =>  item.id
!== rowIndex)
      return updated
    })
```

```
  if (onRemove)
    onRemove(rowIndex)
}

const handleNewColumn = (newName: string) => {
  setColumns((prev) => {
    const updated = [...prev]
    return [
      ...updated,
      {
        id: v4(),
        title: newName,
        items: [],
        suggestions: []
      }
    ]
  })
}

return (
  <DragDropContext.Provider
    value={{
      onSubmit,
      handleNewColumn,
      handleRemoveTask,
      handleDeleteColumn,
      handleDragEnd,
      handleDragStart,
      handleDragUpdate,
      rowDropshadowProps,
      colDropshadowProps,
      columns,
      setColumns,
      handleAddNewItem
    }}
  >
    {children}
  </DragDropContext.Provider>
```

```
  )
}

// eslint-disable-next-line
export function useDragDrop() {
  const context = useContext(DragDropContext)
  if (context === undefined) {
    throw new Error('useDragDrop must be used inside DragDropProvider')
  }

  return context
}

export default DragDropProvider


////////////////////////////////////////////////////


import React, { useCallback, useEffect, useRef, useState } from 'react'
import { DragDropContext, DropResult, Droppable } from '@hello-pangea/dnd'
import Column from '../Column/Column'
import { useDragDrop } from '../DragDropProvider'
import './Board.scss'
import { useSearchContext } from '../../../store/SearchCandidatesDndContext'
import { useAppDispatch } from '../../../store/hooks/hooks'
import { VacancyCandidateResponseWithInfo } from '../../../store/reducers/vacancy/candidate/candidates.state'
import { useVacancyCandidateService } from '../../../services/global/useGlobalServices'
import { updateVacancyCandidateStage } from '../../../store/reducers/vacancy/root/vacancies.thunk'
import { VacancyStageResponseWithInfo } from '../../../store/reducers/vacancy/stage/stages.state'


type BoardProps = {
  vacancyId?: string,
  candidates?: VacancyCandidateResponseWithInfo[],
```

```
  stages?: VacancyStageResponseWithInfo[]
}

const Board: React.FC<BoardProps> = ({ vacancyId, candidates, stages }) => {
  const dispatch = useAppDispatch()
  //const service = useVacancyService()
  const vacancyCandidateService = useVacancyCandidateService()
  const { handleDragEnd, handleDragStart, handleDragUpdate, columns } =
useDragDrop()
  const widthRef = useRef<HTMLDivElement>(null)
  const [boardWidth, setBoardWidth] = useState(0)
  const { searchTerm } = useSearchContext()

  const calculateBoardWidth = useCallback(() => {
    if (widthRef.current && columns.length > 0) {
      const columnWidth = 200
      const gap = 20;
      const totalWidth = columns.length * (columnWidth + gap) - gap
      setBoardWidth(totalWidth)
    }
  }, [columns.length])

  useEffect(() => {
    calculateBoardWidth()
  }, [calculateBoardWidth, columns.length])

  const localeIncludes = (str: string, searchTerm: string) => {
    const searchTermLength = searchTerm.length
    const strLength = str.length

    for (let i = 0; i <= strLength - searchTermLength; i++) {
      const substring = str.substring(i, i + searchTermLength)
      const comparisonResult = substring.localeCompare(searchTerm, undefined, {
sensitivity: 'base' })

      if (comparisonResult === 0) {
        return true
      }
```

```
    }

    return false
  }

  const handleDragEndWithAPI = (result: DropResult) => {
    const { source, destination } = result;
    if (!destination) return;

    handleDragEnd(result);

    if (source.droppableId !== destination.droppableId) {
      const taskId = result.draggableId;
      const candidateId = candidates?.find(candidate => candidate.id === taskId)?.candidateId
      const activeStageName = stages?.find(stage => stage.id === destination.droppableId)?.stageName

      dispatch(updateVacancyCandidateStage({
        vacancyId: vacancyId!,
        candidateId: candidateId!,
        candidateDetails: {
          //id: taskId,
          //stageId: destination.droppableId,
          notes: '',
          //rejectReasonId?: undefined,
          activeStageName: activeStageName!,
          isTerminalStage: true
        },
        service: vacancyCandidateService
      }))
    }
  }

  return (
    <DragDropContext onDragEnd={handleDragEndWithAPI} onDragStart={handleDragStart} onDragUpdate={handleDragUpdate}>
      <Droppable droppableId="all-columns" direction="horizontal" type="column">
```

```
        {(provided) => (
          <div className="board-dnd-container"
            ref={widthRef}
            style={{width: `${boardWidth}px`}}
          >
            <div className="board-dnd" id="task-board" {...provided.droppableProps}
ref={provided.innerRef}>
              {columns.map((column, columnIndex) => {
                const filteredTasks = column.items.filter((task) => {
                  if ('location' in task) {
                    return (
                      localeIncludes(task.fullname, searchTerm) ||
                      localeIncludes(task.position, searchTerm) ||
                      localeIncludes(task.location, searchTerm)
                    )
                  }
                })
                return <Column key={column.id} column={{ ...column, items:
filteredTasks }} columnIndex={columnIndex} />
              })}
              {provided.placeholder}
            </div>
          </div>
        )}
      </Droppable>
    </DragDropContext>
  )
}

export default Board

////////////////////////////////////////////////

export * from './Board'
```

```
////////////////////////////////////////////////

@import "../../../helpers/styles/variables";

.board-dnd-container {
    overflow-x: auto;
    width: auto;
    min-height: 500px;
}

.board-dnd {
    display: grid;
    grid-gap: 20px;
    grid-auto-flow: column;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    grid-auto-columns: minmax(200px, 1fr);

    min-width: max-content;
    height: 100%;
    padding: 10px 20px;
}



////////////////////////////////////////////////



import { Droppable, DroppableProvided, DroppableStateSnapshot } from '@hello-pangea/dnd'
import { ColumnType, TaskType } from '../assets/api'
import { Row } from '../Row'
import './Column.scss'
import React from 'react'
import { useDragDrop } from '../DragDropProvider'

type Props = {
  column: ColumnType
  columnIndex: number
}
```

```
const Column: React.FC<Props> = ({ column }) => {
  const taskCount = column.items.length
  const { rowDropshadowProps } = useDragDrop()

  return (
    // logic for dragging columns
    //
    // <Draggable draggableId={column.id} index={columnIndex}>
    //    {(provided: DraggableProvided) => (

    //    )}
    // </Draggable>

    <div className="column-container">
      <div className="column-title-container">
        <div className="column-title">
          {column.title}
          <p className="task-count"> - {taskCount}</p>
        </div>
      </div>

      <Droppable droppableId={column.id} type="task">
        {(prov: DroppableProvided, snapshot: DroppableStateSnapshot) => (
          <div         className="column-row-container"         ref={prov.innerRef}
{...prov.droppableProps}>
            {column.items.map((task, taskIndex) => (
              <Row key={task.id} task={task as TaskType} index={taskIndex} />
            ))}
            {snapshot.isDraggingOver && (
              <div
                className="column-dropshadow"
                style={{
                  marginTop: rowDropshadowProps.margintop,
                  height: rowDropshadowProps.height,
                  backgroundColor: '#ffbe85'
                }}
              />
```

```
            )}
            {prov.placeholder}
          </div>
        )}
      </Droppable>
    </div>
  )
}

export default Column


////////////////////////////////////////


export * from './Column'


////////////////////////////////////////


.column-container {
    display: flex;
    flex-direction: column;
    width: 200px;
    min-height: 100%;
    flex-shrink: 0;

    .column-title-container {
        display: flex;
        justify-content: space-between;
        align-items: center;
        width: 100%;
        border: none;
        background-color: transparent;
        margin-bottom: 10px;
        justify-content: center;
```

```
    .column-title {
        display: flex;
        justify-content: flex-start;
        align-items: center;
        width: auto;
        color: #333;
        background-color: transparent;
        font-weight: bold;
        font-size: 16px;
        padding: 0;
        cursor: default;

        .task-count {
            padding-left: 5px;
            font-weight: 400;
        }
    }
}

.column-row-container {
    position: relative;
    display: flex;
    align-items: stretch;
    justify-content: flex-start;
    flex-direction: column;
    width: 100%;
    height: 100%;
    flex-grow: 1;

    .column-row {
        width: 100%;
        height: 50px;
        margin-bottom: 10px;
    }

    .column-dropshadow {
        border-radius: 4px;
        position: absolute;
```

```
            top: 0;
            left: 0;
            right: 0;
            background-color: gray;
            opacity: 0.5;
            z-index: -1;
        }
    }
}

//////////////////////////////////////////

import React from 'react'
import { Draggable, DraggableProvided } from '@hello-pangea/dnd'
import { TaskType } from '../assets'
import './Row.scss'

import { useDescriptionMenu } from '../../../store/DescriptionMenuContext'

type Props = {
   task: TaskType
   index: number
}

const Row: React.FC<Props> = ({ task, index }) => {
   const { openDescriptionMenu } = useDescriptionMenu();

   const handleClick = () => {
      openDescriptionMenu(task);
   };

   return (
      <Draggable draggableId={task.id} index={index}>
         {(provided: DraggableProvided) => (
            <>
               <div
```

```
                    className='card-dnd-container'
                    {...provided.draggableProps}
                    {...provided.dragHandleProps}
                    ref={provided.innerRef}
                    onClick={handleClick}
                >
                    <div className='card-image'>
                        <span
                            className="icon is-large is-left"
                        >
                            <i className="fas fa-lg fa-smile has-text-black"></i>
                        </span>
                    </div>

                    <div>
                        <div className='card-body'>
                            <div className='card-fullname'>{task.fullname}</div>
                            <div    className='card-position'>{task.position},    <div
className='card-salary'>{task.salaryExp}$</div></div>
                            <div className='card-location'>{task.location}</div>
                            <div className='card-notice'>Notice: {task.notice}</div>
                        </div>

                    </div>
                </div>
            </>
        )}
    </Draggable>
  )
}

export default Row;


//////////////////////////////////////


export { default as Row } from './Row'
```

```
////////////////////////////////////


.card-dnd-container {
    height: 70px;
    display: flex;
    align-items: center;
    border-radius: 4px;
    margin-bottom: 20px;
    border: 1px solid;
    padding: 1px 5px 3px 0px;

    .card-img {
        align-self: flex-start;
        padding: 20px;
        margin-right: 10px;
    }

    .card-body {
        line-height: 1.2;
        display: flex;
        flex-direction: column;
        font-size: 10px;

        .card-notice {
            font-weight: bold;
        }

        .card-position,
        .card-salary {
            display: inline-block;
            padding-bottom: 2px;
        }

        .card-fullname {
            font-size: 12px;
```

```
            font-weight: bold;
        }

        .card-salary {
            font-size: 12px;
        }
    }

    .card-image {
        align-self: flex-start;
        padding-left: 1px;
        padding-right: 1px;
    }
}



////////////////////////////////////


export type TaskType = {
    id: string
    candidateId: string
    fullname: string
    position: string
    salaryExp: number
    location: string
    notice: string
    image: string
    vacancyDescription?: string | null
    clientDescription?: string | null
}

export type Stage = {
    id: string
    title: string
    available: boolean
}
```

```
export type SuggestionType = {
    id: string
    name: string
}

export type ColumnType = {
    id: string
    title: string
    items: (TaskType | Stage)[]
    suggestions?: SuggestionType[]
}

export type TaskBoardType = {
    columns: ColumnType[]
}


export type ItemColumnType = {
    columnStages: ColumnType[]
    columnRejection: ColumnType[]
}

export const apiForColumn: ItemColumnType = {
    columnStages: [
        {
            id: '1',
            title: '',
            items: [
                {
                    id: 'stage-001',
                    title: 'Invitation Sent',
                    available: true
                },
                {
                    id: 'stage-002',
                    title: 'Invitation Accepted',
                    available: true
```

```
},
{
    id: 'stage-003',
    title: 'Vacancy Discussion',
    available: true
},
{
    id: 'stage-005',
    title: 'CV Received',
    available: true
},
// {
//     id: 'stage-004',
//     title: 'Expected CV',
//     available: true
// },
{
    id: 'stage-006',
    title: 'Recruiter Interview',
    available: true
},
{
    id: 'stage-007',
    title: 'Technical Interview',
    available: true
},
{
    id: 'stage-008',
    title: 'Job Offer Sent',
    available: true
},
{
    id: 'stage-009',
    title: 'Job Offer Accepted',
    available: true
},
{
    id: 'stage-010',
```

```
                title: 'Rejected',
                available: true
            },
            // {
            //     id: 'stage-010',
            //     title: 'Probation Started',
            //     available: true
            // },
            // {
            //     id: 'stage-011',
            //     title: 'Probation Finished Successfully',
            //     available: true
            // }
        ],
        suggestions: [
            {id: '', name: 'Invitation Sent' },
            {id: '', name: 'Invitation Accepted' },
            {id: '', name: 'Vacancy Discussion' },
            {id: '', name: 'Expected CV' },
            {id: '', name: 'CV Received' },
            {id: '', name: 'Recruiter Interview' },
            {id: '', name: 'Technical Interview' },
            {id: '', name: 'Manager Interview' },
            {id: '', name: 'Tech. Task Sent' },
            {id: '', name: 'Tech. Task Completed' },

            {id: '', name: 'Job Offer Sent' },
            {id: '', name: 'Job Offer Accepted' },

            {id: '', name: 'Probation Started' },
            {id: '', name: 'Probation Finished Successfully' }
        ]
    }
],
columnRejection: [
    {
        id: '1',
        title: '',
```

```
            items: [
                { id: 'rejection-002', title: 'Did Not Start on First Day', available: true },
                { id: 'rejection-003', title: 'Not Ready to Change Current Position', available: true },
                { id: 'rejection-004', title: 'Not Looking for a New Job Now', available: true },
                { id: 'rejection-005', title: 'Not Interested in Project', available: true },
                { id: 'rejection-006', title: 'Not Ready to Work with Company', available: true },
                { id: 'rejection-007', title: 'Not Ready to Work in This Domain', available: true },
                { id: 'rejection-008', title: 'Not Ready to Relocate', available: true },
                { id: 'rejection-009', title: 'Rejected by CV', available: true },
                { id: 'rejection-010', title: 'Rejected After Technical Interview', available: true },
                { id: 'rejection-011', title: 'Rejected After HR Interview', available: true },
                { id: 'rejection-012', title: 'Rejected After Manager Interview', available: true },
                { id: 'rejection-013', title: 'Rejected After Technical Test', available: true },
                {
                    id: 'rejection-014',
                    title: 'Rejected After Pre-Screen Call by Recruiter from Agency',
                    available: true
                },
                { id: 'rejection-015', title: 'Job Offer Was Rejected by Candidate', available: true },
                { id: 'rejection-016', title: 'Salary Not Enough', available: true },
                { id: 'rejection-017', title: 'Vacancy on Hold', available: true },
                { id: 'rejection-018', title: 'Will Be Ready to Talk About Vacancy in X Months', available: true },
                {
                    id: 'rejection-019',
                    title: 'The Candidate Was Not Considered/The Vacancy Was Closed by Another Candidate, and This Candidate Was Not in the Recruitment Process',
                    available: true
                },
                { id: 'rejection-020', title: 'Candidate Stopped Responding (Ghost Candidate)', available: true },
```

```
                { id: 'rejection-021', title: 'No Response from Company About Candidate', available: true },
                { id: 'rejection-022', title: 'Blacklist', available: true }
            ],
            suggestions: [
                {id: '', name: 'Accepted Counter Offer :(' },
                {id: '', name: 'Did Not Start on First Day' },
                {id: '', name: 'Not Ready to Change Current Position' },
                {id: '', name: 'Not Looking for a New Job Now' },
                {id: '', name: 'Not Interested in Project' },
                {id: '', name: 'Not Ready to Work with Company' },
                {id: '', name: 'Not Ready to Work in This Domain' },
                {id: '', name: 'Not Ready to Relocate' },
                {id: '', name: 'Rejected by CV' },
                {id: '', name: 'Rejected After Technical Interview' },
                {id: '', name: 'Rejected After HR Interview' },
                {id: '', name: 'Rejected After Manager Interview' },
                {id: '', name: 'Rejected After Technical Test' },
                {id: '', name: 'Rejected After Pre-Screen Call by Recruiter from Agency' },
                {id: '', name: 'Job Offer Was Rejected by Candidate' },
                {id: '', name: 'Salary Not Enough' },
                {id: '', name: 'Vacancy on Hold' },
                {id: '', name: 'Will Be Ready to Talk About Vacancy in X Months' },
                {id: '',
                    name: 'The Candidate Was Not Considered/The Vacancy Was Closed by Another Candidate, and This Candidate Was Not in the Recruitment Process'
                },
                {id: '', name: 'Candidate Stopped Responding (Ghost Candidate)' },
                {id: '', name: 'No Response from Company About Candidate' },
                {id: '', name: 'Blacklist' }
            ]
        }
    ]
}
//////////////////////////////////////
export * from './api'

//////////////////////////////////////
```

Додаток Б. Диск з кваліфікаційною роботою бакалавра