НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Вебзастосунок для покращення ефективності резюме

Виконав студент IV курсу, групи ІП-12
(шифр групи)
Авчаров Григорій Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Фіногенов О. Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант ст.викл, д-р філософії, Головченко М.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент к.т.н., доц. КІОНС ПБФ, Мураховський. С.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Авчарову Григорію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Вебзастосунок для покращення ефективності резюме
керівник проєкту Фіногенов Олексій Дмитрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Передпроєктне обстеження предметної області: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі, моделювання бізнес-процесів.
2) Розроблення вимог до програмного забезпечення: розроблення варіантів використання застосунку, розроблення функціональних та нефункціональних вимог, аналіз економічних показників.
3) Конструювання та розроблення програмного забезпечення: архітектура програмного забезпечення, архітектурні рішення та обґрунтування вибору засобів розробки, конструювання програмного забезпечення, аналіз безпеки даних.
4) Аналіз якості та тестування програмного забезпечення: керівництво користувача, методика випробувань програмного продукту, опис контрольного прикладу.

5) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема бізнес процесу «Модерація резюме адміністратором»

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	20.03.2025	
3	Постановка та формалізація задачі	27.03.2025	
4	Розробка інформаційного забезпечення	31.03.2025	
5	Алгоритмізація задачі	15.04.2025	
6	Обґрунтування вибору використаних технічних засобів	20.04.2025	
7	Розробка програмного забезпечення	19.05.2025	
8	Налагодження програми	24.05.2025	
9	Виконання графічних документів	27.05.2025	
10	Оформлення пояснювальної записки	31.05.2025	
11	Подання ДП на попередній захист	02.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент _____ Григорій АВЧАРОВ
(підпис) (ініціали, прізвище)

Керівник _____ Олексій ФІНОГЕНОВ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 51 таблиць, 46 рисунків та 15 джерел – загалом 81 сторінка.

Дипломний проєкт присвячений розробці веб-застосунку для підвищення ефективності резюме.

Метою проєкту є покращення процесу аналізу резюме.

У розділі «Передпроєктне обстеження предметної області» було проаналізовано предметну область, досліджено існуючі технічні рішення та програмні продукти, а також змодельовано та описано ключові бізнес-процеси, такі як реєстрація користувача, додавання резюме та його модерація.

У розділі «Розроблення вимог до програмного забезпечення» визначено функціональні та нефункціональні вимоги до розробленого продукту, а також зазначено та детально описано варіанти використання програмного забезпечення для різних ролей, таких як звичайний користувач та адміністратор.

У розділі «Конструювання та розроблення програмного забезпечення» присвячений опису архітектури, обґрунтуванню вибору технологічного стеку, конструюванню програмного забезпечення з детальним описом структури бази даних, а також аналізу безпеки даних.

У розділі «Аналіз якості та тестування програмного забезпечення» проведений аналіз якості ПЗ, описані процеси мануального тестування та наведено детальний контрольний приклад, що демонструє ключові сценарії роботи системи.

Розділ «Розгортання та супровід програмного забезпечення» містить опис процесу розгортання застосунку з використанням контейнерів та хмарної платформи Microsoft Azure, а також висвітлює аспекти підтримки та оновлення програмного забезпечення через Docker Hub.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, РЕЗЮМЕ, ANGULAR, JAVA, MINIO, AZURE, DOCKER, БАЗА ДАНИХ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 51 tables, 46 figures and 15 sources – in total 81 pages.

The purpose of the diploma project is development of a web application to improve the effectiveness of resumes.

The aim of the project is to improve the resume analysis process.

In the section “Pre-project survey of the subject area,” the subject area was analyzed, existing technical solutions and software products were researched, and key business processes such as user registration, resume addition, and moderation were modeled and described.

The section “Software Requirements Development” defines the functional and non-functional requirements for the developed product, as well as identifies and describes in detail the options for using the software for different roles, such as a regular user and an administrator.

The section “Software Design and Development” is devoted to describing the architecture, justifying the choice of the technology stack, designing the software with a detailed description of the database structure, and analyzing data security.

The section “Software Quality Analysis and Testing” analyzes the quality of the software, describes the manual testing processes, and provides a detailed control example demonstrating the key scenarios of the system operation.

The section “Software Deployment and Support” contains a description of the application deployment process using containers and the Microsoft Azure cloud platform, as well as highlights aspects of software support and updates via Docker Hub.

KEYWORDS: WEB APP, RESUME, ANGULAR, JAVA, MINIO, AZURE, DOCKER, DATABASE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1201.045440.01.91	Технічне завдання	16	
3	А4	КПІ.ІП-1201.045440.02.81	Пояснювальна записка	81	
4	А4	КПІ.ІП-1201.045440.03.12	Текст програми	28	
5	А4	КПІ.ІП-1201.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-1201.045440.05.34	Керівництво користувача	11	
7	А3	КПІ.ІП-1201.045440.06.99	Графічний матеріал	3	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1201.045440.00.90		
Розроб.		Авчаров Г.О.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Фіногенов О.Д.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-12	
Н.контр.		Головченко М.М.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ РЕЗЮМЕ

Технічне завдання

КПІ.ІП-1201.045440.02.81

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Григорій АВЧАРОВ

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок для покращення ефективності резюме.

Галузь застосування:

Наведене технічне завдання поширюється на розробку “Веб-застосунку для покращення ефективності резюме” [КПІ.ІП-1201.045440.02.81], котра використовується для обміну резюме без персональних даних та призначена для широкого кола шукачів роботи, включаючи студентів, фахівців, що прагнуть змінити місце роботи або покращити своє резюме.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Веб-застосунку для покращення ефективності резюме» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання шукачам роботи інструменту для обміну досвідом та покращення власних резюме шляхом доступу до інших прикладів без персональних даних та отримання зворотного зв'язку від спільноти, полегшення розуміння ринку праці.

Метою розробки є покращення процесу аналізу резюме, за допомогою якого шукачі роботи зможуть підвищити якість та ефективність своїх резюме, шляхом надання доступу до прикладів без персональних даних, створення середовища для обговорення та обміну досвідом.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– сторінка авторизації(рис 4.1);

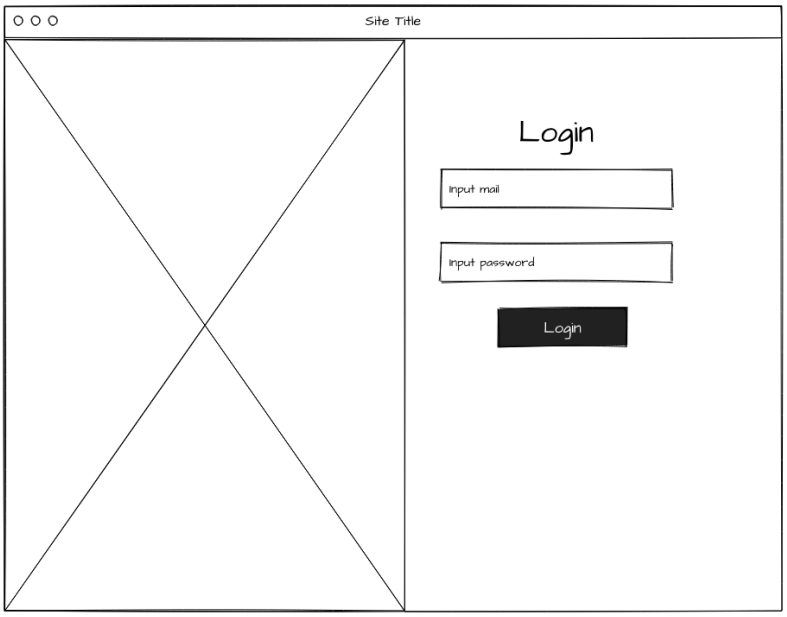


Рисунок 4.1 – Авторизація

– перегляд головної сторінки(рис 4.2);

Рисунок 4.2 – Головна сторінка

– подання резюме на обробку(рис 4.3);

Рисунок 4.3 – Вікно подання резюме

– перегляд резюме користувачів та їх фільтрація(рис 4.4);

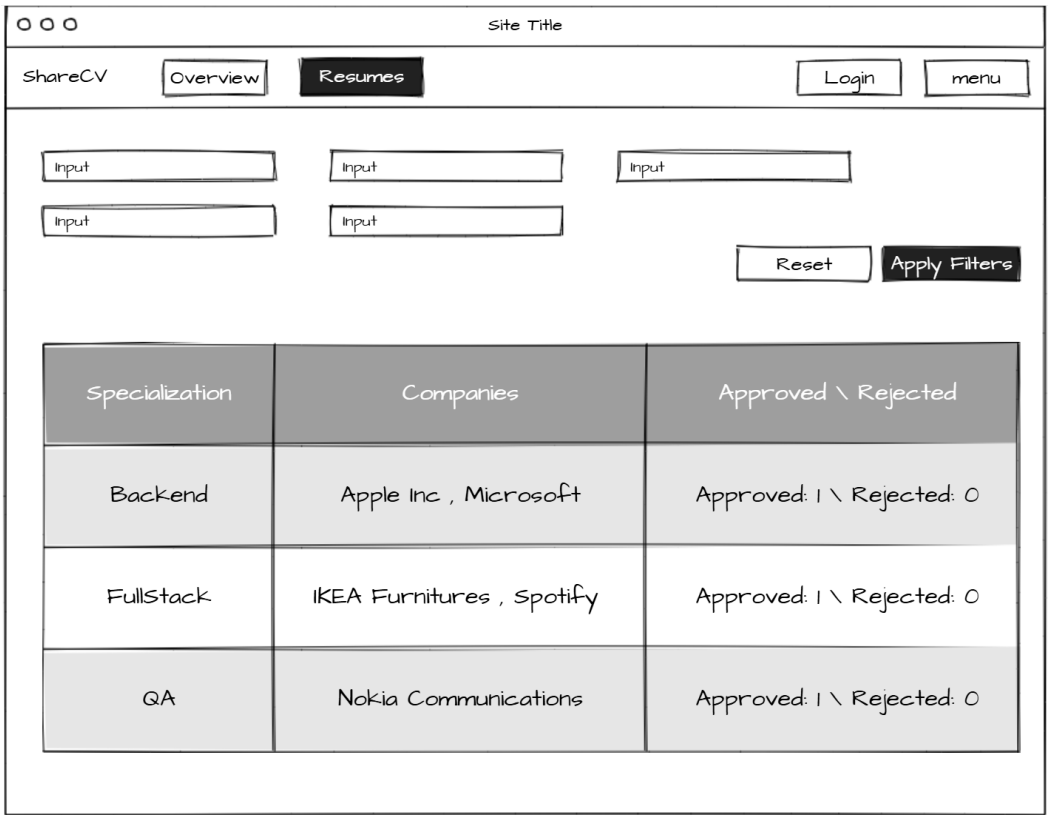


Рисунок 4.4 – Сторінка з резюме користувачів та фільтрацією

– перегляд та коментування резюме користувачів(рис 4.5);

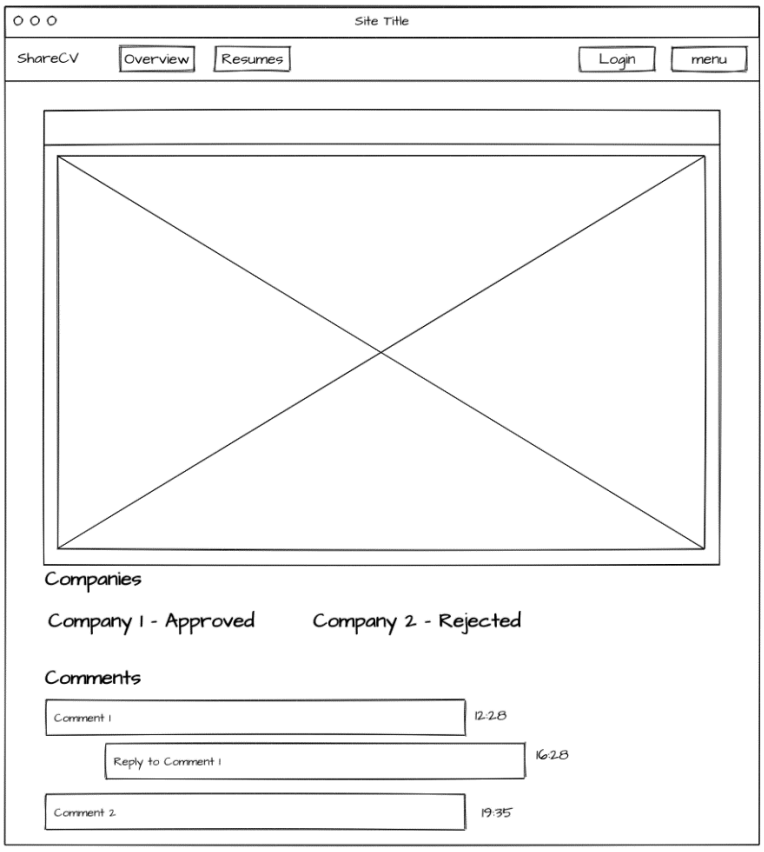


Рисунок 4.5 – Детальний інтерфейс резюме

– перегляд особистих та обраних резюме(рис 4.6);

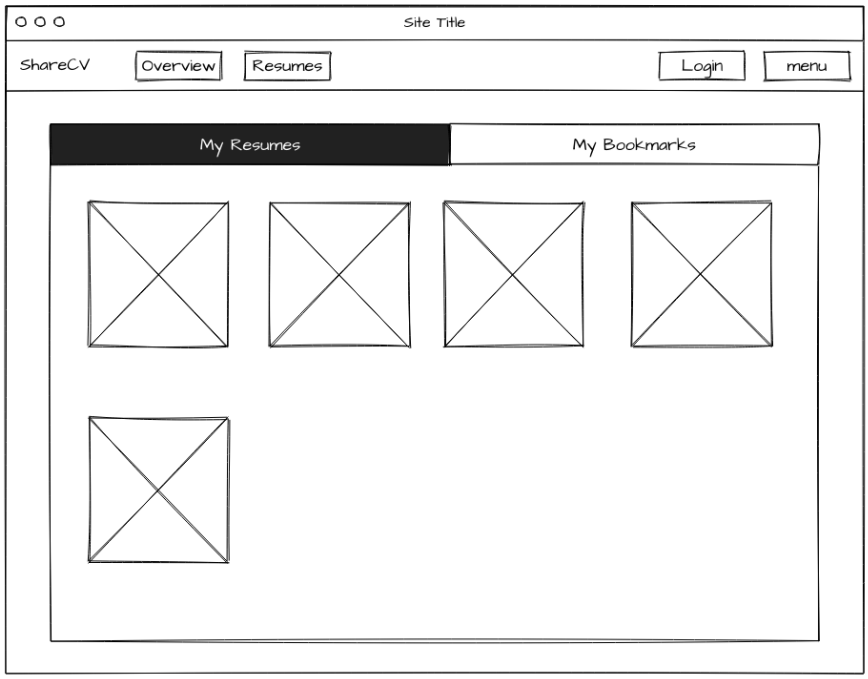


Рисунок 4.6 – Сторінка особистих та обраних резюме

4.1.2 Адміністративного інтерфейсу:

– Перегляд поданих резюме(рис 4.7);

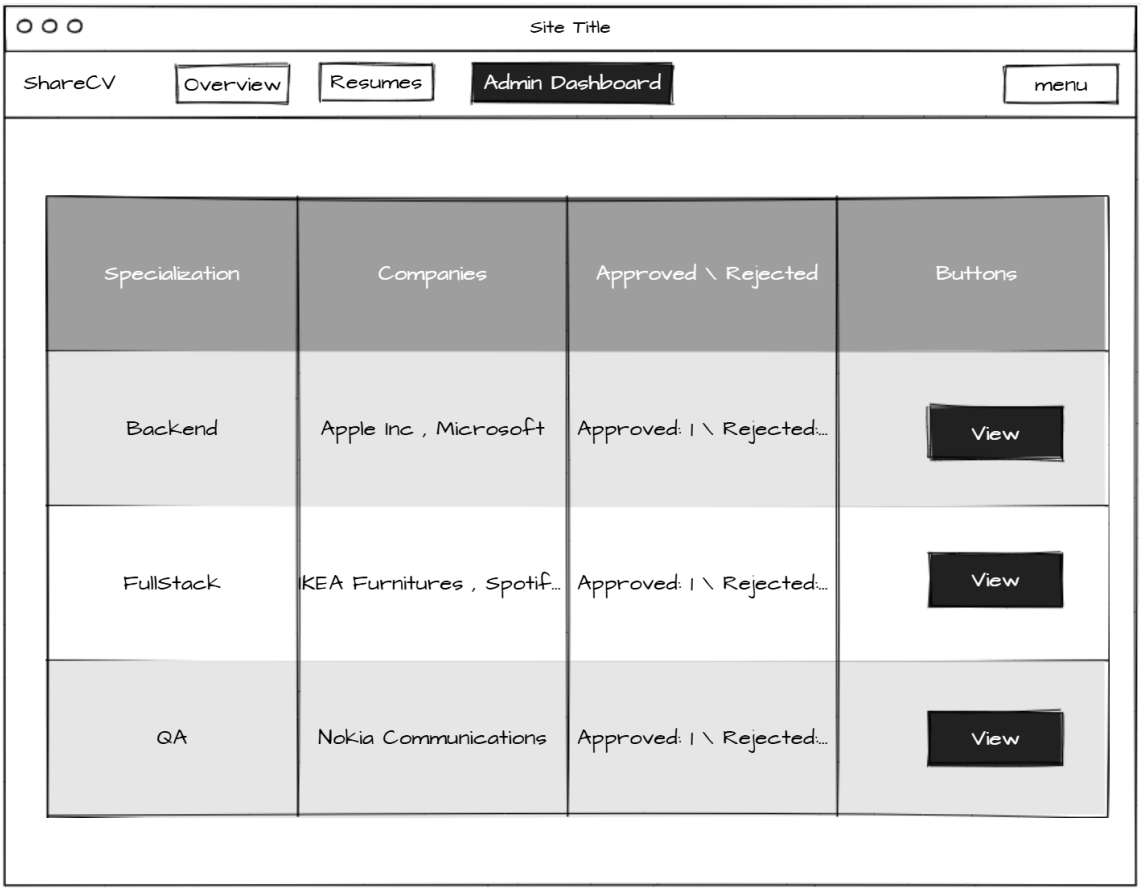


Рисунок 4.7 – Сторінка поданих резюме користувачами

– Перегляд документів резюме та їх схвалення(рис 4.8);

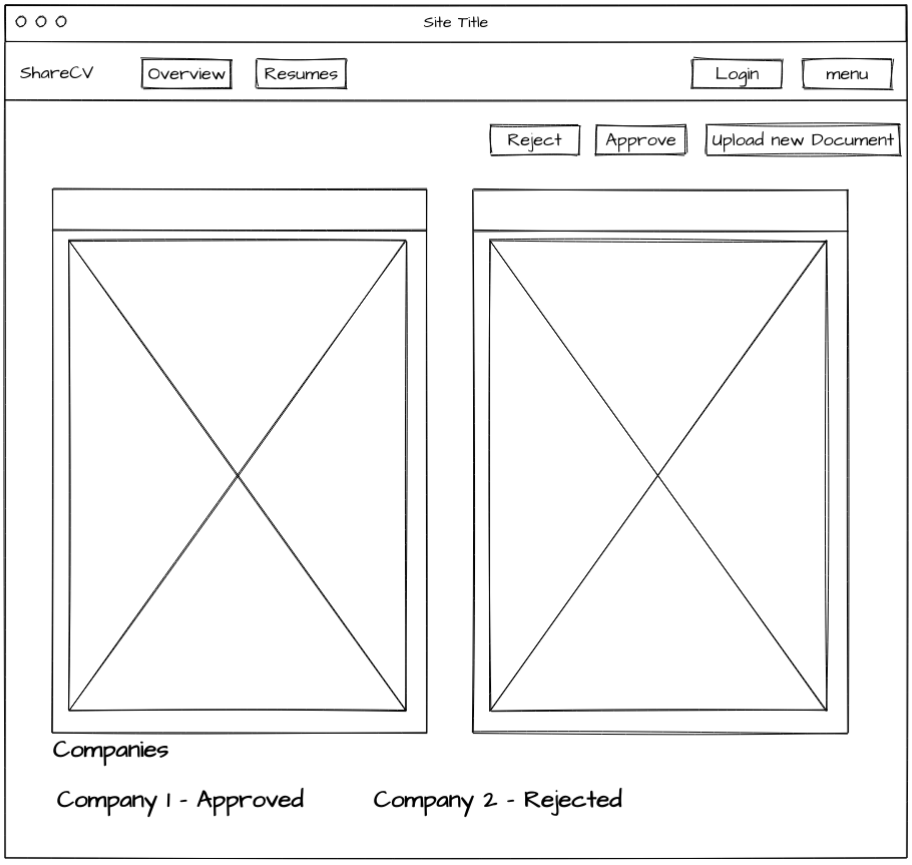


Рисунок 4.8 – Сторінка детального перегляду резюме та його схвалення

4.1.3 Для користувача:

– створення та подача резюме на розгляд адміністрації;
– перегляд списку опублікованих резюме;
– пошук резюме з використанням фільтрів;
– перегляд детальної інформації конкретного опублікованого резюме;
– написання коментаря до резюме;
– написання відповіді на коментар;
– голосування за коментарі (зміна рейтингу коментаря);
– сортування коментарів за критеріями;
– перегляд списку власних відправлених резюме;
– приховування власного резюме з публічного доступу;
– перегляд документу власного відправленого резюме;

– додавання резюме до закладок;
– дерегляд списку резюме в закладках;

4.1.4 Для адміністратора:

– перегляд списку резюме, відправлених на обробку;
– перегляд детальної інформації поданого на обробку резюме;
– зміна статусу поданого резюме;
– заміна/оновлення документа резюме;

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Застосунок повинен зберігати захешовані паролі користувачів у базі даних. При спробі користувача без ролі адміністратора на сторінки, що доступні тільки адміністратору, застосунок повинен обмежити доступ.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;
– Рекомендована конфігурація технічних засобів:
– тип процесору: Ryzen 5 5600H;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Вимоги до вхідних даних не висуваються.

4.5.2 Вимоги до вихідних даних

Вимоги до вихідних даних не висуваються.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Typescript, Java, Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовищ VSCode, IntelliJ Idea, PyCharm.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторіїв в системі контролю версій GitHub, що буде містити весь код програмного забезпечення.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– архітектура програмного забезпечення;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проєкту	15.03	
2.	Розробка технічного завдання	20.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	27.03	Специфікації програмного забезпечення
4.	Проєктування структури програмного забезпечення, проєктування компонентів	31.03	Схема бази даних
5.	Програмна реалізація програмного забезпечення	15.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	19.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проєкту	24.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проєкту	27.05	Графічний матеріал проєкту
9.	Оформлення технічної документації проєкту	31.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Вебзастосунок для покращення ефективності резюме**

КПІ.ІП-1201.045440.02.81

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application programming interface, прикладний програмний Інтерфейс.
CI/CD	–	Continuous Integration / Continuous Delivery – безперервна інтеграція / безперервна доставка.
CLI	–	Command Line Interface – інтерфейс командного рядка.
ER	–	Entity-Relation diagram.
HR	–	Human Resource – управління персоналом / відділ кадрів.
IDE	–	Integrated Development Environment – інтегроване середовище розробки.
IT	–	Інформаційні технології.
JSON	–	JavaScript Object Notation – об'єктна нотація JavaScript.
JWT	–	JSON Web Token – веб-токен JSON.
PDF	–	Portable Document Format – портативний формат документів.
БД	–	База даних.
Скрінінг	–	Попередній аналіз або відбір (наприклад, резюме під час найму)

ВСТУП

В наші часи, коли конкуренція на ринку праці постійно збільшується, зростає потреба в різних навичках, а кількість робочих місць та компенсація не росте з такою ж швидкістю, важливо ефективно просувати свої сильні сторони та досвід, щоб виділитись серед кандидатів на вакантне робоче місце.

Наразі, навіть співробітники з великим стажем, які вже мають роботу, стикаються з труднощами під час зміни робочого місця, після подачі різних версій резюме у різні компанії отримуючи тільки відмови, їм складно зрозуміти у чому проблема, що їх резюме не зацікавлює роботодавців. Друга проблема полягає в тому, що робітники компаній не хочуть, щоб їхні компаній дізналися про те, що вони шукають роботу. Розміщення резюме на публічних форумах призведе до того, що компанія дізнається про плани працівника, а це може вплинути на його професійне життя. Також при пошуку роботи якщо кандидат має на меті конкретну компанію, наразі немає єдиного ресурсу, що містив би приклади успішних резюме, які пройшли відбір до цієї компанії. Це робить процес підготовки резюме, орієнтованого на вимоги компанії, і знижує шанси проходження першого етапу відбору.

Вищезазначене підкреслює актуальність створення платформи, яка дозволяє кандидатам публікувати резюме без персональних даних, що були схвалені роботодавцями, і водночас надає можливість обговорювати різні аспекти процесу працевлаштування. Така платформа може суттєво підвищити рівень успішності працевлаштування, оскільки дозволяє користувачам отримувати зворотній зв'язок та отримувати інформацію з реальних прикладів.

Метою даного дипломного проєкту є покращення процесу аналізу резюме, за допомогою якого шукачі роботи зможуть підвищити якість та ефективність своїх резюме. Передбачається реалізація функціоналу для завантаження, напів автоматичного прибирання персональних даних, модерації та публікації резюме, а також забезпечення обміну думок між користувачами, в контексті оцінки ефективності представлених документів.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

– аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;

– аналіз існуючих рішень обраного завдання розробки у рамках ДП;

– аналіз та моделювання бізнес-процесів;

– розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;

– постановка завдання на розробку програмного забезпечення ДП;

– розроблення архітектури програмного забезпечення;

– розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;

– конструювання та розроблення програмного забезпечення;

– аналіз безпеки даних програмного забезпечення;

– аналіз якості та тестування програмного забезпечення;

– розгортання та супровід програмного забезпечення;

– створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

На сьогодення, ринок праці переживає достатньо складну ситуацію, коли роботодавці намагаються знайти ідеальних кандидатів з великим обсягом знань, кількість вакансій на ринку зменшується, а конкуренція тільки росте. За даними популярного порталу пошуку роботи в США - Indeed, наприклад кількість вакансій у IT-сфері за останні 3 роки знизилась у 4 рази [1], коли кількість робітників у сфері IT в США за останні 5 років збільшилась у з 2.165млн до 2.4млн[2]. Дивлячись на загальну статистику кількості вакансій для всіх професій теж статистика не радує шукачів роботи, так як за останні 3

роки кількість вакансій впала близько на 38%[3]. Розміщуючи вакансію на порталах пошуку роботи, компанія отримує величезну кількість відгуків від шукачів, з яких їм треба вибрати невелику кількість для інтерв'ювання. Авжеж для шукачів роботи це теж є проблемою через те, що серед тони кандидатів їм потрібно виділитись, з цим їм якраз допомагає резюме. Резюме – це документ, що містить персональні дані, опис досвіду людини, освіти та інші відомості, документ, який при правильному оформленні і змісті надасть можливість гарно пройти HR скрінінг.

Наразі серед основних напрямків розвитку в цій галузі є: системи для автоматичної перевірки резюме для компаній, системи для створення резюме по шаблонам та системи-агенти для автоматичного пошуку вакансій на основі резюме користувача.

Серед недоліків поточного стану предметної області можна зазначити відсутність спеціалізованих платформ для обміну досвідом тем дотичних до резюме та порталів де можна подивитись резюме інших людей без персональних даних так в які компанії пройшло це резюме. В області немає сервісу, який би давав можливість користувачам ділитися своїм резюме легко й не публікуючи своїх персональних даних. Так як зміна роботи є достатньо делікатним питанням, шукачі роботи які перебувають у трудових відносинах не хотіли щоб їх роботодавець знав, що вони шукають роботу.

Тому у рамках дипломного проєкту для покращення ефективності резюме шукачів роботи було прийнято рішення розробити додаток, який надасть користувачам можливість ділитись резюме без персональних даних, отримувати поради з приводу резюме, та мати можливість за допомогою інструментів застосунку швидко шукати приклади резюме, які вже пройшли HR скрінінг в якісь конкретні компанії.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку «ShareCV». Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

На сьогодні існує багато форумів де люди можуть обмінюватись різною інформацію та отримувати відгуки від користувачів зі всього світу чи застосунки які дозволяють створювати форми для отримання зворотнього зв’язку стосовно будь якої інформації, в тому числі резюме. Але усі ці застосунки не є вузьконаправленими у сфері надання користувачам структурованих даних з приводу резюме інших людей та їх досвіду та зручні інструменти, щоб поділитися своїм резюме для оцінки.

Одним з аналогів є reddit.com[4] (рис 1.1)

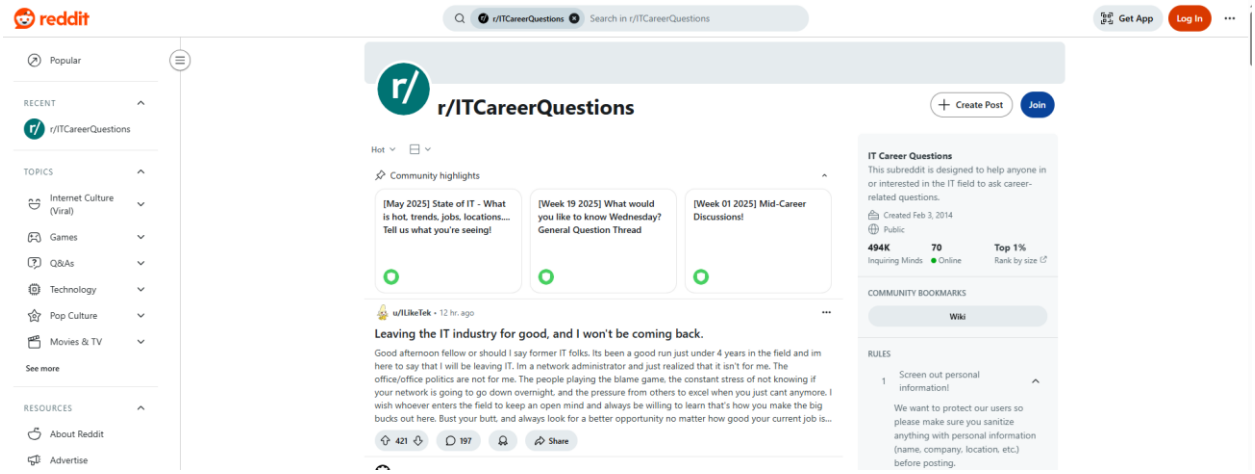


Рисунок 1.1 – Скріншот сайту reddit.com[4]

На форумі Reddit існує багато тематичних спілок, одною з таких є «ITCareerQuestions», де люди обговорюють питання стосовно кар’єри в IT. Користувач може опублікувати будь яке питання з додаванням посилань/фото/відео матеріалів й інші користувачі порталу мають змогу висловити свою думку у виді коментарів до посту.

Функціонал порталу більш зосереджений на дискусіях (рис 1.2), на сайті є можливість пошуку по ключовому слову, без інших фільтрів, що б допомогли користувачу знайти якусь інформацію наприклад по спеціальній компанії.

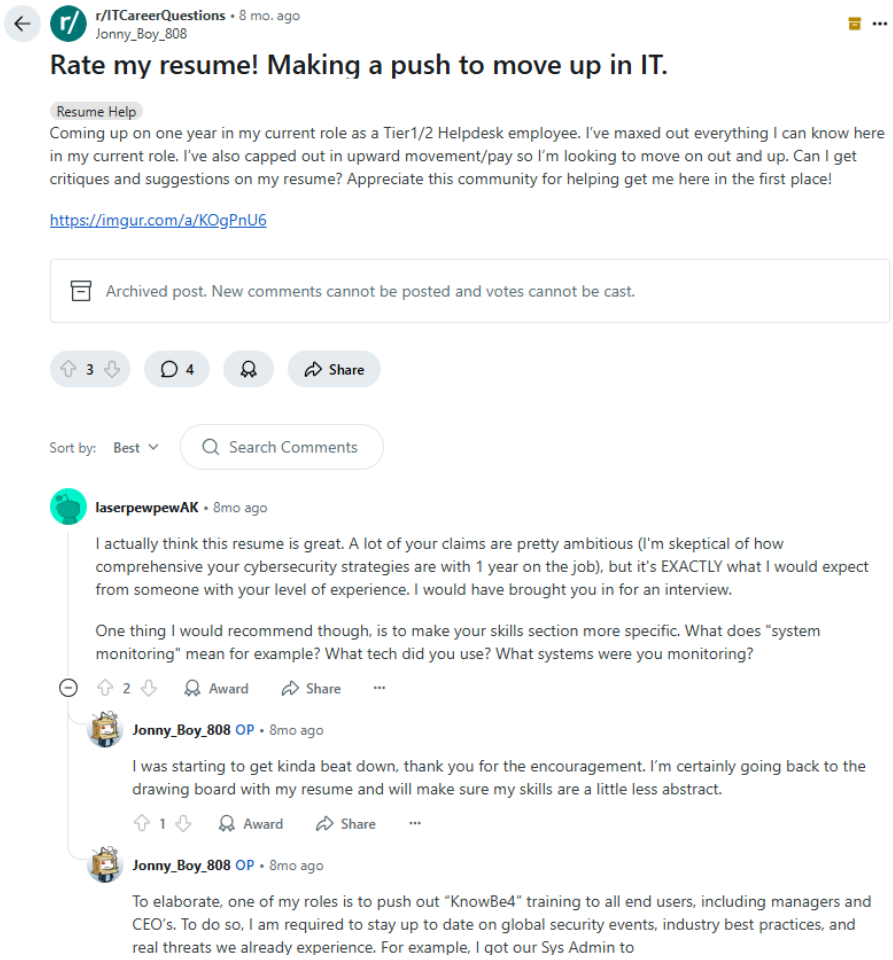


Рисунок 1.2 – Скріншот сторінки коментарів порталу reddit.com [4]

Другим аналогом виступає сервіс – jotform.com[5](рис 1.3).

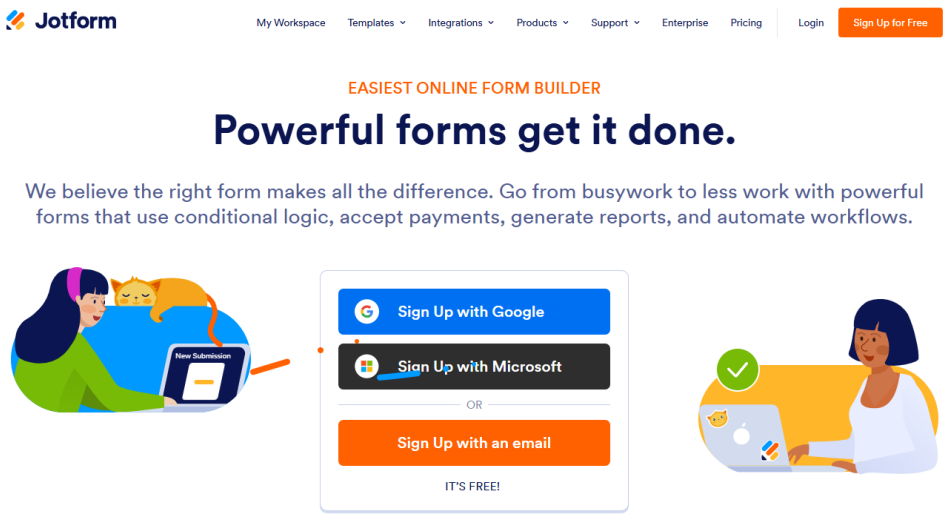


Рисунок 1.3 – Скріншот головної сторінки сайту jotform.com[5]

Даний сервіс надає можливість користувачам створювати кастомізовані форми для збору зворотнього зв’язку(рис 1.4).

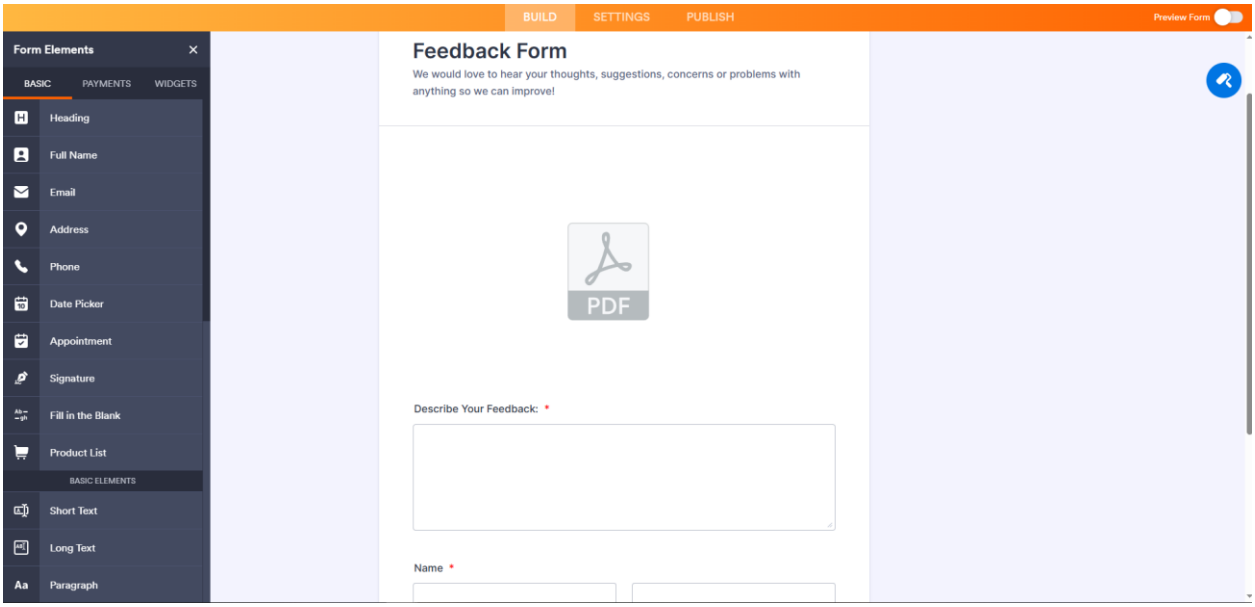


Рисунок 1.4 – Скріншот сторінки створення форми сайту jotform.com[5]

В якості недоліків порівнянно в контексті дипломного проекту, сервіс надає можливість тільки тільки отримувати відгук від користувача без можливості дискусії. Також сервіс не має можливостей переглядати публічно форми інших користувачів, з резюме чи чимось іншим.

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння аналогів

Функціонал	Дипломний проєкт (ShareCV)	reddit.com	jotform.com	Пояснення
Поширення резюме	+	+	+	На порталі Reddit користувачі можуть створити пост з будь якою темою, зокрема попросити відгук про їх резюме теж. На порталі JotForm можливо створити Feedback form для збору відгуків про своє резюме
Пошук резюме інших користувачів з багатьма фільтрами	+	-	-	Reddit: можливо шукати пости тільки по назві.
Додавання допису(резюме) до обраних	+	+	-	Reddit: можливо додати буль який допис до обраних.
Функція прибирання персональних даних резюме перед публікацією	+	-	-	
Можливість коментування дописів(резюме)	+	+	-	Reddit та ShareCV мають дуже схожі системи коментування, де користувачі можуть дискутувати один з одним в коментарях, але в Reddit користувачі бачуть аккаунти одне одного

Продовження таблиці 1.1

Можливість змінювати рейтинг коментаря	+	+	-	

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Вибір архітектури програмного забезпечення є критичним етапом, який визначає його гнучкість, масштабованість та надійність.

В підрозділі буде описано архітектурні патерни проектування ПЗ, та порівняння між ними.

Monolithic Architecture (Монолітна архітектура):

Ідея шаблону полягає в збірці та розгортанні системи як єдине ціле. Усі домені модулі знаходяться в одній кодовій базі, запускаються за допомогою одного процесу.

Серед переваг шаблону модна назвати:

– простота розробки та відладки;

– швидке розгортання системи;

– проста робота з транзакціями, через єдину БД;

Але недоліки теж присутні, такі як:

– складність масштабування;

– при зміні в одному модулі потрібно перезібрати весь застосунок для розгортання;

– тривале розгортання через кількість систем;

Microservices Architecture (Мікросервісна архутектура):

Мікросервіси – це набір ізольованих сервісів, кожний з яких: реалізує єдину бізнес-функцію, розгортається незалежно від інших, за допомогою мережі взаємодіє з іншими сервісами. Кожний сервіс має свою БД та може бути написаний на різних технологіях та у різних оточеннях.

В даного шаблону перевагами виступають:

– горизонтальне масштабування;
– незалежне розгортання;
– ізоляція багів та помилок;

Серед недоліків можна виділити:

– підвищення складність інфраструктури(мережа, балансування, відказостійкість);
– підвищена затримка через міжсервісні визови;
– потребує DevOps(CI/CD, логування, моніторинг);

Layered Architecture (Шарова архітектура)

Шарова архітектура – це підхід, при якому відбувається структурне розбиття коду на логічні шари: Presentational, Application, Domain та Infrastructure.

З приводу переваг:

– Проста й легко зрозуміла організація коду;
– Можливість впровадження unit-тестів по шарам;
– Легка для підтримки та розвитку.

Недоліки:

– Часто виникає сильна зв’язаність між шарами;
– Порушення принципу принцип єдиної відповідальності;
– Труднощі у впроваджені гнучкої архітектури;

Після проведення аналізу архітектурних патернів, для реалізації дипломного проєкту було обрано монолітну архітектуру, де є окремі сервіси, Java сервіс для бізнес логіки, Python сервіс для прибирання персональних даних, Nginx[9] сервіс для фронтенд застосунку. Цей тип може нагадувати Client-Server патерн або мікросервіси. Але в випадку дипломної роботи, кожен представлений вище сервіс є монолітним застосунком, з єдиною БД. Та двома бекенд сервісами, їх було поділено на 2 сервіси через відповідність технологій

для їх задач, Java більш підходить для складних застосунків з багатою бізнес логікою, а Python надає багато інструментів для роботи з машинним навчанням та штучним інтелектом загалом.

1.4 Аналіз та моделювання бізнес-процесів

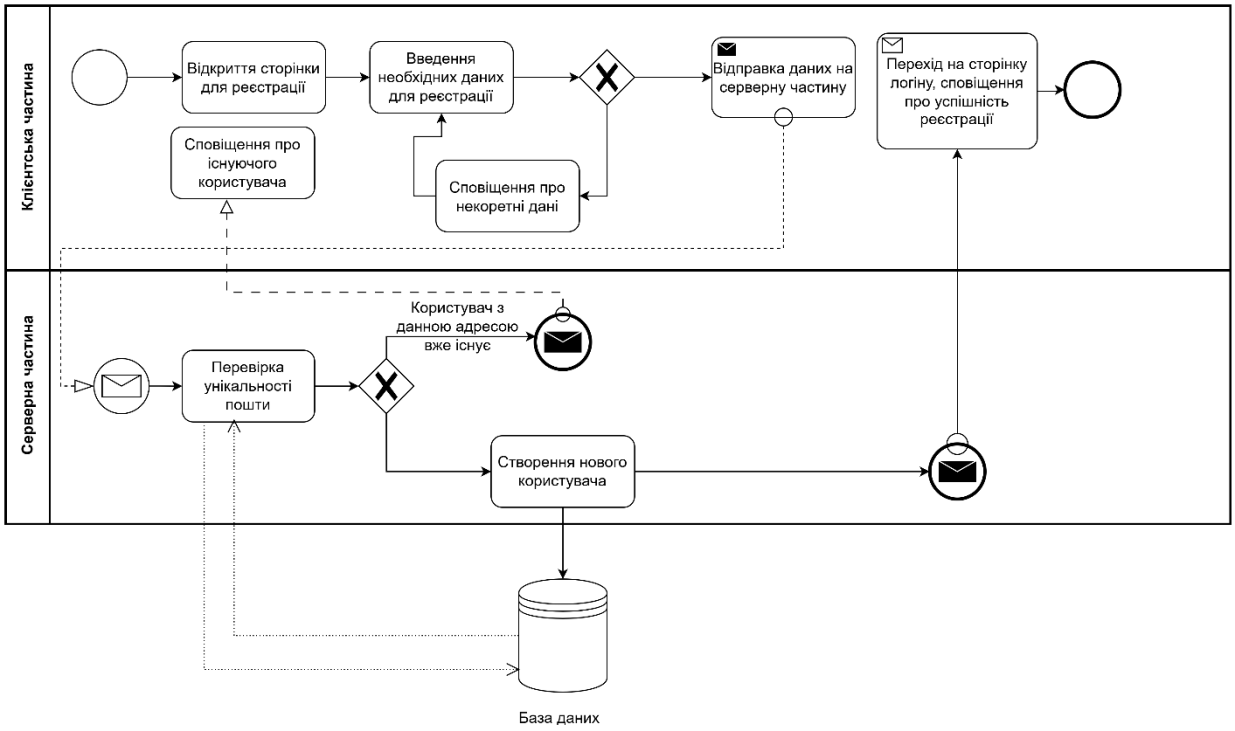


Рисунок 1.5 – BPMN схема реєстрації користувача в системі

Опис моделі бізнес-процесу реєстрації користувача:

– користувач переходить на сторінку реєстрації;

– користувач заповнює поля реєстрації;

– якщо введені дані некоректні, користувачу відображається сповіщення про некоректні дані;

– якщо дані правильні, то дані відправляються до серверу;

– сервер отримує дані від клієнтської частини;

– сервер перевіряє унікальність пошти;

– якщо дана пошта вже існує в базі даних, сервер надсилає повідомлення на клієнтську частину;

– якщо пошта нова, сервер створює нового користувача та відсилає клієнтській частині сповіщення про успішну реєстрацію;

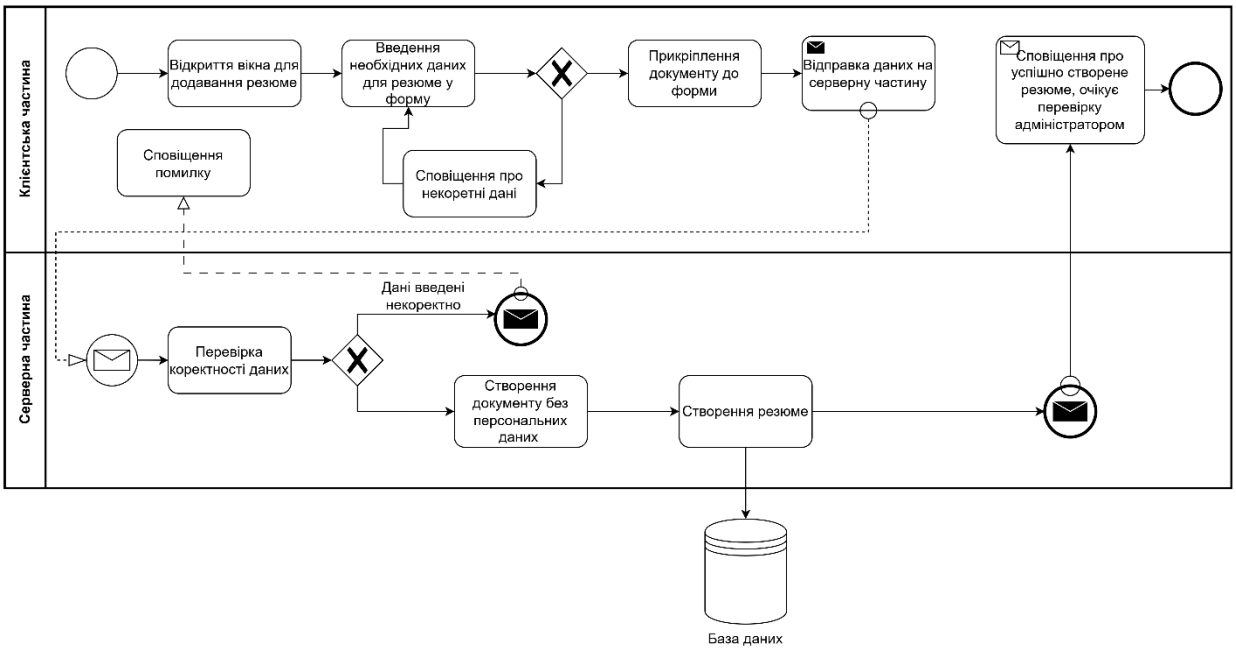


Рисунок 1.6 – BPMN схема додавання резюме до системи

Опис моделі бізнес-процесу додавання резюме:

– користувач відкриває вікно для додавання резюме;

– користувач вводить необхідні дані для резюме;

– якщо введені дані некоректні, користувачу відображається сповіщення про некоректні дані;

– якщо дані правильні, то дані відправляються до серверу;

– сервер отримує дані від клієнтської частини;

– сервер перевіряє дані на коректність;

– якщо дані некоректні, користувачу відображається повідомлення про помилку;

– якщо дані коректні, сервер створює документ без персональних даних;

– сервер створює резюме в базі даних;

– сервер надсилає сповіщення на клієнтську частину про успішне створення резюме;

Діаграма бізнес процесу «Модерація резюме адміністратором» розміщена у графічних матеріалах.

Опис моделі бізнес-процесу підтвердження резюме адміністратором:

– адміністратор входить на сторінку «Admin Dashboard»;

– адміністратор вибирає резюме для детального перегляду в таблиці;

– застосунок перевіряє чи резюме має статус «Підтверджено»;

– якщо статус «Підтверджено», адміністратор повертається до вибору іншого резюме;

– якщо статус не «Підтверджено» продовжується наступний крок;

– адміністратор перевіряє коректність видалення персональних даних;

– якщо видалення некоректне адміністратор вручну має відкоригувати файл та завантажити його в систему. Клієнт відправляє новий документ на сервер. Сервер змінює старий документ на новий та надсилає сповіщення клієнту про успішну заміну документа;

– клієнтська частина відправляє запит до серверу для зміни статусу резюме на «Підтверджено»;

– якщо зміна статусу неможлива(поточний статус резюме не дозволяє це зробити), сервер надсилає клієнтській частині сповіщення про помилку;

– якщо зміна статусу можлива, сервер змінює статус резюме в базі даних на «Підтверджено»;

– сервер надсилає клієнтській частині сповіщення ро успішну зміну статусу;

Висновки до розділу

У розділі проведено комплексне передпроєктне обстеження предметної області, що є основою для розробки веб-застосунку "ShareCV". Воно включало в себе: аналіз поточного стану ринку праці, визначення актуальних проблем,

постановку завдання розробки, дослідження існуючих аналогів та технічних рішень, а також моделювання ключових бізнес процесів.

На відміну від існуючих рішень, запорпоноване рішення фокусується саме на обміні досвіду пов'язаним з створенням ефективного резюме без розкриття персональних даних, а також на створені інструментів фільтрації та коментуванні резюме. З приводу архітектури застосунку було визначено монолітний підхід з виділенням декількох сервісів відповідно до технологічний переваг кожної з мов програмування на який базуються сервіси.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Діаграма варіантів використання знаходиться у графічних матеріалах.

В таблицях 2.1-2.21 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-

Продовження таблиці 2.1

Flow of Events	Користувач переходить на сторінку реєстрації. У полях, призначених для реєстрації, користувач вводить відповідні дані: логін та пароль в системі та його повторення для підтвердження. Після введення даних користувач натискає кнопку «Зареєструватися». У разі успішної реєстрації на екрані з'являється з'являється повідомлення про успішну реєстрацію, і користувач потрапляє на сторінку входу в систему.
Extension	При введенні невірних даних кнопка реєстрації стає неактивною. Якщо користувач неправильно ввів певне поле, то воно підсвічується червоним кольором.
Post-Condition	Генерація сторінки користувача, перехід на сторінку авторизації

Таблиця 2.2 – Варіант використання UC-02

Use case name	Авторизація користувача
Use case ID	UC-02
Goals	Авторизація користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає війти в систему
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач потрапляє на сторінку авторизації. У полях для входу в систему користувач вводить відповідну інформацію: логін та пароль користувача в системі. Після заповнення даних користувач натискає кнопку «Увійти». Після Далі користувач потрапляє на головну сторінку сайту.
Extension	При введенні невірних даних кнопка реєстрації стає неактивною. Якщо користувач неправильно ввів певне поле, то воно підсвічується червоним кольором.
Post-Condition	Перехід на головну сторінку системи

Таблиця 2.3 – Варіант використання UC-03

Use case name	Створення резюме
Use case ID	UC-03
Goals	Відправка резюме до розгляду адміністрації
Actors	Зареєстрований користувач
Trigger	Користувач бажає відправити резюме
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач в шапці сайту або на головній сторінці натиская кнопку "Додати резюме". Заповнює форму про свій досвід, компанії в які було подано резюме, та сам PDF документ резюме. Після заповнення форми користувач натискає кнопку "Відправити".
Extension	При введенні невірних даних кнопка реєстрації стає неактивною. Якщо користувач неправильно ввів певне поле, то воно підсвічується червоним кольором.
Post-Condition	Інформація про резюме додається до бази даних і очікує на затвердження адміністратором.

Таблиця 2.4 – Варіант використання UC-04

Use case name	Перегляд списку опублікованих резюме
Use case ID	UC-04
Goals	Перегляд інформації про всі резюме, що публічно доступні в системі
Actors	Зареєстрований користувач
Trigger	Користувач бажає переглянути інформацію про публічно опубліковані резюме
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач переходить до сторінки "Resumes". Система надає список всіх публічних резюме, так даних про них, таких як компанії в які було подано резюме, кількісь відмов та кількість схвалень резюме в різних компаніях, кількість років досвіду в користувача тощо.

Продовження таблиці 2.4

Extension	-
Post-Condition	Користувач має можливість перейти до детальної інформації резюме

Таблиця 2.5 – Варіант використання UC-05

Use case name	Пошук резюме з фільтрами
Use case ID	UC-05
Goals	Користувач може здійснити фільрацію списку резюме
Actors	Зареєстрований користувач
Trigger	Користувач здійснює фільтрацію резюме
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач переходить до сторінки "Resumes", вибирає фільтри(наприклад компанії в які подано резюме, спеціальність людини, що подала резюме тощо) і активує фільтрацію. Застосунок відображає відфільтровані дані відповідно до заповнених фільтрів.
Extension	При введенні фільтрів за яких не знайдено жодного співпадіння система сповіщає про це користувача.
Post-Condition	Користувач має можливість перейти до детальної інформації резюме

Таблиця 2.6 – Варіант використання UC-06

Use case name	Перегляд детальної інформації конкретного резюме
Use case ID	UC-06
Goals	Переглянути детальну інформацію про резюме
Actors	Зареєстрований користувач
Trigger	Користувач переходить до детальної сторінки резюме
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач переходить до сторінки "Resumes" же вибирає у списку доступних резюме те, які він хоче дослідитти детально й натискає на нього. Після цього користувач може побачити детальну інформацію про резюме(PDF документ, спейіалізація відправника, компанії в які резюме пройшло та не пройшло HR скрінінг тощо).
Extension	-

Продовження таблиці 2.6

Post-Condition	Користувач має можливість написати коментар до резюме чи вступити вже в існуючу дискусію.

Таблиця 2.7 – Варіант використання UC-07

Use case name	Вихід з системи
Use case ID	UC-07
Goals	Вийти з аккаунта
Actors	Зареєстрований користувач
Trigger	Користувач хоче вийти з аккаунту
Pre-conditions	Користувач вже ввійшов в систему
Flow of Events	Користувач відкриває випадаючий список у правому верхньому куту та натискає кнопку "Log Out"
Extension	-
Post-Condition	-

Таблиця 2.8 – Варіант використання UC-08

Use case name	Написання коментаря до резюме
Use case ID	UC-08
Goals	Написати коментар до резюме
Actors	Зареєстрований користувач
Trigger	Користувач хоче написати відгук чи пораду до резюме.
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач знаходиться на детальній сторінці резюме й у нижній частині сторінки заповнює тексту до коментаря й натискає кнопку "Send", щоб опубліковати коментар.
Extension	-
Post-Condition	Коментар користувача опубліковано на детальній сторінці резюме

Таблиця 2.9 – Варіант використання UC-09

Use case name	Написання відповіді до коментаря
Use case ID	UC-09

Продовження таблиці 2.9

Goals	Відповісти на коментар іншого користувача.
Actors	Зареєстрований користувач
Trigger	Користувач хоче відповісти на коментар іншого користувача.
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач знаходиться на детальній сторінці резюме й у частині коментарів натискає кнопку "Reply", пілся цього заповнює форму з текстом відповіді на коментар й натискає кнопку "Send", щоб відправити відповідь.
Extension	-
Post-Condition	Відповідь користувача опубліковано на детальній сторінці резюме

Таблиця 2.10 – Варіант використання UC-10

Use case name	Оновлення стану голосування коментаря
Use case ID	UC-10
Goals	Відреагувати та коментар
Actors	Зареєстрований користувач
Trigger	Користувач хоче відреагувати позитивно/негативно на коментар
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач знаходиться на детальній сторінці резюме й у частині коментарів натискає кнопку стрілки вверх/вниз, щоб відреагувати позитивно/негативно до коментаря.
Extension	-
Post-Condition	Реакція користувача зареєстрована в систему, загальний рейтинг коментаря змінено в залежності від позитивної/негативної реакції користувача

Таблиця 2.11 – Варіант використання UC-11

Use case name	Сортування коментарів
Use case ID	UC-11
Goals	Відсортувати коментарі за критерієм
Actors	Зареєстрований користувач

Продовження таблиці 2.11

Trigger	Користувач хоче відсортувати коментарі за критерієм
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач знаходиться на детальній сторінці резюме й у частині коментарів відкриває випадаючий список з варіантами сортування, та вибирає один з можливих варіантів, сортування за користю(кометарі з найбільшим рейтингом), найновіщі, найстаріщі.
Extension	-
Post-Condition	Коментарі відстортовано в залежносто від вибраної опції

Таблиця 2.12 – Варіант використання UC-12

Use case name	Перегляд відправлених резюме користувача
Use case ID	UC-12
Goals	Переглянути резюме, які користувач відправив до застосунку
Actors	Зареєстрований користувач
Trigger	Користувач хоче переглянути резюме, які відправив до застосунку
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач відкриває випадаючий список у правому верхньому куту та натискає кнопку "My Profile". Користувача переадресовує на сторінку з даними користувача, де він може обрати вкладку "My Resumes" же побачить "картки" резюме, які він відправив.
Extension	Якщо в користувач ще не додав до застосунку жодного резюме йому буде показано сповіщення про це.
Post-Condition	Користувач може управляти станом резюме та їх даними.

Таблиця 2.13 – Варіант використання UC-13

Use case name	Приховування резюме з публічного доступу.
Use case ID	UC-13
Goals	Сховати резюме, яке користувач відправив до застосунку
Actors	Зареєстрований користувач

Продовження таблиці 2.13

Trigger	Користувач хоче сховати резюме, які відправив до застосунку
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	Користувач відкриває випадаючий список у правому верхньому куту та натискає кнопку "Profile". Користувача переадресовує на сторінку з даними користувача, де він може обрати вкладку "My Resumes" де побачить "картки" резюме, які він відправив. У картці резюме користувач натискає кнопку "Hide".
Extension	Якщо в користувач ще не додав до застосунку жодного резюме йому буде показано сповіщення про це.
Post-Condition	-

Таблиця 2.14 – Варіант використання UC-14

Use case name	Перегляд документу резюме відправленого користувачем
Use case ID	UC-14
Goals	Переглянути документ резюме, яке користувач відправив до застосунку.
Actors	Зареєстрований користувач
Trigger	Користувач хоче подивитись документ резюме, яке відправив до застосунку.
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	На сторінці "My Resumes" користувач натискає на "картку" резюме.
Extension	Якщо в користувач ще не додав до застосунку жодного резюме йому буде показано сповіщення про це.
Post-Condition	Користувачу візуалізується вспливаюче вікно з документом.

Таблиця 2.15 – Варіант використання UC-15

Use case name	Додавання резюме в закладки
Use case ID	UC-15
Goals	Додати резюм в закладки
Actors	Зареєстрований користувач

Продовження таблиці 2.15

Trigger	Користувач хоче додати резюме в закладки.
Pre-conditions	Користувач вже зареєстрований в системі
Flow of Events	На детальній сторінці резюме користувач натискає кнопку "Bookmark".
Extension	Якщо резюме вже було додано до закладок, кнопка "Bookmark" буде мати іншу кольорову гаму та текст "Bookmarked".
Post-Condition	Резюме додано до списку обраних для користувача.

Таблиця 2.16 – Варіант використання UC-16

Use case name	Перегляд списку резюме в закладках
Use case ID	UC-16
Goals	Переглянути список резюме в закладках.
Actors	Зареєстрований користувач
Trigger	Користувач хоче переглянути список резюме в закладках.
Pre-conditions	Користувач вже зареєстрований в системі.
Flow of Events	Користувач відкриває випадаючий список у правому верхньому куту та натискає кнопку "My Profile". Користувача переадресовує на сторінку з даними користувача, де він може обрати вкладку "My Bookmarks" же побачить "картки" обраних резюме, які він відмітив.
Extension	Якщо в користувач ще не додав жодного резюме до закладок йому буде показано сповіщення про це.
Post-Condition	-

Таблиця 2.17 – Варіант використання UC-17

Use case name	Перегляд списку відправлених резюме на обробку
Use case ID	UC-17
Goals	Переглянути список відправлених резюме на обробку.
Actors	Користувач з правами адміністратора
Trigger	Адміністратор хоче переглянути список відправлених резюме на обробку.
Pre-conditions	Адміністратор ввійшов в систему та знаходиться на сторінці "Admin Dashboard".

Продовження таблиці 2.17

Flow of Events	Адміністратор ввійшов в систему та знаходиться на сторінці "Admin Dashboard". Застосунок відображає список усіх резюме, що були надіслані звичайними користувачами на обробку.
Extension	-
Post-Condition	Адміністратор переглянув перелік поданих резюме.

Таблиця 2.18 – Варіант використання UC-18

Use case name	Перегляд детальної інформації поданого резюме
Use case ID	UC-18
Goals	Переглянути детальну інформацію поданого резюме.
Actors	Користувач з правами адміністратора
Trigger	Адміністратор хоче переглянути детальну інформацію поданого резюме.
Pre-conditions	Адміністратор ввійшов в систему та знаходиться на сторінці "Admin Dashboard".
Flow of Events	Адміністратор ввійшов в систему та знаходиться на сторінці "Admin Dashboard". У списку усіх резюме адміністратор обирає потрібне й натискає кнопку "Details".
Extension	-
Post-Condition	Застосунок показує детальну інформацію про резюме(два документи, оригінальний та з скритими персональними даними та інші дані).

Таблиця 2.19 – Варіант використання UC-19

Use case name	Зміна статусу поданого резюме
Use case ID	UC-19
Goals	Змінити статус поданого резюме.
Actors	Користувач з правами адміністратора
Trigger	Адміністратор хоче змінити статус поданого резюме.
Pre-conditions	Адміністратор ввійшов в систему та знаходиться на детальній сторінці резюме.
Flow of Events	Адміністратор натискає кнопки "Approve" або "Reject" якщо він хоче підтвердити або відхилити публікацію резюме до застосунку.
Extension	-

Продовження таблиці 2.19

Post-Condition	Статус резюме змінено. Якщо статус резюме змінено на підтверджене воно з'являється у таблиці резюме для звичайних користувачів.

Таблиця 2.20 – Варіант використання UC-20

Use case name	Заміна документу резюме з скритими персональними даними
Use case ID	UC-20
Goals	Замінити новий документ резюме з скритими персональними даними.
Actors	Користувач з правами адміністратора
Trigger	Адміністратор хоче заміни документ резюме з скритими персональними даними.
Pre-conditions	Адміністратор ввійшов в систему та знаходиться на детальній сторінці резюме.
Flow of Events	Адміністратор натискає кнопку "Upload new file" та у спливаючому вікні додає новий документ. Після цього натискає кнопку "Send".
Extension	-
Post-Condition	Було замінено документ з скритими персональними даними на новий.

Таблиця 2.21 – Варіант використання UC-21

Use case name	Видалення акаунту
Use case ID	UC-21
Goals	Видалити акаунту користувача зі всіма його даними.
Actors	Авторизований користувач
Trigger	Користувач хоче видалити свій акаунт.
Pre-conditions	Користувач вже зареєстрований в системі.
Flow of Events	Rjhbcnedfx натискає кнопку "Меню" та у спливаючому списку вибирає опцію «Видалити обліковий запис». Після цього натискає кнопку у спливаючому вікні підтверджує вибір.
Extension	-

Продовження таблиці 2.21

Post-Condition	Дані про користувача були видалені, користувача переведено на сторінку реєстрації.

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.21 наведено загальну модель вимог, а в таблиці 2.22 наведений опис функціональних вимог до програмного забезпечення.

Таблиця 2.22 – Загальна модель вимог

№	Назва	ID	Пріоритет	Ризик
1	Реєстрація користувача	FR-1	Високий	Середній
1.1	Перевірка даних при реєстрації	FR-2	Високий	Середній
1.2	Перенаправлення після реєстрації	FR-3	Середній	Низький
2	Авторизація користувача	FR-4	Високий	Середній
2.1	Повідомлення про помилки при вході	FR-5	Середній	Низький
2.2	Перенаправлення після входу	FR-6	Середній	Низький
2.3	Вихід із системи	FR-22	Середній	Низький
3	Створення резюме	FR-7	Високий	Середній
3.1	Перевірка даних у формі резюме	FR-8	Високий	Середній
3.2	Збереження резюме в базі	FR-9	Високий	Середній
4	Перегляд опублікованих резюме	FR-10	Високий	Середній
4.1	Фільтрація резюме	FR-11	Середній	Середній
4.2	Повідомлення про відсутність результатів	FR-12	Низький	Низький
4.3	Перегляд детальної інформації про резюме	FR-13	Високий	Середній
5	Коментування резюме	FR-14	Середній	Середній
5.1	Відповіді на коментарі	FR-15	Середній	Середній
5.2	Голосування за коментарі	FR-16	Середній	Середній
5.3	Оновлення рейтингу коментарів	FR-17	Середній	Середній
5.4	Сортування коментарів	FR-18	Низький	Низький

Продовження таблиці 2.22

6	Перегляд власних резюме у профілі	FR-19	Середній	Низький
6.1	Повідомлення про відсутність резюме	FR-20	Низький	Низький
6.2	Приховування резюме з публічного доступу	FR-21	Середній	Низький
7	Перегляд резюме у спливаючому вікні	FR-23	Низький	Низький
8	Додавання резюме до закладок	FR-24	Середній	Низький
8.1	Перегляд списку закладок	FR-25	Середній	Низький
8.2	Повідомлення про відсутність закладок	FR-26	Низький	Низький
9	Перегляд усіх резюме адміністратором	FR-27	Високий	Середній
9.1	Перегляд деталей резюме адміністратором	FR-28	Високий	Середній
9.2	Зміна статусу резюме	FR-29	Високий	Середній
9.3	Завантаження документа з прихованими даними	FR-30	Високий	Середній
10	Видалення даних користувача	FR-31	Високий	Високий

Таблиця 2.23 – Перелік функціональних вимог

ID	Опис вимоги
FR-1	Система повинна надавати можливість користувачу зареєструвати обліковий запис, ввівши логін і пароль, а також підтвердження пароля.
FR-2	Система повинна перевіряти правильність введених даних під час реєстрації та відображати повідомлення про помилки.
FR-3	Після успішної реєстрації користувач повинен бути перенаправлений на сторінку входу.
FR-4	Система повинна надавати можливість авторизованому користувачу увійти до системи, ввівши логін і пароль.
FR-5	У разі неправильного введення даних при авторизації система повинна відображати повідомлення про помилку.
FR-6	Після успішної авторизації користувач повинен бути перенаправлений на головну сторінку.

Продовження таблиці 2.23

FR-7	Авторизований користувач повинен мати можливість створити резюме, заповнивши форму з досвідом, компаніями та завантаживши PDF.
FR-8	Система повинна перевіряти правильність введених даних у формі резюме та відображати повідомлення про помилки.
FR-9	Після відправки резюме система повинна зберігати його в базі даних для подальшого затвердження.
FR-10	Авторизований користувач повинен мати можливість переглядати список опублікованих резюме з детальною інформацією.
FR-11	Користувач повинен мати можливість фільтрувати список резюме за різними критеріями (компанія, спеціальність тощо).
FR-12	У разі відсутності результатів фільтрації резюме система повинна повідомити користувача про це.
FR-13	Користувач може переглядати детальну інформацію про конкретне резюме, включаючи PDF, спеціалізацію, статуси по компаніях.
FR-14	Користувач може залишати коментарі до резюме.
FR-15	Користувач може відповідати на коментарі інших користувачів.
FR-16	Користувач може голосувати за коментарі (позитивно або негативно).
FR-17	Система повинна оновлювати рейтинг коментаря відповідно до голосів.
FR-18	Користувач може сортувати коментарі за критеріями: найкорисніші, найновіші, найстаріші.
FR-19	Користувач може переглядати список власних резюме у профілі.
FR-20	Якщо користувач ще не додав жодного резюме, система повинна повідомити про це.
FR-21	Користувач може приховати своє резюме з публічного доступу.
FR-22	Користувач може вийти з системи, завершивши сесію.
FR-23	Користувач може переглядати документ резюме у вигляді спливаючого вікна.
FR-24	Користувач може додати резюме до закладок.
FR-25	Користувач може переглядати список резюме, доданих до закладок.
FR-26	Якщо користувач ще не додав жодного резюме до закладок, система повинна повідомити про це.
FR-27	Адміністратор може переглядати список усіх резюме, що були надіслані на обробку.

Продовження таблиці 2.23

FR-28	Адміністратор може переглядати детальну інформацію про подане резюме, включаючи два документи: оригінальний та з прихованими персональними даними.
FR-29	Адміністратор може змінювати статус поданого резюме (схвалити або відхилити).
FR-30	Адміністратор може завантажити новий документ резюме з прихованими персональними даними.
FR-31	Користувач повинен мати можливість видалити свій обліковий запис з усіма даними у веб-застосунку.

Матрицю трасування вимог представлено на рисунку 2.1.

Use Case ID	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28	FR-29	FR-30	FR-31
UC-01	✔	✔	✔																												
UC-02				✔	✔	✔																									
UC-03							✔	✔	✔																						
UC-04										✔																					
UC-05											✔	✔																			
UC-06													✔																		
UC-07																						✔									
UC-08														✔																	
UC-09															✔																
UC-10																✔	✔														
UC-11																		✔													
UC-12																			✔	✔											
UC-13																					✔										
UC-14																							✔								
UC-15																								✔	✔						
UC-16																										✔					
UC-17																											✔				
UC-18																												✔			
UC-19																													✔		
UC-20																														✔	
UC-21																															✔

Рисунок 2.1 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Розробка нефункціональних вимог є основою для роботи з програмами, оскільки ці умови визначають весь користувацький досвід веб-додатку. Нижче наведені нефункціональні вимоги:

– безпека (програма повинна захищати дані зареєстрованих користувачів від несанкціонованого використання);

– зручність використання (інтерфейс має бути інтуїтивно зрозумілим для користувачів);

– доступність (застосунок має забезпечувати час безвідмовної роботи не менше 99,9% (за винятком планового технічного обслуговування);

– відповідність (застосунок повинен відповідати чинним нормам і законам);

2.4 Аналіз економічних показників програмного забезпечення

У даному підрозділі здійснюється розрахунок та аналіз ключових економічних показників для розроблюваного програмного забезпечення. Оцінка проводилася за допомогою моделі COCOMO II (Post-Architecture), виходячи з обсягу програмного коду та номінальних характеристик проекту.

Ключові вхідні параметри для розрахунку були наступними:

– Розмір ПЗ: 12791 нових рядків коду (SLOC), що еквівалентно 12.791 KSLOC.

– Фактори масштабу та вартості: Всі 5 факторів масштабу (Precedentedness, Development Flexibility, Architecture/Risk Resolution, Team Cohesion, Process Maturity) та 17 факторів вартості (щодо продукту, персоналу, платформи та проекту) були встановлені на рівень Nominal. Це призвело до розрахункового значення експоненти зусиль B ≈ 1.10 та коефіцієнта коригування зусиль EAF = 1.00.

– Вартість праці: $600 за людино-місяць.

Розрахунок основних показників для центральних фаз розробки (Elaboration та Construction) дав наступні результати:

1. Зусилля (Effort): За формулою Effort (PM) = A * (KSLOC)^B * EAF, де A = 2.94, KSLOC = 12.791, B ≈ 1.10, EAF = 1.00, розрахункові зусилля склали Effort = 2.94 * (12.791)^1.10 * 1.00 ≈ 48.5 Людино-місяців (PM).
2. Тривалість (Schedule): За формулою Schedule (Months) = C * (Effort)^F, де C = 3.67, Effort = 48.5 PM, а експонента F = 0.28 + 0.20 * (B - 0.91) ≈ 0.318, розрахункова тривалість склала Schedule = 3.67 * (48.5)^0.318 ≈ 12.6 Місяців.

3. Вартість (Cost): За формулою Cost = Effort * Cost per Person-Month, вартість для цих фаз склала Cost = 48.5 PM * $600/PM = $29,100

Таблиця 2.24 – Економічні показники

Фаза	Зусилля (ПМ)	Тривалість (міс.)	Вартість ($)
Inception	2.9	1.6	$1,745
Elaboration	11.6	4.7	$6,982
Construction	36.8	7.9	$22,109
Transition	5.8	1.6	$3,491
Разом	57.1	15.8 (послід.)	$34,327

Таким чином, загальні показники проекту складають:

– загальні зусилля: 57.1 Людино-місяців;

– загальна послідовна тривалість: 15.8 Місяців (приблизно 1 рік і 4 місяці);

– загальна вартість: $34,327;

2.5 Постановка завдання на розробку програмного забезпечення

В рамках розробки програмного забезпечення «ShareCV» необхідно створити веб-застосунок, призначений для надання шукачам роботи інструменту для обміну досвідом та покращення власних резюме шляхом доступу до інших прикладів без персональних даних. Необхідно спроєктувати архітектуру на основі контейнеризованих сервісів, що забезпечить гнучкість, масштабованість та надійність системи. Застосунок має реалізовувати напівавтоматичне прибирання персональних даних.

Метою розробки програмного забезпечення є покращення процесу аналізу резюме, за допомогою якого шукачі роботи зможуть підвищити ефективність своїх резюме.

Мета досягається завдяки вирішенню наступних функціональних задач:

– застосунок повинен дозволяти користувачам завантажувати резюме та забезпечувати напівавтоматичне видалення персональних даних з подальшою адміністраторською модерацією перед публікацією;

– користувачі повинні мати можливість колективного обговорення та оцінки резюме через систему коментарів;

– застосунок повинен надавати інструменти для перегляду, пошуку та фільтрації опублікованих резюме за різними критеріями;

Висновки до розділу

У розділі було проведено аналіз та розробка вимог до програмного забезпечення.

Описано варіанти використання системи з ілюстрацією у виді діаграми варіантів використання. Близько двадцяти основних сценаріїв охоплюють повний цикл роботи користувача та адміністратора, від реєстрації та авторизації та створення й перегляду резюме до модерації резюме відправлених користувачами.

На основі варіантів використання сформульовано функціональні вимоги, які детально описують поведінку користувачів у застосунку. Для кожної вимоги було визначено ризик та пріоритет.

Визначено нефункціональні вимоги, що стосуються безпеки застосунку, зручності інтерфейсу, доступності та відповідності нормам.

Обраховано економічні показники програмного забезпечення й поставлено завдання на його розробку.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Веб-застосунок ShareCV побудована з окремими незалежними сервісами, кожен з яких призначений для виконання певних бізнес-функцій. Однак поточна стратегія розгортання, яка використовує єдиний docker-compose файл, орієнтована на контейнеризоване монолітне розгортання на одній віртуальній машині. Це означає, що навіть якщо сервіси контейнеризовані і розроблені окремо, вони розроблені для спільної роботи як один модуль у спільному хост-середовищі. Таким чином, ця модель поєднує переваги ізольованої розробки сервісів з простотою розгортання як моноліту, особливо на ранніх стадіях або в невеликих середовищах.

Для відображення архітектури було обрано тип діаграми C4 model. Існує чотири рівні структури цих діаграм: рівні 1 і 2 пропонують узагальнений погляд на систему, а рівні 3 і 4 вводять в складну організацію компонентів і взаємозв'язків між ними.

На першому рівні діаграми відображається система у контексті її зовнішніх залежностей, наприклад це може бути зовнішні системи, користувачі застосунку тощо. За допомогою цього представлення можна отримати загальне уявлення про застосунок та його окремі частини не вдаючись в технічні деталі. Застосунок ShareCV має декілька типів користувачів (звичайний користувач, адміністратор) та різні інтерфейси для кожної з ролей. Також через потребу в великій базі даних про компанії зі всього світу застосунок інтегрований з стороннім сервісом CompaniesAPI[6] для отримання даних про компанії(рис 3.1).

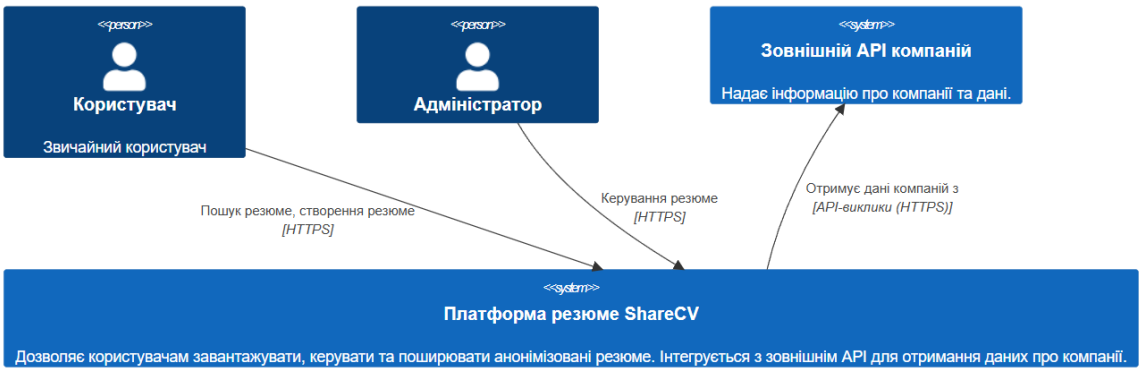


Рисунок 3.1 – Діаграма контексту

На другому рівні діаграми C4 model контекстна діаграма деталізується до рівня контейнерів (Контейнери представляють собою групи елементів, які разом розглядаються як єдине ціле з точки зору масштабованості, адміністрування та контролю системи). У випадку застосунку ShareCV діаграма складається з: фронтенд додатку, бекенд сервісу для обробки бізнес логіки застосунку, сервісу для прибирання персональних даних документів, бази даних та сховища для документів(рис 3.2).

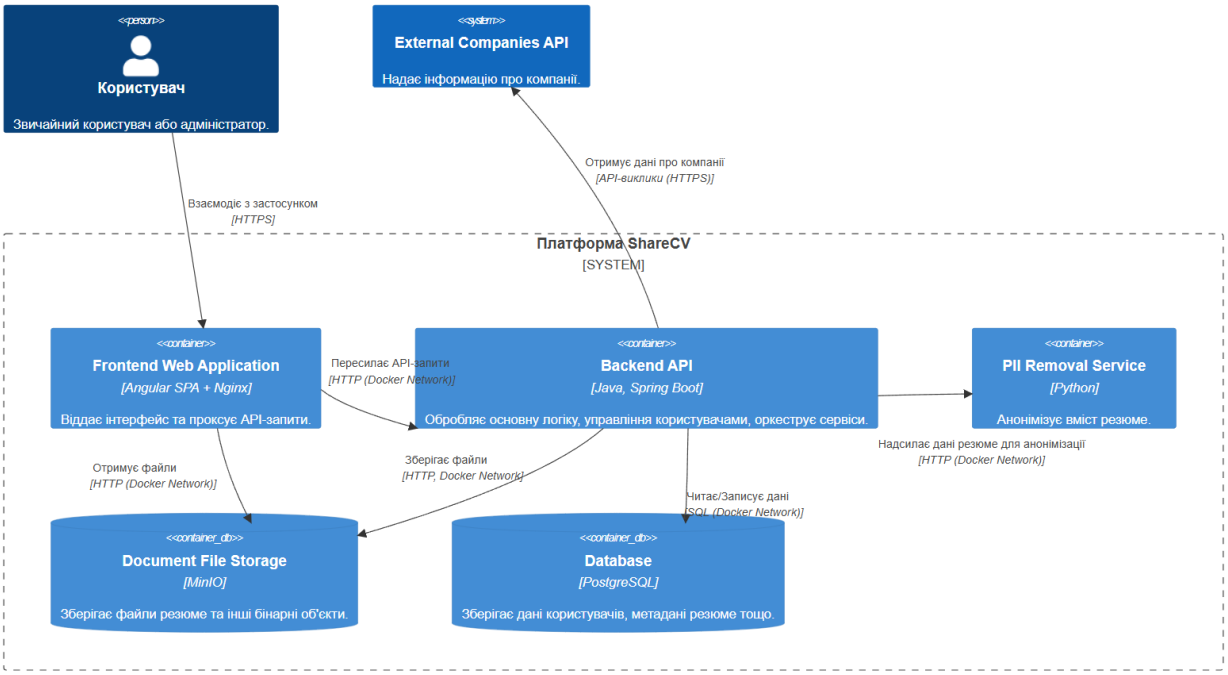


Рисунок 3.2 – Діаграма контейнерів

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

3.2.1 Вибір фреймворку

При виборі фреймворку для клієнтської частини було два основних кандидати: React та Angular[10]. Для того щоб зробити правильний вибір розглянемо переваги та недоліки кожного з них:

Таблиця 3.1 – Порівняння фронтенд фреймворків

Платформа	Переваги	Недоліки (в контексті ShareCV)
React	- React дозволяє обирати самостійно будь-які інструменти для імплементації функціоналу, таких як: маршрутизація, стан, API запити тощо; - бібліотека має активну спульноту, що створює багато стороннії бібліотек та інструментів; - Virtual DOM: технологія ефективного оновлення DOM, що покращує продуктивність;	- при виборі кожної з бібліотек, навіть для базового функціоналу, потім їх потрібно налаштовувати; - без строгих домовленостей в команді можливе порушення структури проекту, особоливо при зміні розробників проекту;
Angular[10]	- фреймворк має усі потрібні інструменти "з коробки", не потребує інших бібліотек для створення складних додатків; - структурованість(чітка організація компонентів, сервісів, директив і тд. Надає перевагу для розробки у команді); - Має особистий CLI, допомагає у збірці, генерації компонентів, тестуванні; - Новий реактивний підхід Signals підвищує продуктивність застосунків[11];	- початковий розмір збірки може бути трохи більшим за розмір застосунку React;

Отже для проекту ShareCV, який передбачає розробку складного, багатофункціонального фронтенд-додатку з чіткою структурою, високими вимогами до надійності та довгострокової підтримки, Angular[10] є більш доцільним вибором.

3.2.2 Вибір бази даних

У веб-застосунку ShareCV, для якого характерні високо структуровані дані, зокрема профілі користувачів, деталі резюме та перехресні зв'язки між досвідом і навичками, існує велика потреба у підтримці цілісності даних, підтримці транзакцій та обробці складних операцій пошуку і фільтрації. Тому реляційна база даних, а саме PostgreSQL, добре підходить для використання в цьому контексті. Бази даних, що використовують цю модель, забезпечують рівень узгодженості та надійності, які є одними з основних характеристик системи управління біографіями. Наприклад, PostgreSQL, який підтримує типи даних JSON, забезпечує гнучкість у зберіганні менш структурованих частин біографії, не втрачаючи при цьому переваг моделі зв'язків.

3.2.3 Вибір стороннього сервісу для отримання даних про компанії

Так як веб-застосунок ShareCV працює з резюме й має надавати максимально легкий процес подання резюме до платформи, отримання даних про компанії зі сторонніх сервісів може значно допомогти отримувати актуальні дані про назви компаній, їх логотипи тощо, що надасть користувачам більш швидкий метод заповнення форми про компанії в які користувач подавав свої резюме.

Для ShareCV, де основна мета інтеграції полягає у спрощенні процесу заповнення резюме шляхом автоматичного отримання коректної назви компанії та її логотипу, TheCompaniesAPI[6] виглядає більш збалансованим та доречним вибором.

Ключові переваги TheCompaniesAPI в контексті ShareCV:

– прямий доступ до необхідних даних (API чітко надає стандартизовану назву компанії та її квадратний логотип – саме ті дані, які є пріоритетними для покращення користувацького досвіду при заповненні інформації про місця роботи);

– достатній обсяг даних (база у 17 мільйонів компаній є достатньою для покриття потреб більшості користувачів при заповненні резюме);

– легкість інтеграції(хоча всі API пропонують документацію, фокус TheCompaniesAPI на наданні ключових даних про компанію без надлишкових функцій, орієнтованих на продажі чи маркетинг, може спростити інтеграцію для конкретного завдання ShareCV);

Хоча Clearbit[7] пропонує збагачення в реальному часі та Name to Domain API, який може надавати логотипи, його загальний фокус на ширшому спектрі B2B даних роблять його менш доречним до проєкту. ZoomInfo[8], з його великою базою даних та орієнтацією на великі підприємства, є надлишковим для поставлених завдань.

Отож, TheCompaniesAPI надає необхідний функціонал (назва компанії та логотип) з більш ніж достатнім архівом компаній, що робить його найкращим вибором для інтеграції в ShareCV.

3.2.4 Вибір IDE

Для роботи з клієнтським застосунком на Angular[10] є маса IDE які можна використовувати, одні з них це VSCode та WebStorm. Порівнюючи їх, було обрано VSCode через його простоту, та не навантажений інтерфейс. Хоч і WebStorm має велику кількість вбудованих інструментів для роботи з Angular, VSCode виграє тим, що має дуже велику бібліотеку розширень, за допомогою яких можна кастомізувати IDE навіть більше ніж WebStorm. Також важливо мати на увазі, що WebStorm, як повноцінна IDE використовує достатньо велику кількість оперативної пам’яті, що іноді заважає робити зміни в коді швидко на не самому потужному комп’ютері.

Для роботи з серверними застосунками було обрано продукції компанії JetBrains. Для Java застосунку – IntelliJ Idea, для Python застосунку – PyCharm. На серверній частині це є вибором на користь спеціалізованих інструментів, які оптимізовані конкретно під одну мову та їх екосистему, хоч наприклад VSCode пропонує підтримку цих мов, але спеціалізовані IDE дають більш глибоку інтеграцію, кращу продуктивність та більш продуманий набір функцій для конкретної мови.

3.3 Конструювання програмного забезпечення

В таблиці 3.2 детально описано компоненти клієнтської частини застосунку, та їх використання, також до таблиці було додано перевикористовані компоненти, що були створені для підтримання чистоти коду.

Таблиця 3.2 – Опис компонентів фронтенд застосунку

Назва	Опис
AppComponent	Головний компонент застосунку, який слугує кореневим контейнером для всіх інших представлень.
AuthLayoutComponent	Компонент для відображення макету сторінок автентифікації.
FooterComponent	Компонент для відображення нижнього колонтитула (футера) застосунку.
HeaderComponent	Компонент для відображення верхнього колонтитула (хедера) застосунку.
MainLayoutComponent	Компонент для відображення основного макету сторінок застосунку після автентифікації.
ShellComponent	Компонент-оболонка, що може містити основну структуру сторінки.
ToasterComponent	Компонент для відображення спливаючих повідомлень (тостів).
UploadResumePopupComponent	Компонент для спливаючого вікна завантаження резюме.
AdminTableComponent	Компонент таблиці для відображення даних в адмін-панелі.
CompanyStatCardComponent	Компонент картки статистики компанії на головній сторінці.

Продовження таблиці 3.2

StatCardComponent	Компонент картки статистики на головній сторінці.
StatisticsSectionComponent	Компонент секції статистики на головній сторінці.
TopSectionComponent	Компонент верхньої секції на головній сторінці.
ResumeDetailComponent	Компонент для детального перегляду резюме.
ResumeFilterComponent	Компонент для фільтрації резюме.
ResumeTableComponent	Компонент для відображення списку резюме у вигляді таблиці.
ButtonComponent	Компонент кастомної кнопки для повторного використання.
ChipsComponent	Компонент 'чіпсів' для повторного використання.
CompanyAutocompleteComponent	Компонент автозаповнення для вибору компанії.
DropdownComponent	Компонент випадаючого списку для повторного використання.
IconComponent	Компонент для відображення іконок.
InputComponent	Компонент кастомного поля вводу для повторного використання.
PaginatorComponent	Компонент пагінації для таблиць або списків.
PopupComponent	Компонент для створення спливаючих вікон (попапів).
TableComponent	Компонент кастомної таблиці для повторного використання.

3.3.1 Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання користувачів, даних про резюме та компанії. Опис таблиць бази даних наведено у таблицях 3.3-3.10.

Таблиця 3.3 – Опис таблиці users

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер користувача.
email	VARCHAR	Електронна пошта користувача.

Продовження таблиці 3.3

password	VARCHAR	Захешований пароль до облікового запису.
nick	VARCHAR	Пошта чи нік користувача.
role	VARCHAR	Роль користувача в системі.

Таблиця 3.4 – Опис таблиці resumes

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер резюме.
AUTHOR_ID	UUID	Ідентифікатор автора резюме.
createdAt	TIMESTAMP	Дата та час створення резюме.
yearsOfExperience	INT	Кількість років досвіду.
speciality	VARCHAR	Спеціалізація.
isHidden	BOOLEAN	Прапорець, що показує чи приховане резюме.
status	VARCHAR	Статус резюме.

Таблиця 3.5 – Опис таблиці companies

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер компанії.
name	VARCHAR	Назва компанії.
logoUrl	VARCHAR	URL-адреса логотипу компанії.

Таблиця 3.6 – Опис таблиці resumes_companies

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер запису зв'язку.
resume_id	UUID	Ідентифікатор резюме.
company_id	UUID	Ідентифікатор компанії.
isHrScreeningPassed	BOOLEAN	Прапорець, що показує чи пройдено HR скринінг у цій компанії для резюме.

Таблиця 3.7 – Опис таблиці comments

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер коментаря.
resume_id	UUID	Ідентифікатор резюме, до якого належить коментар.
parentCommentId	UUID	Ідентифікатор батьківського коментаря.
reactionsRate	INT	Рейтинг реакцій на коментар.
message	TEXT	Текст коментаря.
createdAt	TIMESTAMP	Дата та час створення коментаря.

Таблиця 3.8 – Опис таблиці users_favorites_resumes

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер запису зв'язку.
user_id	UUID	Ідентифікатор користувача.
resume_id	UUID	Ідентифікатор резюме.

Таблиця 3.9 – Опис таблиці user_comment_vote_state

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер запису.
user_id	UUID	Ідентифікатор користувача.
comment_id	UUID	Ідентифікатор коментаря.
voteState	VARCHAR	Стан голосу.

Таблиця 3.10 – Опис таблиці documents

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер документа.
accessType	VARCHAR	Тип доступу до документа.
resume_id	UUID	Ідентифікатор резюме.
name	VARCHAR	Назва файлу документа.
directory	VARCHAR	Шлях до файлу в сховищі.

Схема бази даних наведена у графічних матеріалах.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.11.

Таблиця 3.11 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Середовище Java розробки програмного забезпечення серверної частини курсової роботи.
2	Postman	Програмне забезпечення необхідне для тестування API запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
3	Datagrip	Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних.
4	VSCode	Середовище розробки програмного забезпечення клієнтської частини курсової роботи.
5	PyCharm	Середовище Python розробки програмного забезпечення серверної частини курсової роботи.
6	Presidio Analyzer[13]	Python бібліотека для виявлення персональних даних в тексті.

3.4 Аналіз безпеки даних

У застосунку ShareCV безпека даних є дуже важливою частиною розробки, так як застосунок зберігає персональні дані користувачів. Було зроблено як умога більше, щоб забезпечити безпечне збереження та обмін даними.

З приводу паролів користувачів до системи – застосунок використовує хешування для скриття паролів, та при можливому витіканні даних зловмисники не зможуть отримати доступи до аккаунтів.

Усі шляхи API(окрім пов’язаних з авторизацією) полягає в наступному: використання JWT токену для доступу до API; розмежування ролей та використання різних JWT токенів для них. Таким чином, звичайний

користувач, навіть маючи JWT токен, не зможе використовувати API які були розроблені для адміністратора.

З приводу JWT токена авторизації був зроблений динамічний варіант аутентифікації, час придатності токена був встановлений на 20 хвилин, а після виходу з придатності він автоматично оновлюється системою.

Для убезпечення середовища в хмарі усі контейнери було об’єднано в єдину Docker[14] мережу яка не доступна для публічного доступу, що робить неможливим для зловмисників робити запити до серверів з інших місць, окрім фронтенд застосунку. Публічним є тільки контейнер сховища документів та база даних. Для доступу з клієнтського застосунку було використано NGINX[9], що перенаправляє запити ззовні й переправляє їх до локальної мережі.

Для безпечного доступу до файлів у сховищі документів Minio[12], кожен раз коли відбувається запит будь-якого документу користувачу Minio генерує тимчасове посилання для доступу до документу на 15 хвилин.

Доступ до бази даних здійснюється за допомогою логіну та паролю.

Взаємодія між Java та Python сервісами також є безпечним, так як Python сервіс приймає запити тільки з секретним ключем, що має Java сервіс.

Взаємодія між Java сервісом та зовнішнім сервісом компанії CompaniesAPI[6] також зашифрована за допомогою секретного ключа доступу до Java сервісу.

Вразливість системи полягає в тому, що при отриманні доступу до посилання документу миттєво зловмисником, то в нього буде невеликий час для його викрадення. Додатково йому потрібен доступ до комп’ютера користувача, що вже не є можливим для регулювання зі сторони розробки ShareCV.

Висновки до розділу

У розділі спроєктовано архітектуру веб-застосунку ShareCV, обґрунтовано вибір ключових технологій на засобів для розробки.

Веб-застосунок побудовано на основі незалежний монолітних сервісів, які розгортаються в межах єдиного середовища, що дозволяє зробити розгортання на ранніх етапах простим, але в той же час надає можливість для маштабування сервісів в незалежності один від одного та полегшує тестування.

Для реалізації функціональності автоматичного підвантаження назв компаній та їх логотипів було інтегровано сервіс CompaniesAPI[6].

Підчас розробки були використані наступні IDE, в залежності від частини застосунку: для серверної частини використовувалися IntelliJ IDEA та PyCharm, що забезпечують максимальну інтеграцію з відповідними мовами; Для фронтенд застосунку було використано VSCode.

Розроблено архітектуру веб-застосунку ShareCV, яка відповідає функціональним на нефункціональним вимогам.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Мета аналізу якості веб-застосунків полягає у забезпеченні відповідності між розробленим продуктом та встановленими стандартами, виявленням потенційних проблем у безпеці та архітектурі та оцінка готовності системи до роботи з користувачами.

Для аналізу якості дипломного проєкту ShareCV було використано інструмент codequality.browserstack.com[15].

Для оцінки якості ПЗ було обрано наступні метрики:

– вразливості (потенційні слабкі місця в коді, які можуть бути використані зловмисниками);

– анти-патерни (поширені, але неефективні або не продуктивні підходи до розв'язання проблем програмування);

– дублювання коду (повторювані фрагменти коду, що ускладнюють підтримку та рефакторинг);

– загальний рейтинг (комплексна оцінка якості коду);

На рисунках 4.1-4.4 показано результати статичного аналізу коду інструментом codequality.browserstack.com[15].

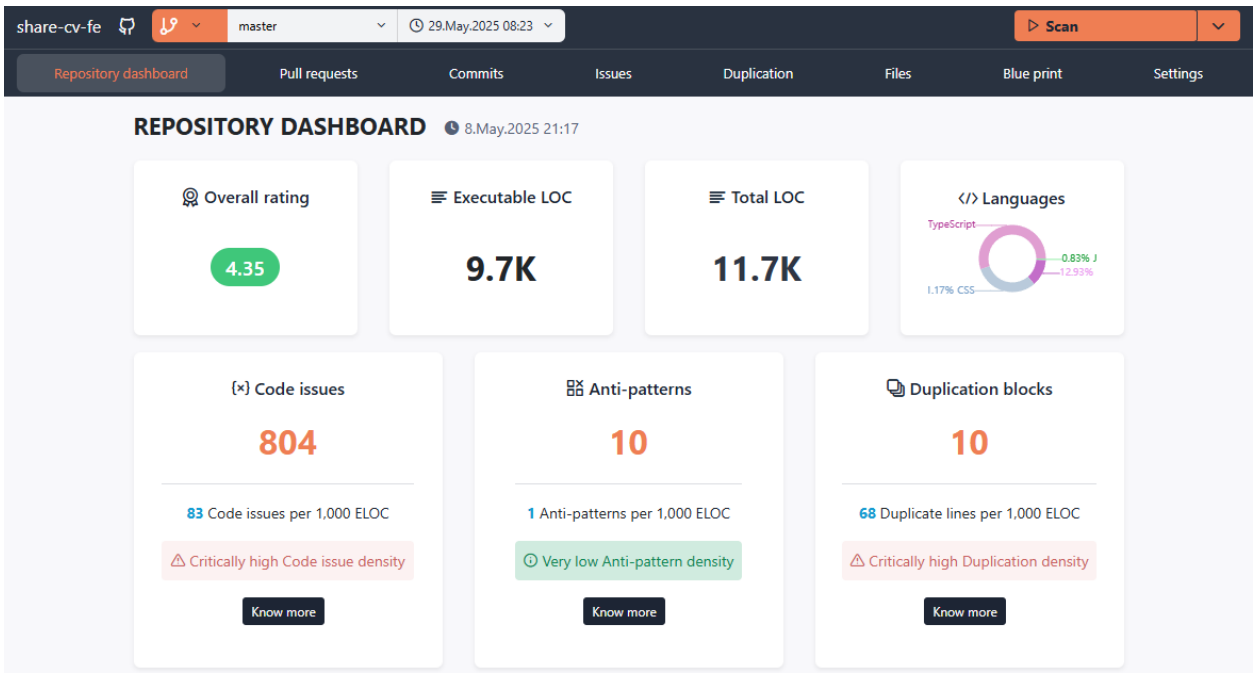


Рисунок 4.1 – Аналіз клієнтського застосунку

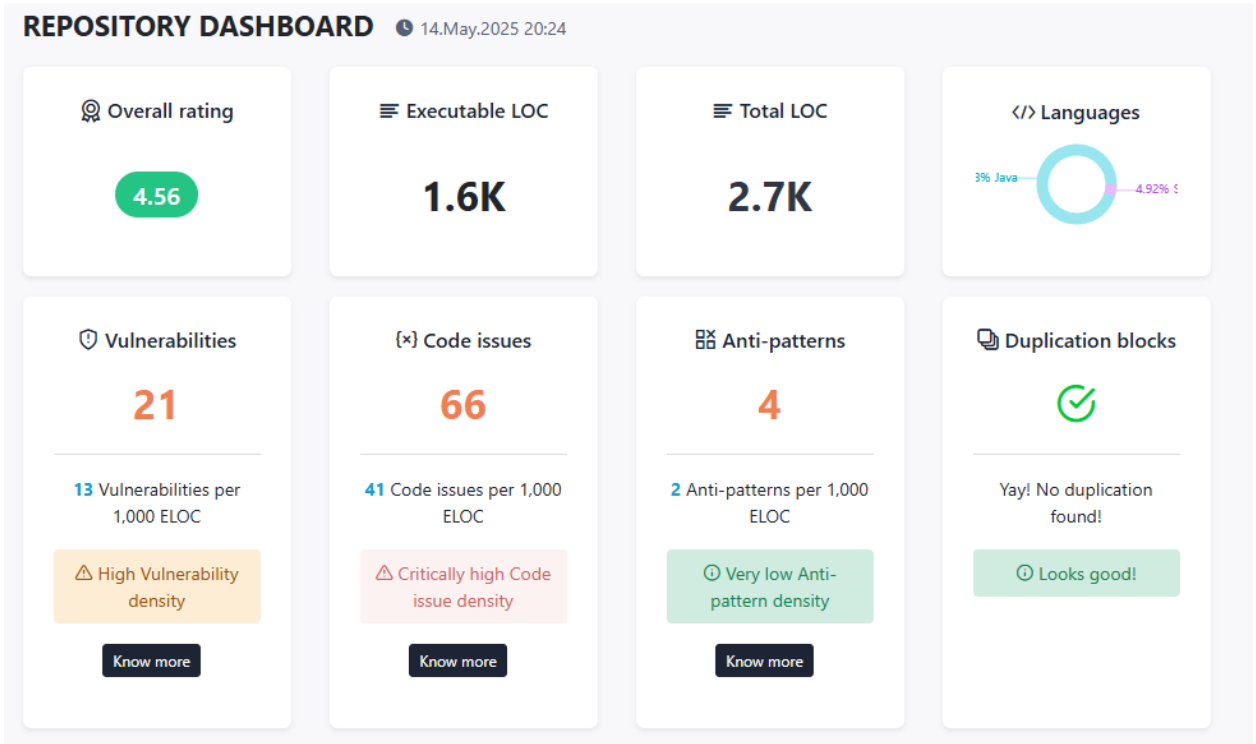


Рисунок 4.2 – Аналіз серверного застосунку

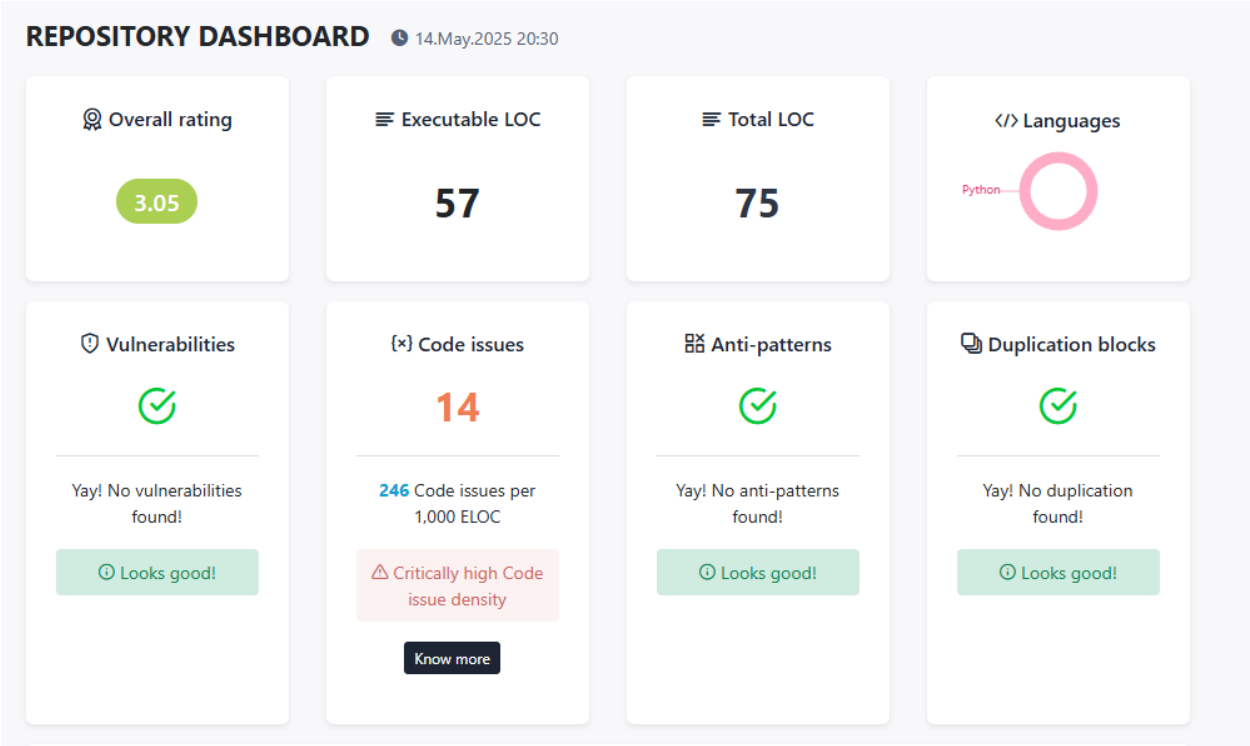


Рисунок 4.3 – Аналіз сервісу прибирання персональних даних

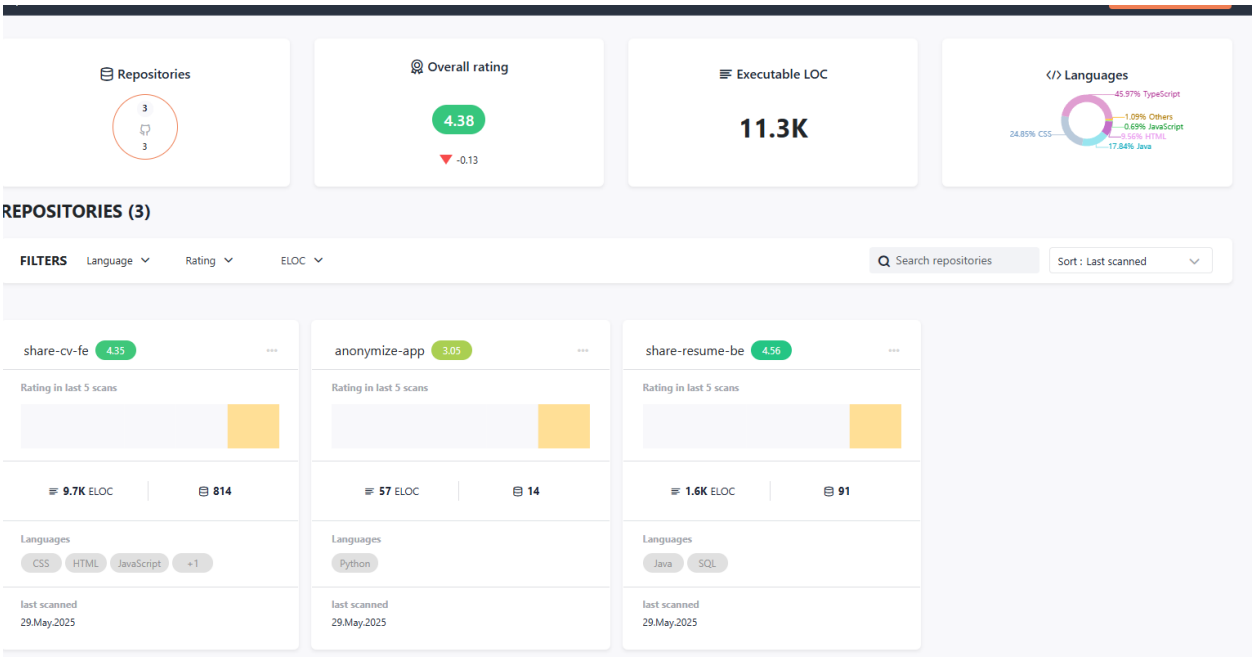


Рисунок 4.4 – Загальний рейтинг застосунку ShareCV

Проаналізувавши метрики можна в першу чергу побачити велику кількість проблем в коді, близько 800. Але так як сервіс, що аналізував застосунок не повністю заточений під фреймворк Angular[10], то дуже часто застосовує загальні правила коду до файлів Angular. Наприклад на рисунку 4.5 можна побачити помилку у HTML файлах про необхідність використання тегу

<!DOCTYPE> на першому ж рядку. Через те, що Angular є одно сторінковим фреймворком це не є можливим й кожен компонент описуючи структуру шаблону не має використовувати вищезазначений тег.

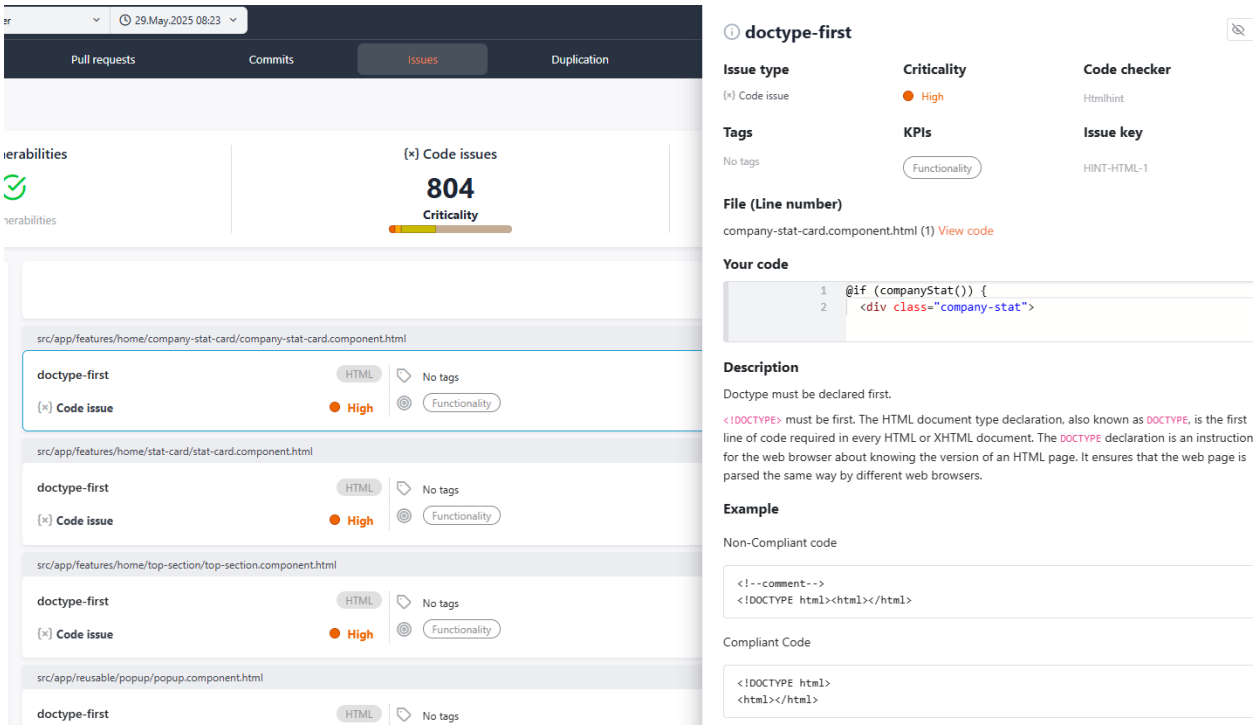


Рисунок 4.5 – Детальний опис помилки у клієнтської застосунку

Таких проблем у коді було зазначено дуже багато, тому реальна кількість проблем у коді є значно меншою. Але навіть при таких результатах загальний рейтинг клієнтського застосунку склав 4.35 по шкалі від -5 до 5, що є більш ніж задовільно.

Дивлячись на результати інших частин застосунку та загальний рейтинг можна свідчити про те, що код не є ідеальним й має місця для доробок такі як вразливості у серверного сервісу чи проблеми у коді у сервісі прибирання персональних даних, але загалом система має якість коду вище ніж середня.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.15.

Таблиця 4.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 8 символів, який містить одне число і один спеціальний символ, мінімум одну літеру верхнього регістру, мінімум одну літеру нижнього регістру; підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації. Електронна пошта `test@example.com` вже зареєстрована в системі.
Вхідні дані	Електронна пошта: `test@example.com`, Пароль: `ValidPassword1!`, Підтвердження паролю: `ValidPassword1!`
Опис проведення тесту	Користувач вводить у відповідні поля електронну пошту, яка вже існує в системі, валідний пароль та його підтвердження. Натискає кнопку підтвердження реєстрації.
Очікуваний результат	Система відображає повідомлення про помилку, що користувач з такою електронною поштою вже існує. Реєстрація не відбувається. Користувач залишається на сторінці реєстрації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта: `newuser@example.com`, Пароль: `ValidPassword1!`, Підтвердження паролю: `DifferentPassword1!`
Опис проведення тесту	Користувач вводить у відповідні поля нову електронну пошту, валідний пароль, але в поле підтвердження паролю вводить інший пароль. Натискає кнопку підтвердження реєстрації.
Очікуваний результат	Система відображає повідомлення про помилку, що паролі не співпадають. Поле підтвердження паролю підсвічується. Реєстрація не відбувається. Користувач залишається на сторінці реєстрації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 2.1 Успішна авторизація користувача

Початковий стан системи	Користувач `user@example.com` з паролем `ValidPassword1!` зареєстрований в системі. Користувач знаходиться на сторінці авторизації.
Вхідні дані	Електронна пошта: `user@example.com`, Пароль: `ValidPassword1!`
Опис проведення тесту	Користувач вводить коректні зареєстровані електронну пошту та пароль у відповідні поля. Натискає кнопку "Увійти".
Очікуваний результат	Авторизація проходить успішно. Користувач перенаправляється на головну сторінку застосунку для авторизованих користувачів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 2.2 Авторизація користувача

Початковий стан системи	Користувач `user@example.com` зареєстрований в системі. Користувач знаходиться на сторінці авторизації.
Вхідні дані	Електронна пошта: `user@example.com`, Пароль: `InvalidPassword1!`
Опис проведення тесту	Користувач вводить коректну зареєстровану електронну пошту та невірний пароль у відповідні поля. Натискає кнопку "Увійти".

Продовження таблиці 4.5

Очікуваний результат	Система відображає повідомлення про помилку "Невірна електронна пошта або пароль". Авторизація не відбувається. Користувач залишається на сторінці авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 3.1 Успішне створення та відправка резюме на модерацію

Початковий стан системи	Користувач авторизований в системі та знаходиться на сторінці, де є можливість додати резюме (наприклад, головна сторінка).
Вхідні дані	Заповнені поля форми створення резюме (наприклад, компанія, посада, роки досвіду), PDF файл резюме.
Опис проведення тесту	Користувач натискає кнопку "Додати резюме". Заповнює всі необхідні поля у формі, завантажує PDF файл резюме. Натискає кнопку "Відправити".
Очікуваний результат	Резюме успішно додається до системи зі статусом "очікує модерації". Користувач отримує сповіщення про успішну відправку. Резюме з'являється у списку його відправлених резюме та у списку резюме на модерацію для адміністратора.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 3.2 Спроба створення резюме без завантаження PDF файлу

Початковий стан системи	Користувач авторизований в системі. Відкрито форму створення резюме.
Вхідні дані	Заповнені текстові поля форми створення резюме, PDF файл резюме не завантажено.
Опис проведення тесту	Користувач заповнює всі необхідні текстові поля у формі, але не завантажує PDF файл резюме. Натискає кнопку "Відправити".
Очікуваний результат	Система відображає повідомлення про помилку, що необхідно завантажити PDF файл резюме. Поле для завантаження файлу підсвічується. Резюме не відправляється на модерацію.

Продовження таблиці 4.7

Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 4.1 Перегляд списку опублікованих резюме

Початковий стан системи	Користувач авторизований в системі. В системі є декілька опублікованих резюме (після модерації). Користувач переходить на сторінку "Resumes".
Вхідні дані	Немає (перегляд загального списку).
Опис проведення тесту	Користувач відкриває сторінку зі списком опублікованих резюме.
Очікуваний результат	Відображається список резюме. Кожен рядок містить основну інформацію (наприклад, назва компанії, спеціальність, кількість років досвіду, кількість схвалень).
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 4.2 Пошук резюме за назвою компанії

Початковий стан системи	Користувач авторизований в системі та знаходиться на сторінці перегляду списку резюме. В системі є опубліковані резюме для компанії "Tech Solutions".
Вхідні дані	Фільтр "Компанія": "Tech Solutions".
Опис проведення тесту	Користувач вводить "Tech Solutions" у поле фільтра "Компанія" та обирає зі списку та застосовує фільтр.
Очікуваний результат	У списку відображаються тільки ті резюме, які були подані до компанії "Tech Solutions" і пройшли модерацію. Якщо таких резюме немає, відображається відповідне повідомлення.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 5.1 Перегляд детальної інформації резюме

Початковий стан системи	Користувач авторизований. Є опубліковане резюме. Користувач знаходиться на сторінці списку резюме.
Вхідні дані	Клік на конкретне резюме зі списку.

Продовження таблиці 4.10

Опис проведення тесту	Користувач обирає та натискає на одне з резюме у списку.
Очікуваний результат	Відкривається сторінка з детальною інформацією про обране резюме: повний опис, компанії, статуси, спеціалізація. Вбудований переглядач відображає обфускований PDF документ резюме. Є секція для коментарів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 6.1 Додавання коментаря до резюме

Початковий стан системи	Користувач авторизований та знаходиться на сторінці детального перегляду опублікованого резюме.
Вхідні дані	Текст коментаря: "Дуже інформативне резюме! Можливо, варто додати більше деталей про проект X?"
Опис проведення тесту	Користувач вводить текст коментаря у відповідне поле під деталями резюме та натискає кнопку "Надіслати" (або "Додати коментар").
Очікуваний результат	Коментар успішно додається та відображається у секції коментарів під резюме. Інші користувачі можуть бачити цей коментар.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 7.1 Додавання резюме до обраних

Початковий стан системи	Користувач авторизований та знаходиться на сторінці детального перегляду опублікованого резюме, яке ще не додане до обраних.
Вхідні дані	Клік на кнопку "Додати до обраних" (або "Bookmark").
Опис проведення тесту	Користувач натискає на кнопку для додавання резюме до обраних.
Очікуваний результат	Резюме успішно додається до списку обраних користувача. Іконка/кнопка змінює свій вигляд, показуючи, що резюме додано (наприклад, стає заповненою або змінює текст).
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 8.1 Перегляд списку резюме, що очікують на модерацію адміністратором

Початковий стан системи	Користувач з правами адміністратора авторизований в системі. Є кілька резюме, які були відправлені користувачами та очікують на модерацію. Адміністратор знаходиться на "Admin Dashboard".
Вхідні дані	Перехід до розділу модерації резюме "Admin Dashboard".
Опис проведення тесту	Адміністратор переглядає список резюме, які мають статус "очікує модерації" або аналогічний.
Очікуваний результат	Відображається список резюме, що очікують на модерацію. Для кожного резюме вказана основна інформація (хто подав, коли, назва компанії тощо) та є опції для модерації (затвердити/відхилити, переглянути деталі).
Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 8.2 Затвердження резюме адміністратором

Початковий стан системи	Адміністратор авторизований та знаходиться на сторінці перегляду деталей резюме, що очікує на модерацію. Всі дані резюме коректні, анонімізація пройшла успішно.
Вхідні дані	Клік на кнопку "Затвердити" (або "Approve").
Опис проведення тесту	Адміністратор перевіряє резюме та натискає кнопку "Затвердити".
Очікуваний результат	Статус резюме змінюється на "Опубліковано" (або "Approved"). Резюме стає видимим для всіх авторизованих користувачів у загальному списку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.15 – Тест 8.3 Відхилення резюме адміністратором

Початковий стан системи	Адміністратор авторизований та знаходиться на сторінці перегляду деталей резюме, що очікує на модерацію. Резюме містить неприйнятний контент або помилки.
Вхідні дані	Клік на кнопку "Відхилити" (або "Reject").

Продовження таблиці 4.15

Опис проведення тесту	Адміністратор перевіряє резюме, знаходить причину для відхилення та натискає кнопку "Відхилити".
Очікуваний результат	Статус резюме змінюється на "Відхилено" (або "Rejected"). Резюме не стає видимим для користувачів.
Фактичний результат	Збігається з очікуваним.

4.3 Опис контрольного прикладу

Для опису контрольного прикладу було обрано реєстрацію на авторизацію, процес відправки резюме, його публікації у застосунку адміністратором, пошук резюме за критеріями, написання коментаря та видалення облікового запису користувача.

Виконано перехід до реєстрації й введено некоректний логін(рис 4.6).

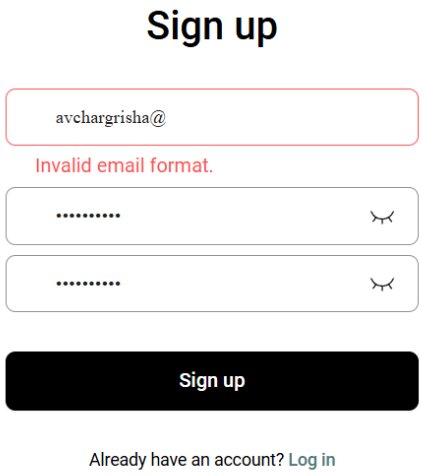


Рисунок 4.6 – Реєстрація користувача з некоректною поштою

Після корегування логіну, було введено некоректний пароль(рис 4.7).

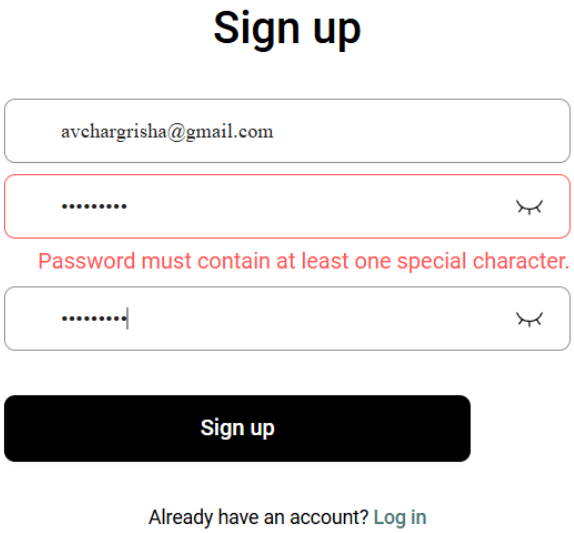


Рисунок 4.7 – Реєстрація з некоректним паролем

Введено не однаковий повтор паролю й після виправлено(рис 4.8).

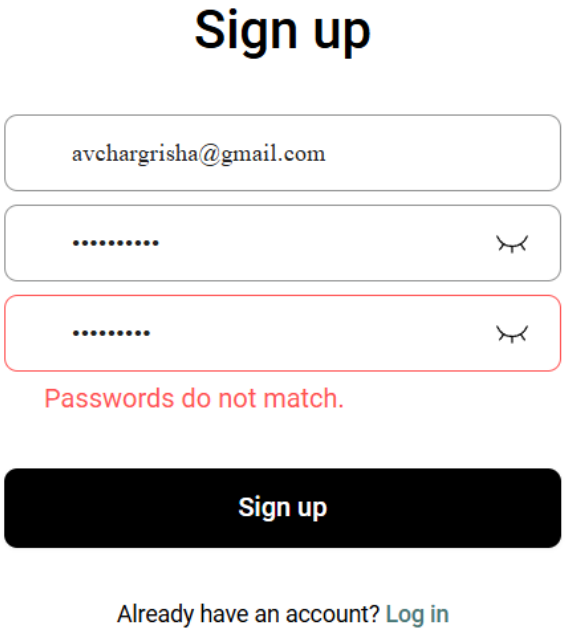


Рисунок 4.8 – Реєстрація з неспівпадаючим паролем

Після коректної реєстрації, введено дані користувача у форму входу(рис 4.9).

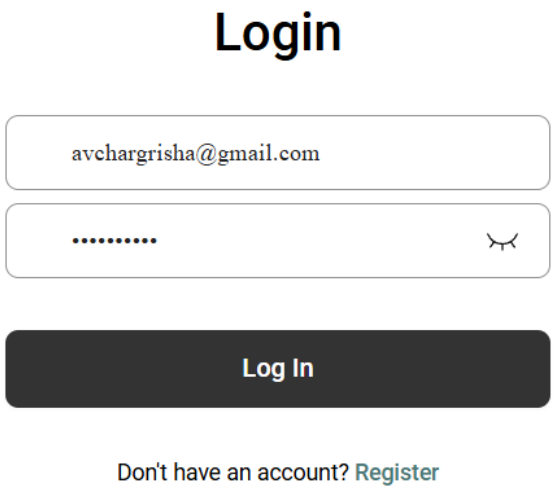


Рисунок 4.9 – Авторизація користувача

Користувача було перенаправлено на головну сторінку(рис 4.10).

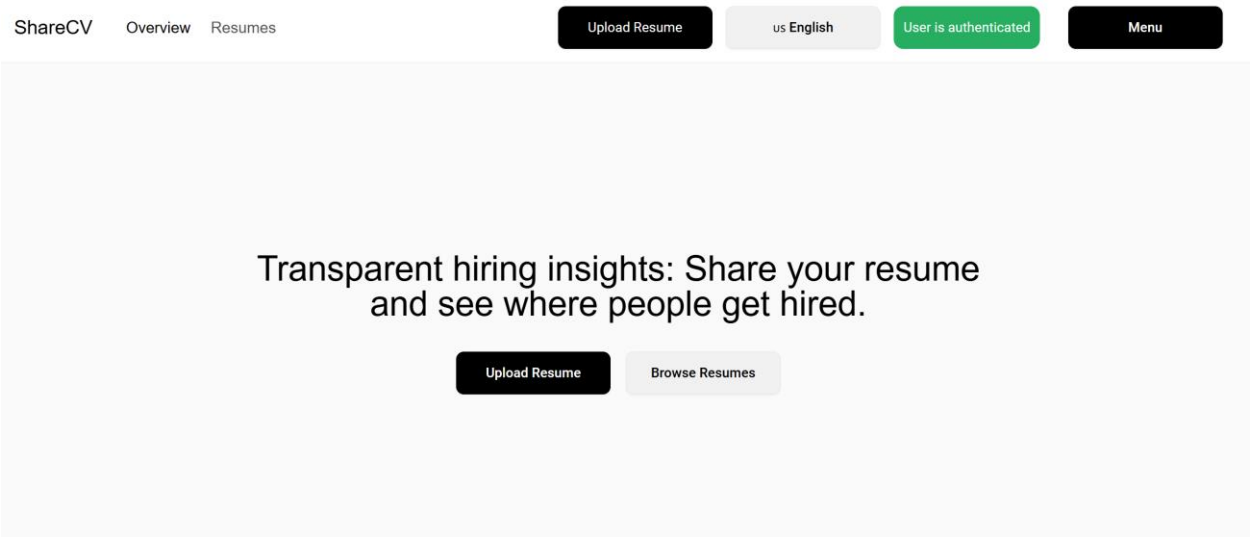


Рисунок 4.10 – Перехід на головну сторінку

Відкрито форму відправки резюме й введено відповідні дані(рис 4.11).

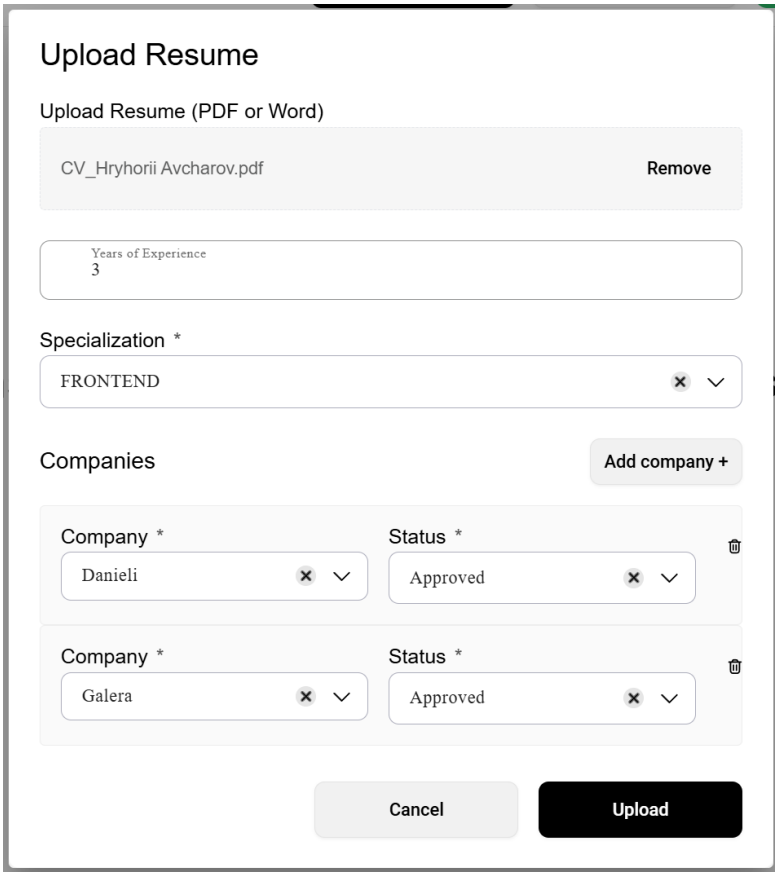


Рисунок 4.11 – Завантаження резюме користувачем

Після відправки резюме отримано сповіщення про успішну відправку(рис 4.12).

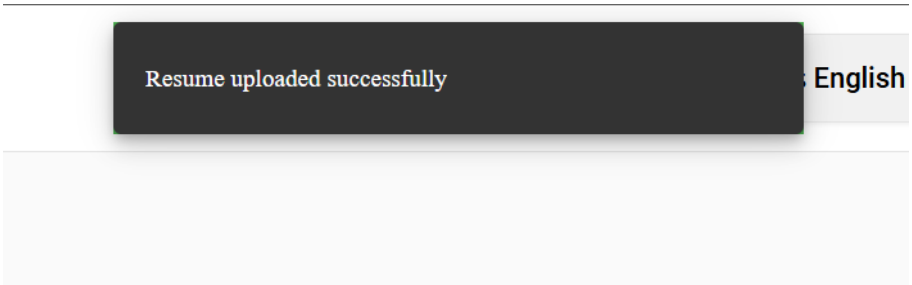


Рисунок 4.12 – Успішне додавання резюме до застосунку

Відкрито випадаючий список «Меню» для виходу з облікового запису(рис 4.13).

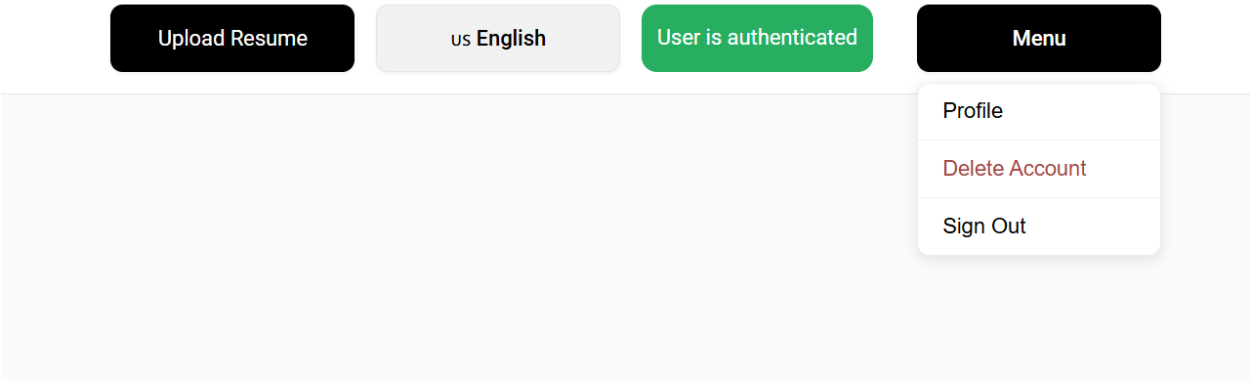


Рисунок 4.13 – Вихід з облікового запису

Авторизовано за допомогою даних адміністратора(рис 4.14).

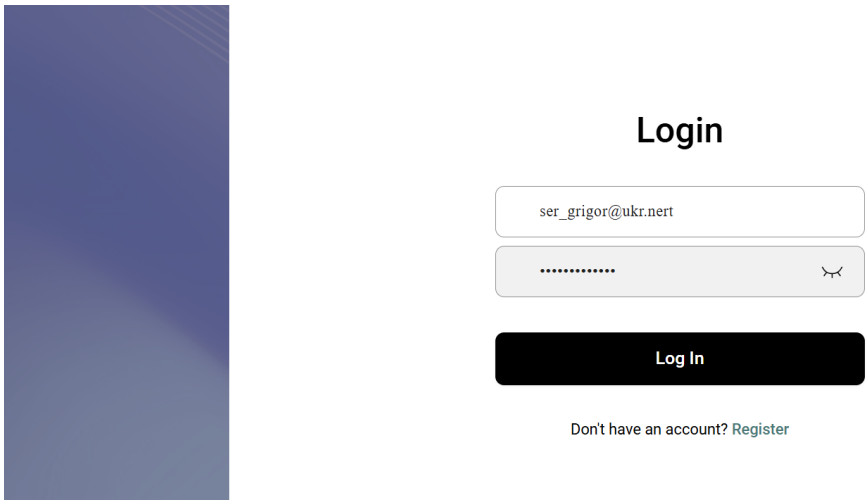


Рисунок 4.14 – Авторизація адміністратора

Після авторизації адміністратору візуалізовано список резюме поданих користувачами(рис 4.15).

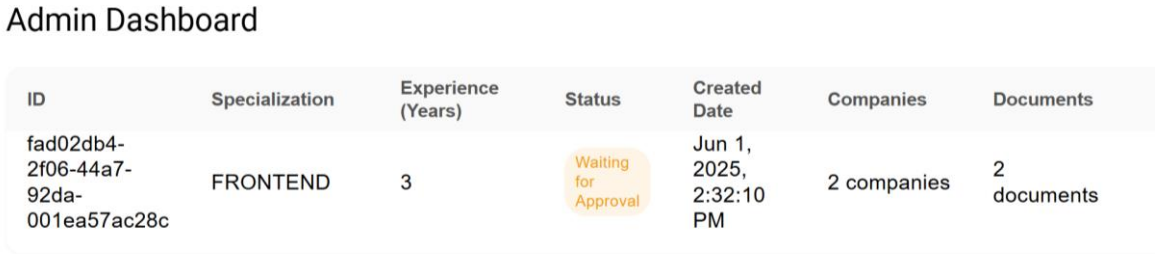


Рисунок 4.15 – Перегляд поданих резюме адміністратором

Після відкриття резюме, показано детальну інформацію про нього(рис 4.16).

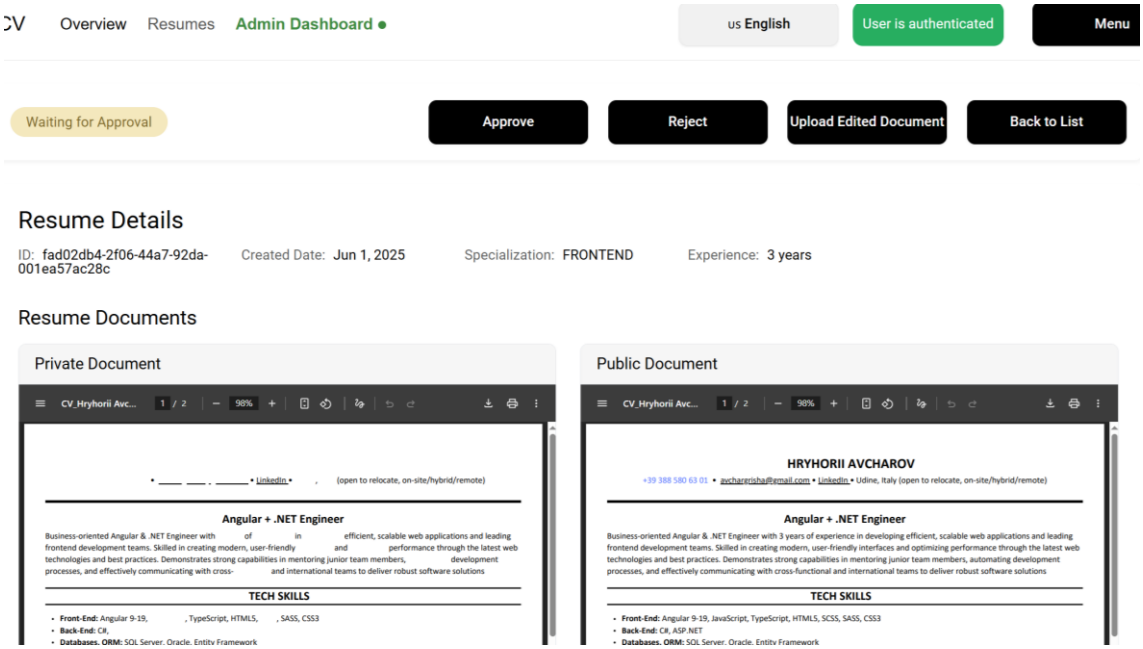


Рисунок 4.16 – Перегляд конкретного резюме адміністратором

Після верифікації документів адміністратором, він змінює статус резюме (рис 4.17).

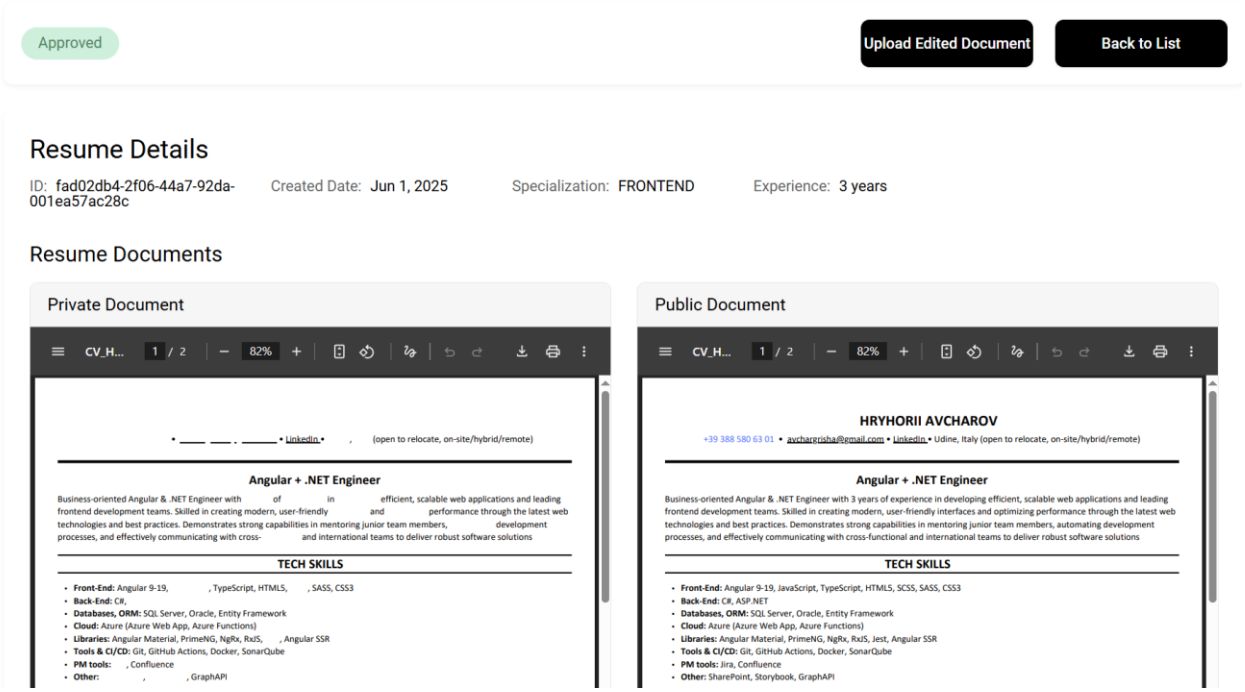


Рисунок 4.17 – Підтвердження статусу резюме адміністратора

Для перевірки було завантажено друге резюме тим же користувачем(рис 4.18).

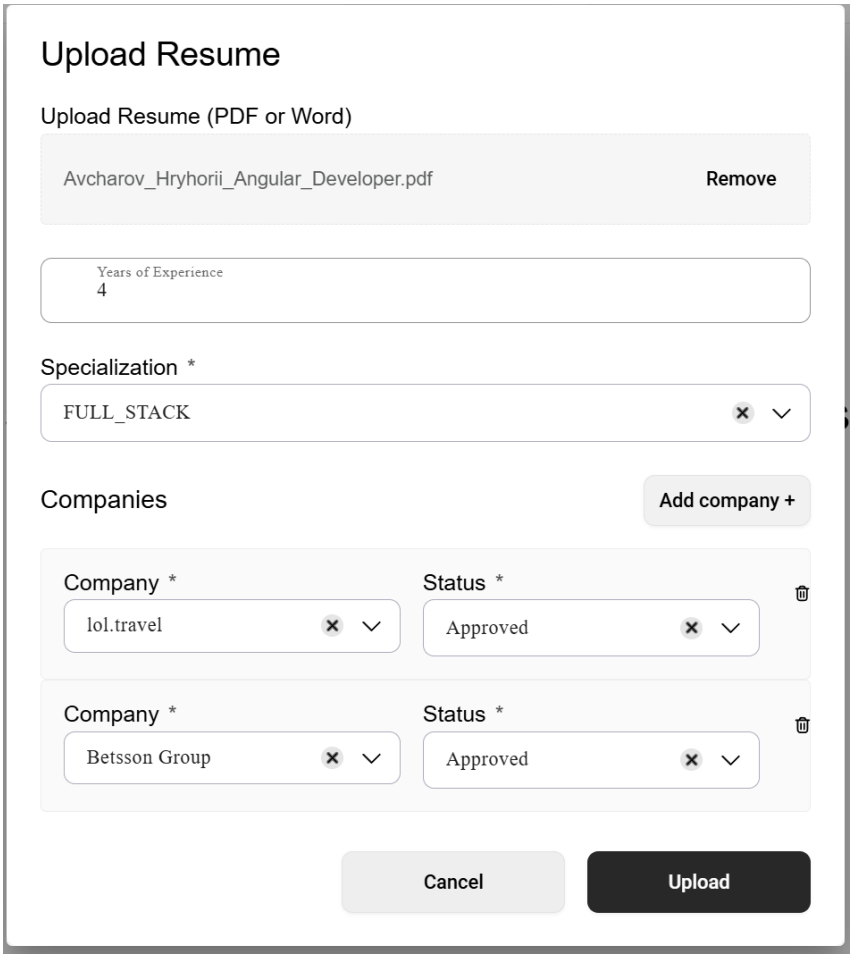


Рисунок 4.18 – Подання другого резюме користувачем

Друге подане резюме у таблиці адміністратора(рис 4.19).

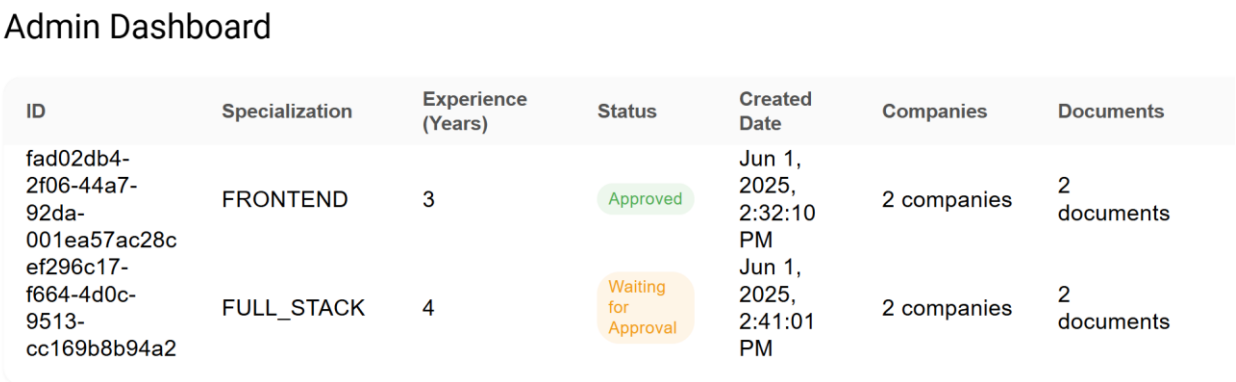


Рисунок 4.19 – Перегляд списку поданих резюме адміністратором

Адміністратор здійснює підтвердження другого резюме(рис 4.20).

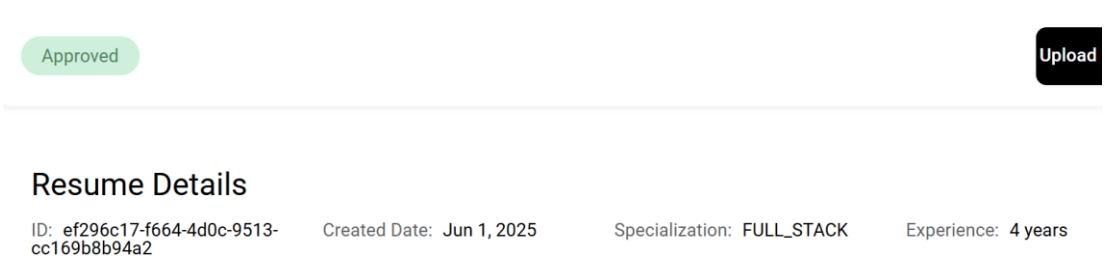


Рисунок 4.20 – Підтвердження статусу другого резюме

Перегляд обох підтверджених резюме у публічному доступі(рис 4.21).

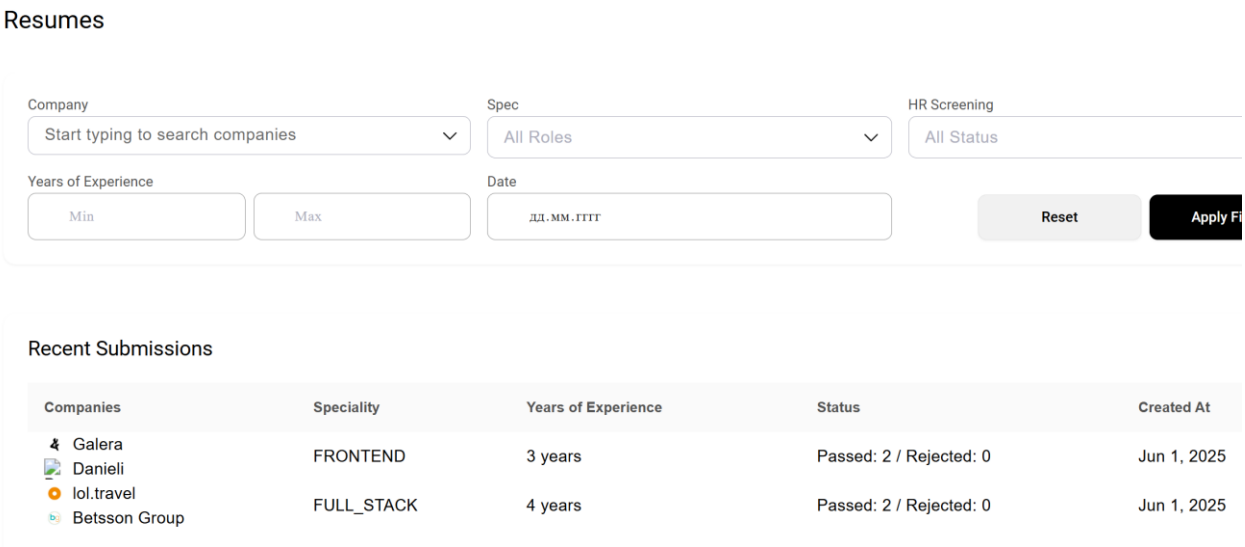


Рисунок 4.21 – Перегляд публічних резюме користувачем

Пошук компанії для фільтрації резюме(рис 4.22).

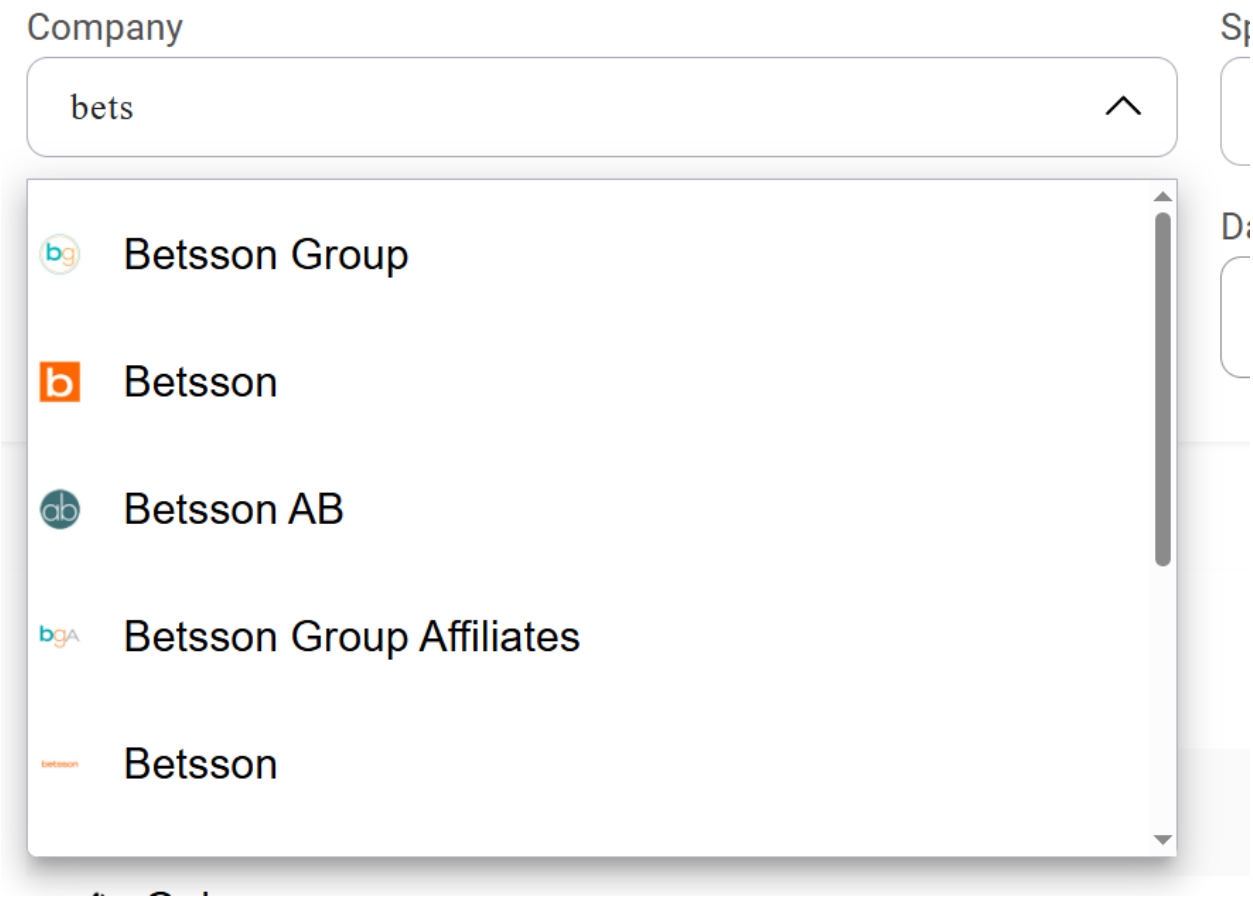


Рисунок 4.22 – Пошук компанії по фразі у фільтрах резюме

Перегляд результату фільтрації за введеними фільтрами(рис 4.23).

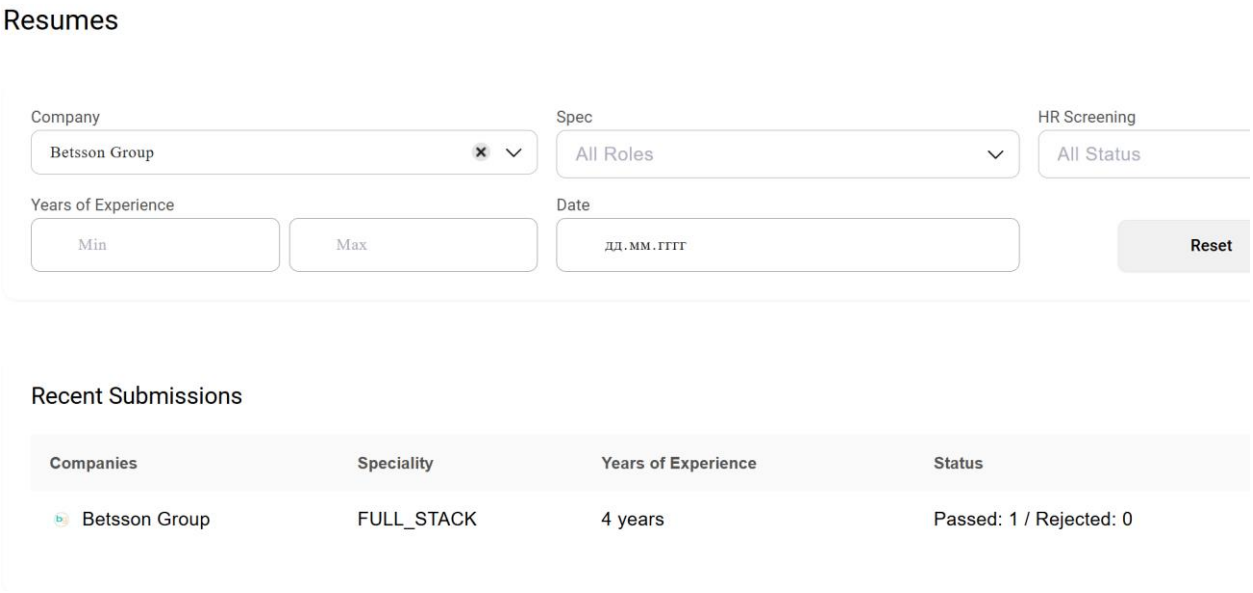


Рисунок 4.23 – Фільтрація резюме за введеними даними

Натискаючи на резюме зроблено перехід до детального вигляду резюме, й написано коментар до нього(рис 4.24).

Рисунок 4.24 – Написання коментаря до резюме

Після написання коментар було опубліковано(рис 4.25).

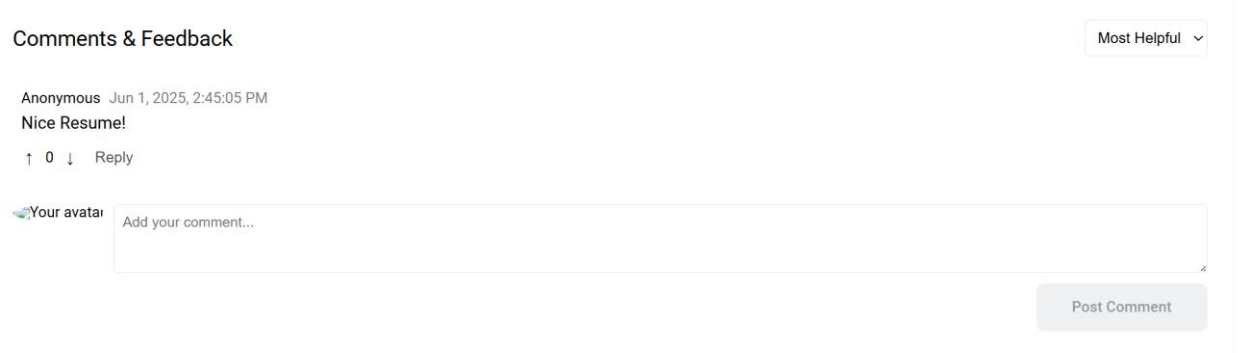


Рисунок 4.25 – Публікація коментаря

Для видалення облікового запису було відкрито меню, й настито кнопку «Delete Account» (рис 4.26).

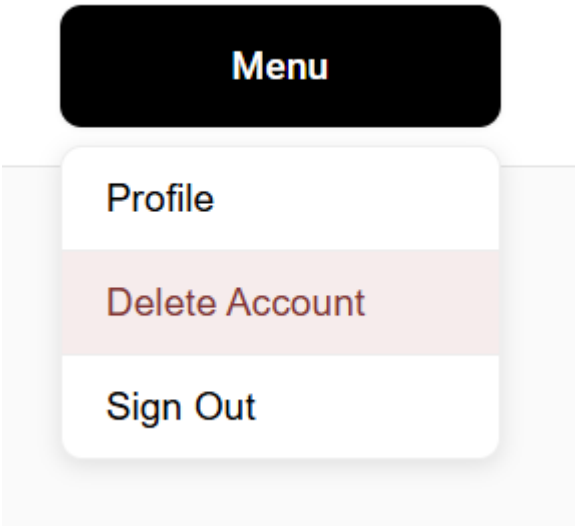


Рисунок 4.26 – Випадаючий список "Меню"

Після було підтверджено видалення облікового запису(рис 4.27).

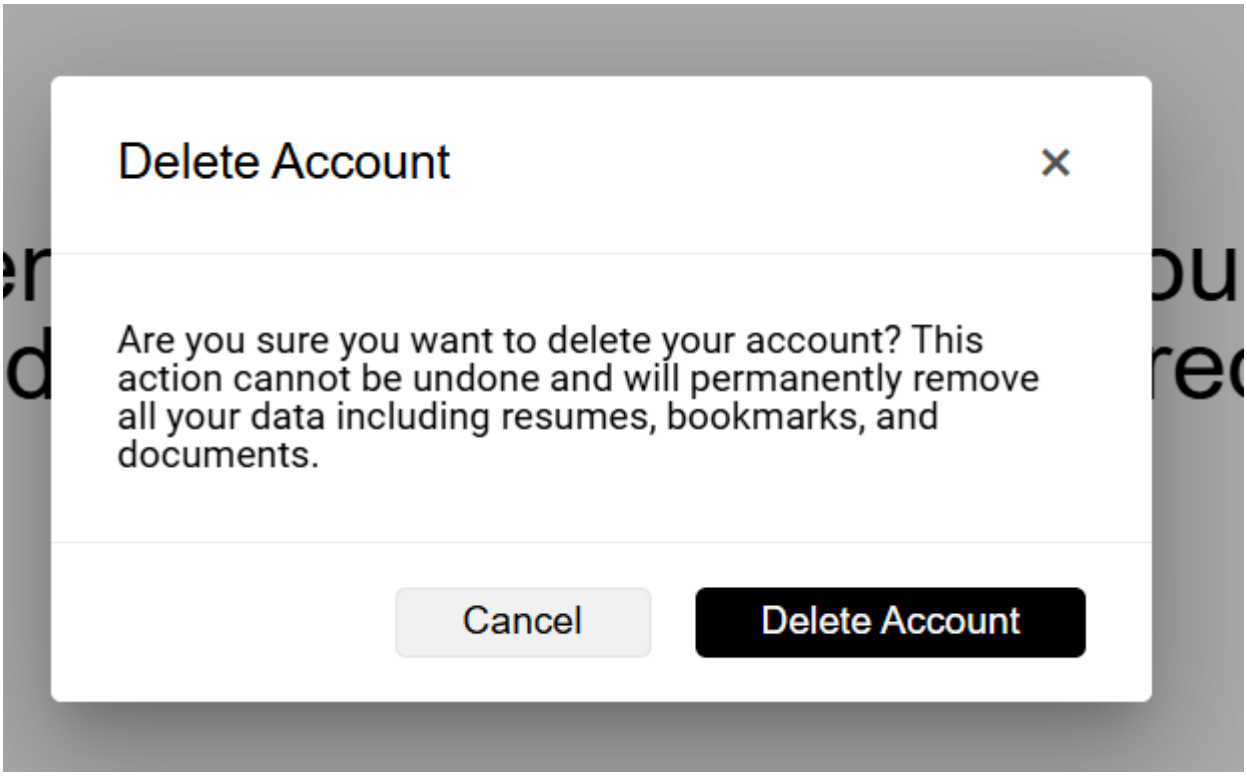


Рисунок 4.27 – Видалення облікового запису

Отримано повідомлення про успішне видалення даних(рис 4.28).

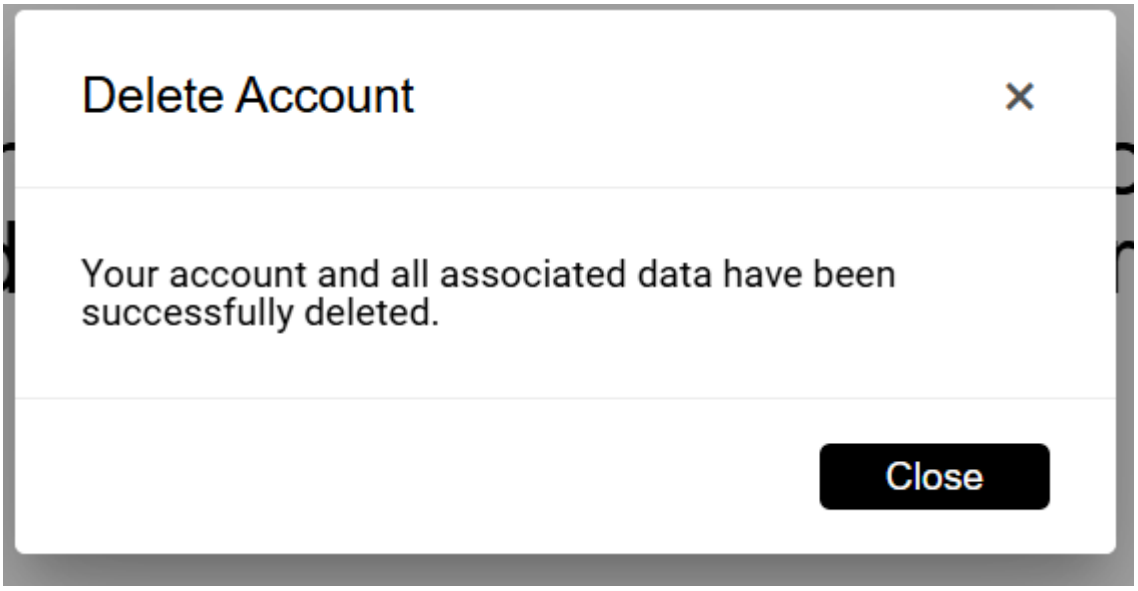


Рисунок 4.28 – Результат видалення облікового запису й даних

Після видалення облікового запису, резюме на інші дані зникнули з застосунку(рис 4.29).

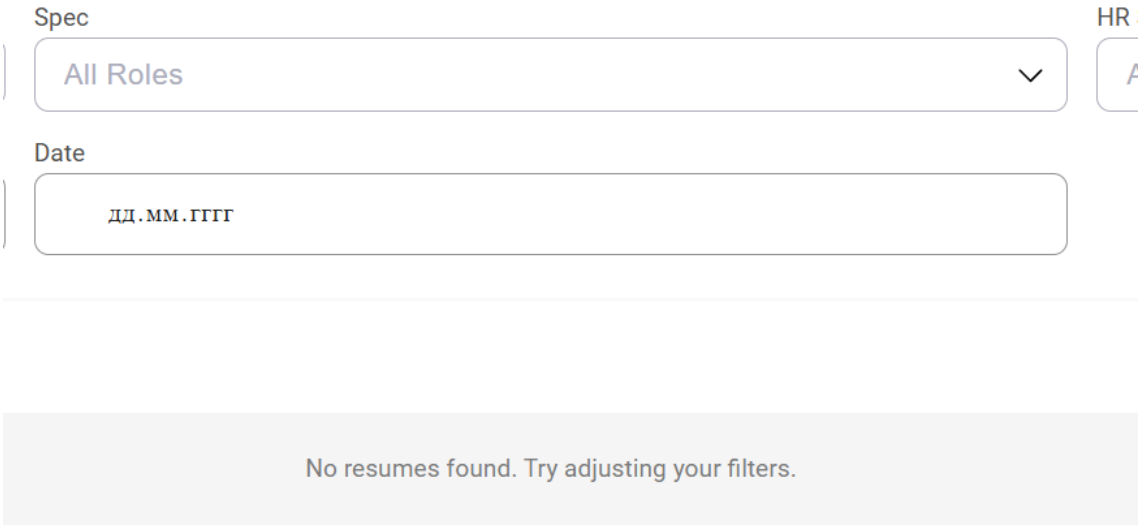


Рисунок 4.29 – Результат видалення резюме з облікового запису

Висновки до розділу

Проведено оцінку веб-застосунку ShareCV, що складалась з аналізу якості програмного забезпечення, мануального тестування основних

функціональних можливостей та опису контрольного прикладу роботи застосунку.

За допомогою інструменту codequality.browserstack.com[15] було проведено аналіз якості коду, який охоплює метрики вразливості, анти-патернів, дублювання коду. Аналіз усього застосунку визначив рейтинг, що дорівнює 4.35(за шкалою від -5 до 5), що є більш ніж задовільно.

Тестування проведено згідно з розробленою програмою та методикою тестування. Було виконано 15 тест-кейсів, що охоплюють головні сценарії взаємодії з системою. Було протестовано сценарії роботи звичайного користувача: реєстрація, авторизація, створення та відправка резюме, пошук та коментування публічних резюме, а також функціонал адміністратора: перегляд поданих резюме та їхнє затвердження. Усі проведені тести показали роботу застосунку, що відповідає очікуваним результатам.

Контрольним прикладом продемонстроване послідовне виконання основних користувацьких та адміністративних варантів використання: від реєстрації та створення резюме до його підтвердження адміністратором, з подальшим переглядом звичайним користувачем опублікованих резюме, й після видалення особистого облікового запису користувача з усіма його даними.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Оскільки при розробці було створено багато сервісів, було прийнято рішення створити відокремлені контейнери Docker для кожного сервісу.

Для зручності розгортання усі контейнери було об’єднано в єдиний docker compose, що включає в себе 5 контейнерів: контейнер сервісу для прибирання персональних даних, контейнер сховища документів, контейнер бази даних, контейнер клієнтського застосунку та контейнер бекенд сервісу з бізнес логікою.

Для розгортання було обрано хмарний сервіс Microsoft Azure. Перед розгортанням потрібно підготувати інфраструктуру у сервісі. Спочатку було створено Resource Group, це організаційна сутність, що допомагає об’єднати усі створені ресурси у єдину групу.

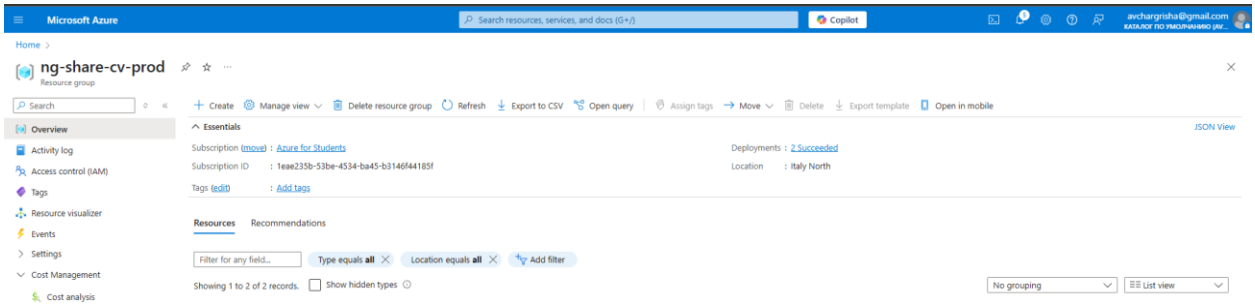


Рисунок 5.1 – Сутність Azure Resource Group

Після цього у створеній ресурсній групі було створено App Service plan, ця сутність відповідає за управління коштів сервісами, потужності, збір статистики тощо.

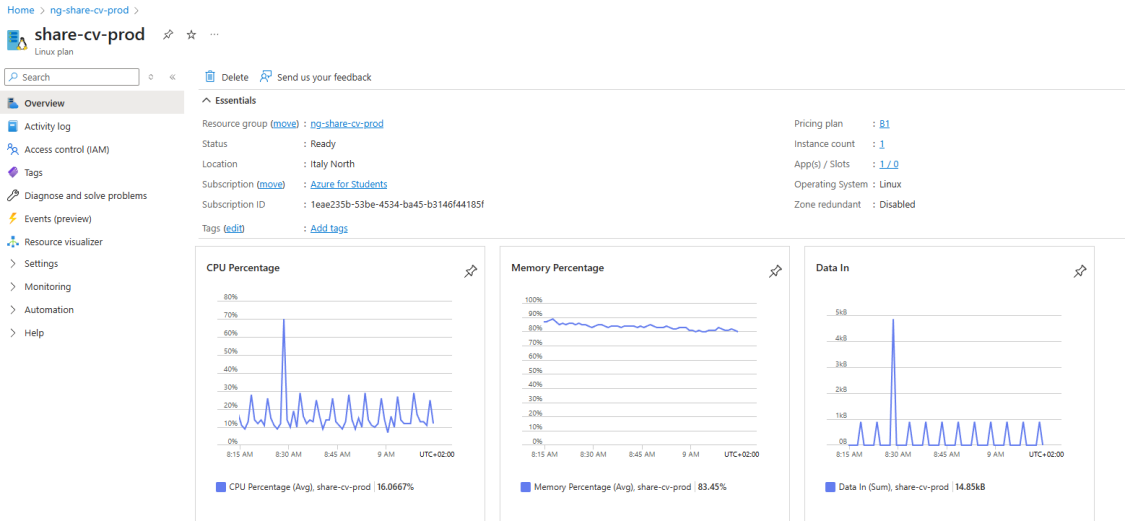


Рисунок 5.2 – Сутність Azure App Service plan

Наступними кроком було створено й конфігуровано сам сервіс застосунку Azure App Service, що прив’язаний до створеного сервісного плану.

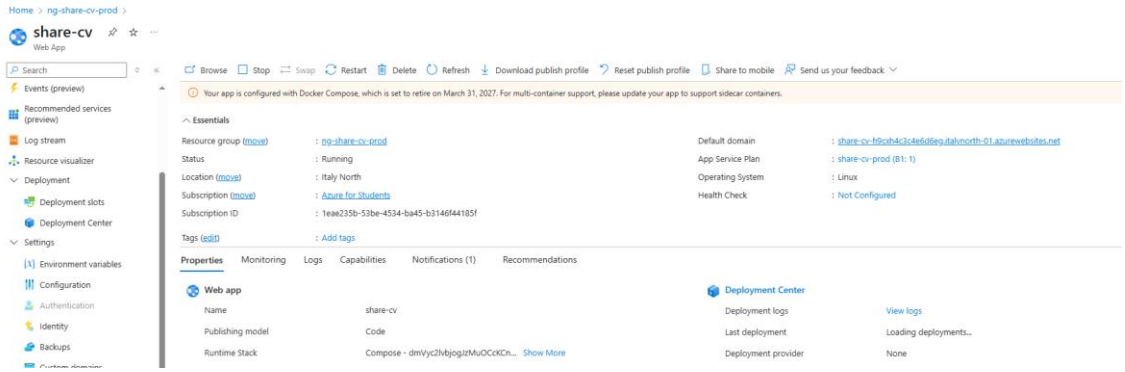


Рисунок 5.3 – Сутність Azure App Service

Також були додані усі змінні оточення для роботи усіх сервісів, це надає гнучкість, що дає змогу змінювати конфігурації сервісів без додаткового розгортання.

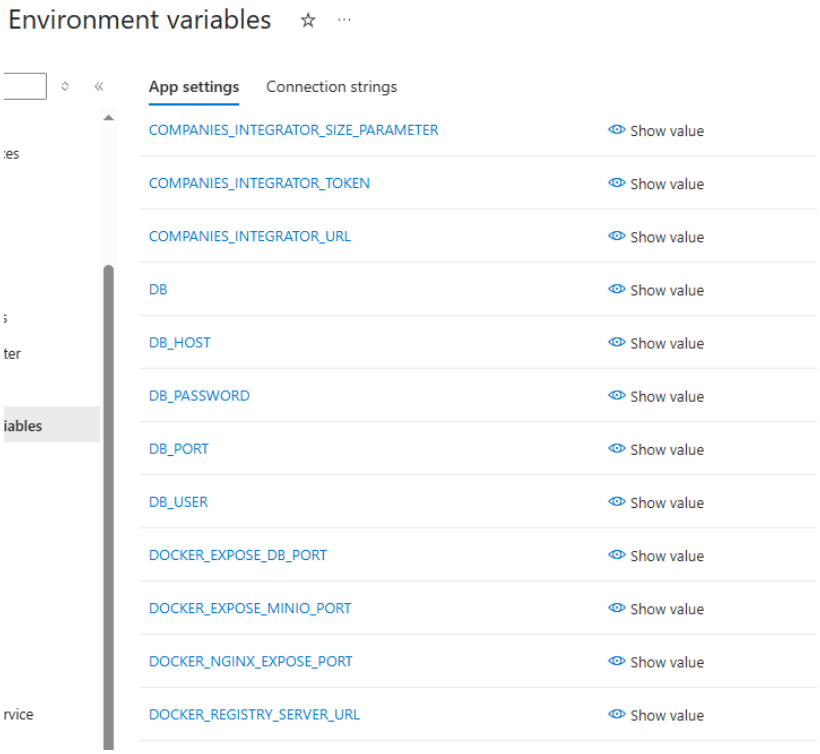


Рисунок 5.4 – Змінні оточення у середовищі Azure

Тепер коли оточення готове до розгортання потрібно створити сховище для образів, що описують кожен сервіс, для цього було використано Docker Hub.

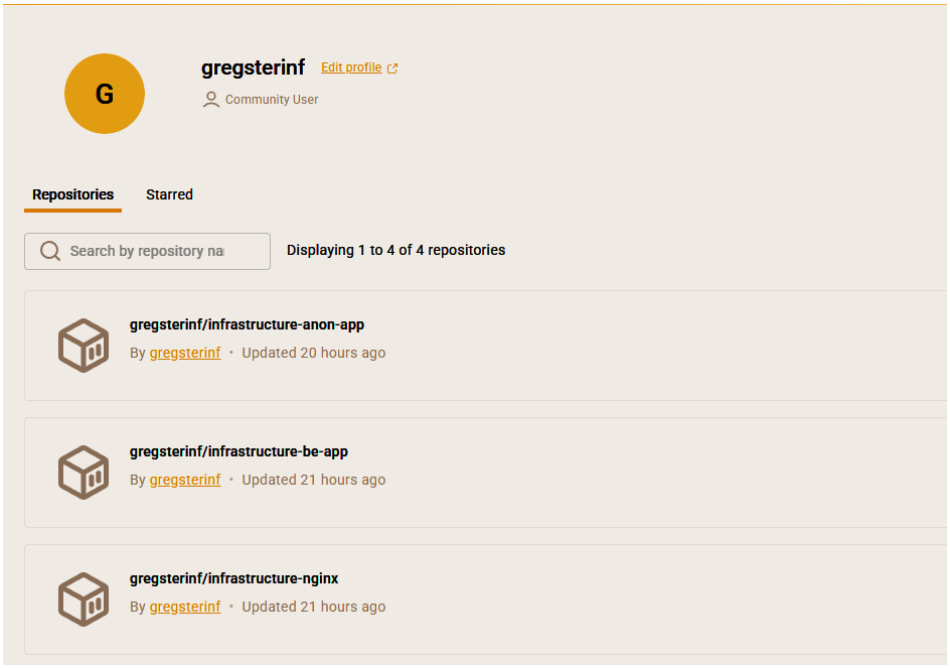


Рисунок 5.5 – Бібліотека образів Docker Hub

Кінцевим етапом є конфігурація Azure App Service, де для розгортання потрібно вказати docker compose файл, що використовує образи з Docker Hub. Таким чином кожен раз коли у Docker Hub буде завантажена нова версія одного з образів, застосунок автоматично оновиться відповідно до нових образів.

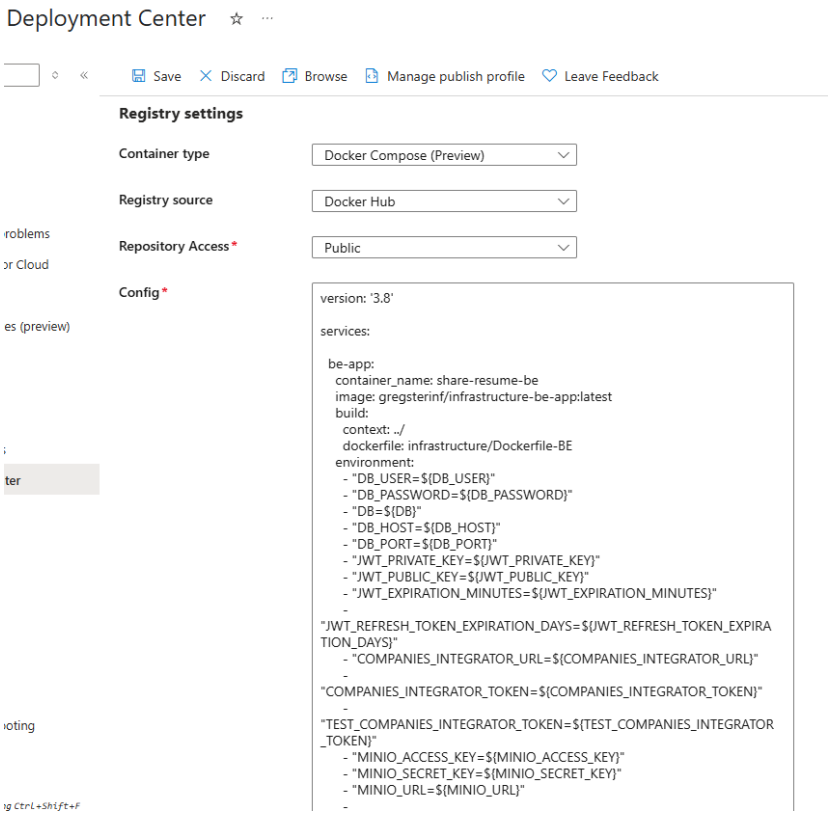


Рисунок 5.6 – Конфігурація розгортання Azure App Service

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «Керівництво користувача».

Для того щоб користувачі отримали нову версію застосунку після того як вона буде готова для публікації було використано Docker Hub.

Створення нової версії починається з збірки образів сервісів застосунку, що потребують оновлення, за допомогою docker compose або окремого Dockerfile.

Після створення образів достатньо оновити їх у Docker Hub(рис 5.7) й усі зміни будуть автоматично заватажені на хмарний сервіс, через те, що усі образи у хмарі орієнтуються на збірки у Docker Hub.

```
Admin@Xiaomi MINGW64 ~/Documents/ShareCV/infrastructure (main)
$ docker push gregterinf/infrastructure-nginx:latest
```

Рисунок 5.7 – Команда для відправки образу у Docker Hub

Єдина конфігурація, що не підв’язана до змінних оточення у хмарі це фронтенд сервіс, для зміни конфігурацій потрібно знайти файл environment.prod.ts(рис 5.8), змінити потрібні дані та завантажити новий образ у Docker Hub.

```
export const environment = {
  production: true,
  apiBaseUrl: 'https://share-cv-h9cxh4c3c4e6d6eg.italynorth-01.azurewebsites.n
  useMockApi: false, // Disable mock API in production environment
};
```

Рисунок 5.8 – Змінні оточення клієнтського застосунку

Висновки до розділу

Описано процес розгортання застосунку з використанням контейнеризації та хмарних сервісів.

За використання Docker контейнерів вдалося ізолювати окремі сервіси – зокрема, клієнт, сервер, базу даних, сервіс прибирання персональних даних та сховище документів. Такий підхід спрощує масштабування системи, та пришвидшує доставку змін до користувачів.

Завдяки використанню Docker Compose створено єдине середовище для управління всіма сервісами як єдиним цілісним застосунком. Також через повну контейнеризацію система є повністю платформо-незалежною, її розгортання можливо не тільки в середовищі Microsoft Azure, яке було використано, але й будь-якому іншому, що підтримує запуск Docker контейнерів. Додатково було використано Docker Hub, як централізоване місце для зберігання та поширення контейнерів.

Описано процес підтримки та оновлення застосунку, завдяки автоматизованому розгортанню оновлень через зв’язок між Docker Hub та Azure App Service. Особливу увагу приділено конфігурації клієнтського сервісу, що вимагає перебудови контейнера при зміні параметрів, на відміну від сервісів серверної частини, які легко переналаштовуються через змінні середовища.

ВИСНОВКИ

У ході виконання дипломного проєкту було спроєктовано та розроблено веб-застосунок "ShareCV", що призначений для покращення ефективності аналізу резюме шляхом створення веб-застосунку для обміну резюме та досвідом проходження співбесід.

Користувачі надана можливість завантажувати резюме, які проходять процес напівавтоматичної прибирання персональних даних та адміністраторської модерації перед публікацією. Веб-застосунок дозволяє іншим користувачам переглядати ці резюме, фільтрувати їх за різними критеріями (наприклад, компанія, спеціальність) та залишати коментарі, сприяючи колективному покращенню якості резюме.

Усі ключові етапи, передбачені завданням дипломного проєктування, були успішно виконані:

– проведено комплексне передпроєктне обстеження, включаючи аналіз актуальних проблем ринку праці, існуючих аналогів та моделювання ключових бізнес-процесів;

– сформульовано детальні функціональні та нефункціональні вимоги до системи, а також проведено економічний аналіз проєкту;

– розроблено архітектуру програмного забезпечення на основі контейнеризованих сервісів (клієнтська частина на Angular, серверна на Java Spring Boot, сервіс прибирання персональних даних на Python), обґрунтовано вибір технологічного стеку (PostgreSQL, MinIO, Docker) та інструментів розробки;

– здійснено конструювання та розробку програмного забезпечення, включаючи реалізацію клієнтської та серверної частин, а також впровадження механізмів для забезпечення безпеки даних;

Проведено аналіз якості коду за допомогою інструменту статичного аналізу CodeQuality від BrowserStack[15] та описано підходи до тестування системи.

Розроблено та описано процес розгортання застосунку з використанням Docker та хмарної платформи Microsoft Azure, а також окреслено основні аспекти супроводу програмного забезпечення.

Створений веб-застосунок "ShareCV" є актуальним та корисним інструментом, що відповідає нагальним потребам сучасного ринку праці. Потенційні напрямки подальшого розвитку включають:

– вдосконалення алгоритмів автоматичної обфускації резюме для мінімізації ручної модерації;

– впровадження розширених аналітичних інструментів для оцінки ефективності резюме;

– інтеграція з популярними платформами для пошуку роботи та професійними соціальними мережами;

Таким чином, проєкт "ShareCV" має значний потенціал для подальшого розвитку та вдосконалення, підвищення конкурентоспроможності на ринку праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Software Development Job Postings on Indeed in the United States. Federal Reserve Economic Data | FRED | St. Louis Fed. URL: https://fred.stlouisfed.org/series/IHLIDXUSTPSOFTDEVE (дата звернення: 11.05.2025).

2. All Employees, Computer Systems Design and Related Services. Federal Reserve Economic Data | FRED | St. Louis Fed. URL: https://fred.stlouisfed.org/series/CES6054150001 (дата звернення: 11.05.2025).

3. Job Postings on Indeed in the United States. Federal Reserve Economic Data | FRED | St. Louis Fed. URL: https://fred.stlouisfed.org/series/IHLIDXUS (дата звернення: 11.05.2025).

4. Reddit. Reddit. URL: https://www.reddit.com (дата звернення: 11.05.2025).

5. Jotform Spring Sale. Free Online Form Builder & Form Creator | Jotform. URL: https://jotform.com (дата звернення: 11.05.2025).

6. Enrichment API for Companies. The Companies API. URL: https://www.thecompaniesapi.com/api (дата звернення: 11.05.2025).

7. ZoomInfo. ZoomInfo. URL: https://www.zoominfo.com/ (дата звернення: 11.05.2025).

8. Clearbit is Now Breeze Intelligence for HubSpot. Clearbit is Now Breeze Intelligence for HubSpot. URL: https://clearbit.com (дата звернення: 11.05.2025).

9. nginx. *nginx*. URL: https://nginx.org (дата звернення: 30.05.2025).

10. Angular. Angular. URL: https://angular.dev (дата звернення: 11.05.2025).

11. Chebbi L. Reactive Patterns with RxJS and Angular Signals: Elevate Your Angular 17 Applications with RxJS Observables, Subjects, Operators, and Angular Signals. Packt Publishing, Limited, 2024.

12. MinIO | S3 Compatible Storage for AI. MinIO. URL: https://min.io (дата звернення: 11.05.2025).

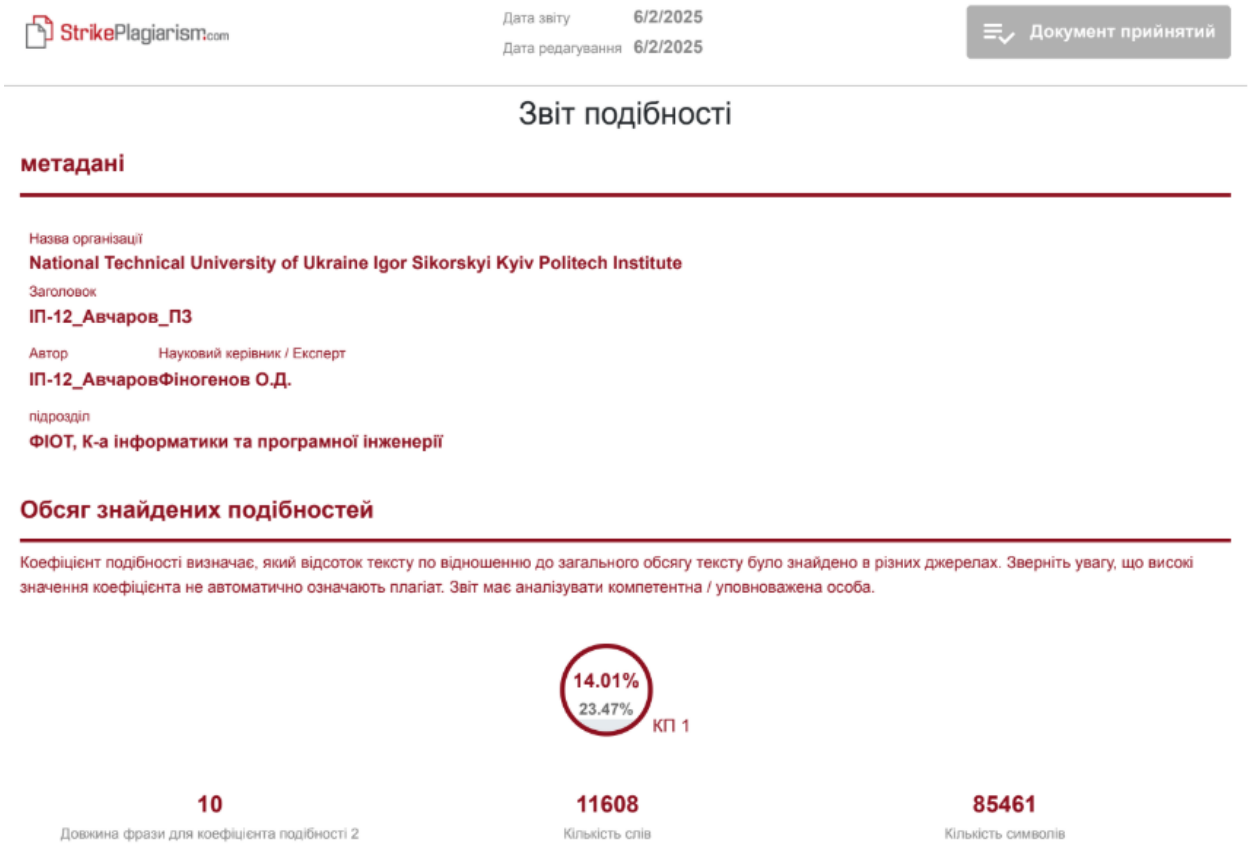
13. Microsoft Presidio. Microsoft on GitHub. URL: https://microsoft.github.io/presidio/analyzer/ (дата звернення: 11.05.2025).

14. Docker: Accelerated Container Application Development. Docker. URL: https://www.docker.com (дата звернення: 11.05.2025).

15. BrowserStack Code Quality | BrowserStack. BrowserStack. URL: https://www.browserstack.com/codequality (дата звернення: 11.05.2025).

ДОДАТКИ

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



StrikePlagiarism.com

Дата звіту 6/2/2025
Дата редагування 6/2/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-12_Авчаров_ПЗ
Автор Науковий керівник / Експерт
ІП-12_АвчаровФіногенов О.Д.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

14.01%
23.47%
КП 1

10	11608	85461
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

Вебзастосунок для покращення ефективності резюме

Текст програми

КПІ.ІП-1201.045440.02.81

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Григорій АВЧАРОВ

Київ – 2025

Посилання на репозиторій з повним текстом програмного коду
https://github.com/ShareResume/share-resume-be
https://github.com/ShareResume/share-cv-fe
https://github.com/ShareResume/anonymize-app
https://github.com/ShareResume/infrastructure

Файл ResumeStateMachinePersist.java

Реалізація функціональної задачі зміни статусу резюме адміністратором

```
 package share.resume.com.services.fsm;

import lombok.RequiredArgsConstructor;
import org.springframework.statemachine.StateMachineContext;
import org.springframework.statemachine.StateMachinePersist;
import org.springframework.statemachine.support.DefaultStateMachineContext;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;
import share.resume.com.entities.ResumeEntity;
import share.resume.com.entities.enums.ResumeEvent;
import share.resume.com.entities.enums.ResumeStatus;
import share.resume.com.exceptions.EntityNotFoundException;
import share.resume.com.repositories.ResumeRepository;

import java.util.UUID;

@Component
@RequiredArgsConstructor
public class ResumeStateMachinePersist implements
StateMachinePersist<ResumeStatus, ResumeEvent, UUID> {
    private final ResumeRepository resumeRepository;

    @Override
    @Transactional
    public void write(StateMachineContext<ResumeStatus, ResumeEvent> context,
UUID resumeId) {
        ResumeEntity resume = resumeRepository.findById(resumeId)
                .orElseThrow(() -> new EntityNotFoundException("Resume with
id " + resumeId + " not found while updating state machine state"));
        resume.setStatus(context.getState());
        resumeRepository.save(resume);
    }

    @Override
    @Transactional
    public StateMachineContext<ResumeStatus, ResumeEvent> read(UUID resumeId)
{
        ResumeEntity resume = resumeRepository.findById(resumeId)
                .orElseThrow(() -> new EntityNotFoundException("Resume with
id " + resumeId + " not found while updating state machine state"));
        return new DefaultStateMachineContext<>(resume.getStatus(), null,
null, null);
    }

}
```

Файл ResumeStateMachineConfiguration.java

Реалізація функціональної задачі зміни статусу резюме адміністратором

```
package share.resume.com.configs.fsm;

import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.statemachine.StateMachinePersist;
import org.springframework.statemachine.config.EnableStateMachineFactory;
import org.springframework.statemachine.config.StateMachineConfigurerAdapter;
import
org.springframework.statemachine.config.builders.StateMachineStateConfigurer;
import
org.springframework.statemachine.config.builders.StateMachineTransitionConfig
urer;
import org.springframework.statemachine.persist.DefaultStateMachinePersister;
import org.springframework.statemachine.persist.StateMachinePersister;
```

```
import share.resume.com.entities.enums.ResumeEvent;
import share.resume.com.entities.enums.ResumeStatus;

import java.util.EnumSet;
import java.util.UUID;

@Configuration
@EnableStateMachineFactory
@RequiredArgsConstructor
public class ResumeStateMachineConfiguration extends
StateMachineConfigurerAdapter<ResumeStatus, ResumeEvent> {
    private final StateMachinePersist<ResumeStatus, ResumeEvent, UUID>
persist;

    @Override
    public void configure(StateMachineStateConfigurer<ResumeStatus,
ResumeEvent> states) throws Exception {
        states
                .withStates()
                .initial(ResumeStatus.WAITING_FOR_APPROVE)
                .states(EnumSet.allOf(ResumeStatus.class));
    }

    @Override
    public void configure(StateMachineTransitionConfigurer<ResumeStatus,
ResumeEvent> transitions) throws Exception {
        transitions
                .withExternal()
                .source(ResumeStatus.WAITING_FOR_APPROVE)
                .target(ResumeStatus.APPROVED)
                .event(ResumeEvent.APPROVED)
                .and()
                .withExternal()
                .source(ResumeStatus.WAITING_FOR_APPROVE)
                .target(ResumeStatus.REJECTED)
                .event(ResumeEvent.REJECTED);
    }

    @Bean
    public StateMachinePersister<ResumeStatus, ResumeEvent, UUID> persister()
{
        return new DefaultStateMachinePersister<>(persist);
    }
}
```

Файл CommentService.java

Реалізація функціональних задач коментування резюме, відповіді на коментарі, голосування за коментарі й оновлення їх рейтингу

```
package share.resume.com.services;

import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import share.resume.com.controllers.dto.request.CreateCommentRequestBody;
import share.resume.com.controllers.dto.request.ReactToCommentRequestBody;
import share.resume.com.controllers.dto.response.CommentResponseBody;
import share.resume.com.entities.CommentEntity;
import share.resume.com.entities.ResumeEntity;
import share.resume.com.entities.UserCommentVoteStateEntity;
import share.resume.com.entities.UserEntity;
import share.resume.com.entities.enums.VoteStateEnum;
import share.resume.com.exceptions.ActionNotAllowed;
import share.resume.com.exceptions.EntityNotFoundException;
import share.resume.com.repositories.CommentRepository;
import share.resume.com.security.dto.UserDetailsDto;

import java.time.LocalDateTime;
import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class CommentService {
    private final CommentRepository commentRepository;
    private final ResumeService resumeService;
```

```
    @Transactional
    public void createComment(CreateCommentRequestBody
createCommentRequestBody) {
        CommentEntity commentEntity = new CommentEntity();
        UUID resumeId = createCommentRequestBody.getResumeId();
        ResumeEntity resume = resumeService.getById(resumeId);
        UUID parentCommentId = createCommentRequestBody.getParentCommentId();
        if (parentCommentId != null && !existsById(parentCommentId)) {
            throw new EntityNotFoundException("Parent comment does not
exist");
        }
        commentEntity.setResume(resume);
        commentEntity.setParentCommentId(parentCommentId);
        commentEntity.setMessage(createCommentRequestBody.getText());
        commentEntity.setCreatedAt(LocalDateTime.now());
        commentRepository.save(commentEntity);
    }

    @Transactional
    public void reactToComment(ReactToCommentRequestBody
reactToCommentRequestBody) {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        UserEntity user = userDetailsDto.getUserEntity();
        CommentEntity comment =
getById(reactToCommentRequestBody.getCommentId());
        if (comment.getUserCommentVoteStates().stream()
                .anyMatch(userCommentVoteStateEntity ->
userCommentVoteStateEntity.getUser().equals(user))) {
            throw new ActionNotAllowed("Cannot react to comment because it is
already voted");
        }
        VoteStateEnum voteState = reactToCommentRequestBody.getVoteState();
        if (VoteStateEnum.UNDEFINED.equals(voteState)) {
            throw new ActionNotAllowed("You can't react with undefined vote
state");
        }
        int commentRate = comment.getReactionsRate();
        if (VoteStateEnum.UP.equals(voteState)) {
            comment.setReactionsRate(commentRate + 1);
        }
        if (VoteStateEnum.DOWN.equals(voteState) && commentRate > 0) {
            comment.setReactionsRate(commentRate - 1);
        }
        UserCommentVoteStateEntity userCommentVoteStateEntity = new
UserCommentVoteStateEntity();
        userCommentVoteStateEntity.setComment(comment);
        userCommentVoteStateEntity.setVoteState(voteState);
        userCommentVoteStateEntity.setUser(user);
        comment.getUserCommentVoteStates().add(userCommentVoteStateEntity);
        commentRepository.save(comment);
    }

    @Transactional
    public List<CommentResponseBody> getAllByResumeId(UUID resumeId) {
        return commentRepository.findAllByResume_Id(resumeId).stream()
                .map(CommentResponseBody::new)
                .toList();
    }

    @Transactional
    public boolean existsById(UUID commentId) {
        return commentRepository.existsById(commentId);
    }

    @Transactional
    public CommentEntity getById(UUID commentId) {
        return commentRepository.findById(commentId)
                .orElseThrow(() -> new EntityNotFoundException("Comment with
id" + commentId + " not found"));
    }
}
```

Файл ResumeService.java

Реалізація функціональних задач створення й перегляду особистих резюме та отримання документів резюме.

```
package share.resume.com.services;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.ApplicationEventPublisher;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.multipart.MultipartFile;
import share.resume.com.async.transactions.events.ResumeCreatedEvent;
import share.resume.com.async.transactions.events.ResumeDeletedEvent;
import share.resume.com.controllers.dto.DocumentView;
import share.resume.com.controllers.dto.request.CompanyWithStatusDto;
import share.resume.com.controllers.dto.request.CreateResumeRequestBody;
import share.resume.com.controllers.dto.request.UpdateResumeRequestBody;
import share.resume.com.controllers.dto.response.ResumeResponseBody;
import share.resume.com.entities.*;
import share.resume.com.entities.enums.DocumentAccessTypeEnum;
import share.resume.com.entities.enums.ResumeStatus;
import share.resume.com.entities.enums.RoleEnum;
import share.resume.com.exceptions.EntityNotFoundException;
import share.resume.com.repositories.ResumeRepository;
import share.resume.com.security.dto.UserDetailsDto;
import share.resume.com.services.commands.facade.ResumeUpdateCommandFacade;
import share.resume.com.services.files.FileService;
import share.resume.com.services.integrators.AnonymizerIntegratorService;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
@Slf4j
public class ResumeService {
    private final ApplicationEventPublisher eventPublisher;
    private final CompanyService companyService;
    private final ResumeRepository resumeRepository;
    private final FileService fileService;
    private final AnonymizerIntegratorService anonymizerIntegratorService;
    private final ResumeUpdateCommandFacade resumeUpdateCommandFacade;

    @Transactional
    public void save(CreateResumeRequestBody createResumeRequestBody) {
        ResumeEntity resume = new ResumeEntity();
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        UserEntity user = userDetailsDto.getUserEntity();
        resume.setAuthor(user);
        resume.setCreatedAt(LocalDateTime.now());
        resume.setSpeciality(createResumeRequestBody.getSpeciality());
        resume.setYearsOfExperience(createResumeRequestBody.getYearsOfExperie
nce());
        resume.setStatus(ResumeStatus.WAITING_FOR_APPROVE);

        MultipartFile publicCv = createResumeRequestBody.getDocument();
        String directoryName = "user-directory-" + user.getId();
        String publicCvFileName = publicCv.getOriginalFilename();
        DocumentEntity publicCvEnitiy = new DocumentEntity();
        publicCvEnitiy.setAccessType(DocumentAccessTypeEnum.PUBLIC);
        publicCvEnitiy.setResume(resume);
        publicCvEnitiy.setDirectory(directoryName);
        publicCvEnitiy.setName(publicCvFileName);

        MultipartFile privateCvFile =
anonymizerIntegratorService.anonymize(publicCv);

        DocumentEntity privateCvEntity = new DocumentEntity();
        privateCvEntity.setAccessType(DocumentAccessTypeEnum.PRIVATE);
        privateCvEntity.setResume(resume);
        privateCvEntity.setDirectory(directoryName);
        privateCvEntity.setName(privateCvFile.getOriginalFilename());
        resume.setDocuments(List.of(publicCvEnitiy, privateCvEntity));

        List<CompanyEntity> companies =
companyService.getAllByIds(createResumeRequestBody.getCompanies().stream().ma
p(CompanyWithStatusDto::getId).collect(Collectors.toList()));
        List<ResumesCompaniesEntity> resumesCompaniesEntities = new
ArrayList<>();
```

```
        for (CompanyEntity company : companies) {
            ResumesCompaniesEntity resumesCompaniesEntity = new
ResumesCompaniesEntity();
            resumesCompaniesEntity.setCompany(company);
            resumesCompaniesEntity.setResume(resume);
            resumesCompaniesEntity.setIsHrScreeningPassed(createResumeRequest
Body.getCompanies().stream().filter(companyWithStatusDto ->
company.getId().equals(companyWithStatusDto.getId())).findFirst().get().getIs
HrScreeningPassed());
            resumesCompaniesEntities.add(resumesCompaniesEntity);
        }
        resume.setResumesCompanies(resumesCompaniesEntities);

        eventPublisher.publishEvent(new
ResumeCreatedEvent(createResumeRequestBody.getDocument(), publicCvFileName,
directoryName, privateCvFile, privateCvFile.getOriginalFilename()));
        resumeRepository.save(resume);
    }

    @Transactional
    public void update(UUID id, UpdateResumeRequestBody
updateResumeRequestBody) throws Exception {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        resumeUpdateCommandFacade.execute(userDetailsDto.getRole(), id,
updateResumeRequestBody);
    }

    @Transactional
    public ResumeEntity getById(UUID id) {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        return resumeRepository.findByIdAndAuthor(id,
userDetailsDto.getUserEntity())
                .orElseThrow(() -> new EntityNotFoundException("Resume with
id: " + id + " not found for user: " + userDetailsDto.getEmail()));
    }

    @Transactional
    public void delete(UUID id) {
        ResumeEntity resume = getById(id);
        resumeRepository.deleteById(id);
        DocumentEntity publicDoc = resume.getDocuments().stream()
                .filter(d -> d.getAccessType() ==
DocumentAccessTypeEnum.PUBLIC)
                .findAny()
                .get();
        DocumentEntity privateDoc = resume.getDocuments().stream()
                .filter(d -> d.getAccessType() ==
DocumentAccessTypeEnum.PRIVATE)
                .findAny()
                .get();
        String directoryName = publicDoc.getDirectory();
        eventPublisher.publishEvent(new
ResumeDeletedEvent(publicDoc.getName(), privateDoc.getName(),
directoryName));
    }

    @Transactional
    public List<ResumeResponseBody> getAll() {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        List<ResumeEntity> resumes;
        if (RoleEnum.USER.equals(userDetailsDto.getRole())) {
            resumes =
resumeRepository.findAllByAuthor(userDetailsDto.getUserEntity());
        } else {
            resumes = resumeRepository.findAll();
        }
        List<ResumeResponseBody> resumeResponseBodies = new ArrayList<>();
        for (ResumeEntity resume : resumes) {
            List<DocumentView> documentViews = getDocumentsByResume(resume);
            resumeResponseBodies.add(new ResumeResponseBody(resume,
documentViews));
        }
        return resumeResponseBodies;
    }

    public List<DocumentView> getDocumentsByResume(ResumeEntity resume) {
        List<DocumentView> documentViews = new ArrayList<>();
        for (DocumentEntity document : resume.getDocuments()) {
```

```
            DocumentView documentView = new DocumentView();
            documentView.setAccessType(document.getAccessType());
            documentView.setName(document.getName());
            documentView.setUrl(fileService.getAccessLink(document.getDirecto
ry(), document.getName()));
            documentViews.add(documentView);
        }
        return documentViews;
    }

    public DocumentView getPrivateDocumentByResume(ResumeEntity resume) {
        DocumentEntity document = resume.getDocuments().stream()
                .filter(doc ->
DocumentAccessTypeEnum.PRIVATE.equals(doc.getAccessType()))
                .findFirst()
                .orElseThrow(() -> new EntityNotFoundException("Private
document for resume with id: " + resume.getId() + " not found"));
        DocumentView documentView = new DocumentView();
        documentView.setAccessType(DocumentAccessTypeEnum.PRIVATE);
        documentView.setName(document.getName());
        documentView.setUrl(fileService.getAccessLink(document.getDirectory()
, document.getName()));
        return documentView;
    }
}
```

Файл UsersResumesService.java

Реалізація функціональних задач перегляду публічних резюме й зміни документу резюме.

```
package share.resume.com.services;

import jakarta.transaction.Transactional;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import share.resume.com.controllers.dto.ResumeFilterDto;
import share.resume.com.controllers.dto.response.UserResumeResponseBody;
import share.resume.com.entities.DocumentEntity;
import share.resume.com.entities.ResumeEntity;
import share.resume.com.entities.ResumesCompaniesEntity;
import share.resume.com.entities.enums.DocumentAccessTypeEnum;
import share.resume.com.entities.enums.ResumeStatus;
import share.resume.com.entities.enums.RoleEnum;
import share.resume.com.exceptions.ActionNotAllowed;
import share.resume.com.exceptions.EntityNotFoundException;
import share.resume.com.repositories.ResumeRepository;
import share.resume.com.repositories.criteria.ResumeSpecification;
import share.resume.com.security.dto.UserDetailsDto;
import share.resume.com.services.files.FileService;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class UsersResumesService {
    private final ResumeSpecification resumeSpecification;
    private final ResumeRepository resumeRepository;
    private final ResumeService resumeService;
    private final FileService fileService;

    @Transactional
    public List<UserResumeResponseBody> getAll(ResumeFilterDto filter) {
        List<ResumeEntity> resumes =
resumeRepository.findAll(resumeSpecification.filterBy(filter));
        resumes.forEach(resume -> {
            List<ResumesCompaniesEntity> filteredCompanies =
resume.getResumesCompanies().stream()
                    .filter(rc -> (filter.getCompanyId() == null ||
rc.getCompany().getId().equals(filter.getCompanyId())) &&
                            (filter.getIsHrScreeningPassed() == null ||
rc.getIsHrScreeningPassed().equals(filter.getIsHrScreeningPassed())))
                    .collect(Collectors.toList());
```

```
            resume.setResumesCompanies(filteredCompanies);
        });
        return resumes.stream()
                .filter(resume ->
ResumeStatus.APPROVED.equals(resume.getStatus()))
                .map(resume -> new UserResumeResponseBody(resume,
resumeService.getPrivateDocumentByResume(resume)))
                .collect(Collectors.toList());
    }

    @Transactional
    public ResumeEntity getById(UUID id) {
        return resumeRepository.findByIdAndStatus(id, ResumeStatus.APPROVED)
                .orElseThrow(() -> new EntityNotFoundException("Resume with
id: " + id + " not found"));
    }

    @Transactional
    public void updateUserPrivateResume(UUID resumeId, MultipartFile
newPrivateResume) {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        if (!RoleEnum.ADMIN.equals(userDetailsDto.getRole())) {
            throw new ActionNotAllowed("Only admins can update public
resume");
        }
        ResumeEntity resume = resumeRepository.findById(resumeId)
                .orElseThrow(() -> new EntityNotFoundException("Resume with
id: " + resumeId + " not found"));
        DocumentEntity privateDocumentToUpdate =
resume.getDocuments().stream()
                .filter(documentEntity ->
DocumentAccessTypeEnum.PRIVATE.equals(documentEntity.getAccessType()))
                .findAny()
                .orElseThrow(() -> new EntityNotFoundException("Private
document not found for resume with id: " + resumeId));
        fileService.delete(privateDocumentToUpdate.getDirectory(),
privateDocumentToUpdate.getName());
        privateDocumentToUpdate.setName("PRIVATE-" +
newPrivateResume.getOriginalFilename());
        fileService.upload(privateDocumentToUpdate.getDirectory(),
newPrivateResume, privateDocumentToUpdate.getName());
        resumeRepository.save(resume);
    }
}
```

Файл UsersFavoritesResumesService.java

Реалізація функціональної задачі додавання резюме до обраних.

```
package share.resume.com.services;

import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import share.resume.com.controllers.dto.request.BookmarkResumeRequestBody;
import share.resume.com.controllers.dto.response.ResumeResponseBody;
import share.resume.com.entities.ResumeEntity;
import share.resume.com.entities.UserEntity;
import share.resume.com.entities.UsersFavoritesResumesEntity;
import share.resume.com.exceptions.ActionNotAllowed;
import share.resume.com.repositories.UsersFavoritesResumesRepository;
import share.resume.com.security.dto.UserDetailsDto;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class UsersFavoritesResumesService {
    private final UsersFavoritesResumesRepository
usersFavoritesResumesRepository;
    private final ResumeService resumeService;
    private final UsersResumesService usersResumesService;

    @Transactional
```

```
    public void bookmarkResume(BookmarkResumeRequestBody
bookmarkResumeRequestBody) {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        UserEntity user = userDetailsDto.getUserEntity();
        ResumeEntity resume =
usersResumesService.getById(bookmarkResumeRequestBody.getResumeId());
        Optional<UsersFavoritesResumesEntity> usersFavoritesResumesEntity =
usersFavoritesResumesRepository.findByUser_IdAndResume_Id(user.getId(),
resume.getId());
        if (usersFavoritesResumesEntity.isPresent()) {
            throw new ActionNotAllowed("You can not bookmark resume with id "
+ bookmarkResumeRequestBody.getResumeId() + " twice");
        }
        UsersFavoritesResumesEntity usersFavoritesResumes = new
UsersFavoritesResumesEntity();
        usersFavoritesResumes.setUser(user);
        usersFavoritesResumes.setResume(resume);
        usersFavoritesResumesRepository.save(usersFavoritesResumes);
    }

    @Transactional
    public void removeResumeFromFavorites(UUID bookmarkedResumeId) {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        usersFavoritesResumesRepository.deleteByUser_IdAndResume_Id(userDetai
lsDto.getId(), bookmarkedResumeId);
    }

    @Transactional
    public List<ResumeResponseBody> getFavoriteResumes() {
        UserDetailsDto userDetailsDto = (UserDetailsDto)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        List<UsersFavoritesResumesEntity> usersFavoritesResumesEntities =
usersFavoritesResumesRepository.findByUser_Id(userDetailsDto.getId());
        return usersFavoritesResumesEntities.stream()
                .map(UsersFavoritesResumesEntity::getResume)
                .map(resume -> new ResumeResponseBody(resume,
List.of(resumeService.getPrivateDocumentByResume(resume))))
                .toList();
    }
}
```

Файл UserDataDeletionService.java

Реалізація функціональної задачі видалення даних користувача й його облікового запису.

```
package share.resume.com.services;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import share.resume.com.entities.*;
import share.resume.com.repositories.*;
import share.resume.com.services.files.FileService;

import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor
@Slf4j
public class UserDataDeletionService {

    private final UserRepository userRepository;
    private final ResumeRepository resumeRepository;
    private final FileService fileService;
    private final JdbcTemplate jdbcTemplate;

    @Transactional
    public void deleteUserData(UUID userId) {
        log.info("Starting deletion of all data for user with ID: {}",
userId);

        UserEntity user = userRepository.findById(userId)
```

```
                .orElseThrow(() -> new IllegalArgumentException("User not
found with ID: " + userId));

        deleteDocumentsFromMinioForUser(user);

        deleteUserDataWithNativeQueries(userId);

        log.info("Successfully deleted all data for user with ID: {}",
userId);
    }

    private void deleteDocumentsFromMinioForUser(UserEntity user) {
        log.info("Deleting documents from MINIO for user: {}", user.getId());

        try {
            List<ResumeEntity> userResumes =
resumeRepository.findAllByAuthor(user);

            for (ResumeEntity resume : userResumes) {
                deleteDocumentsFromMinio(resume.getDocuments());
            }

            log.info("Completed MINIO document deletion for {} resumes",
userResumes.size());
        } catch (Exception e) {
            log.error("Error deleting MINIO files, continuing with database
deletion", e);
        }
    }

    private void deleteDocumentsFromMinio(List<DocumentEntity> documents) {
        for (DocumentEntity document : documents) {
            try {
                fileService.delete(document.getDirectory(),
document.getName());
                log.info("Deleted file from MINIO: {}/{}",
document.getDirectory(), document.getName());
            } catch (Exception e) {
                log.error("Failed to delete file from MINIO: {}/{}, error:
{}",
                        document.getDirectory(), document.getName(),
e.getMessage());
            }
        }
    }

    private void deleteUserDataWithNativeQueries(UUID userId) {
        log.info("Deleting user data using native SQL queries for user: {}",
userId);

        try {
            int deletedVotes = jdbcTemplate.update(
                "DELETE FROM user_comment_vote_state WHERE user_id = ?",
userId);
            log.info("Deleted {} user comment vote states", deletedVotes);

            int deletedFavorites = jdbcTemplate.update(
                "DELETE FROM users_favorites_resumes WHERE user_id = ?",
userId);
            log.info("Deleted {} user favorites", deletedFavorites);

            int deletedCommentVotes = jdbcTemplate.update(
                "DELETE FROM user_comment_vote_state WHERE comment_id IN (" +
                "SELECT c.id FROM comments c " +
                "INNER JOIN resumes r ON c.resume_id = r.id " +
                "WHERE r.author_id = ?)", userId);
            log.info("Deleted {} comment vote states on user's resumes",
deletedCommentVotes);

            int deletedComments = jdbcTemplate.update(
                "DELETE FROM comments WHERE resume_id IN (" +
                "SELECT id FROM resumes WHERE author_id = ?)", userId);
            log.info("Deleted {} comments on user's resumes",
deletedComments);

            int deletedDocuments = jdbcTemplate.update(
                "DELETE FROM documents WHERE resume_id IN (" +
                "SELECT id FROM resumes WHERE author_id = ?)", userId);
            log.info("Deleted {} documents", deletedDocuments);

            int deletedResumeCompanies = jdbcTemplate.update(
```

```
                "DELETE FROM resumes_companies WHERE resume_id IN (" +
                "SELECT id FROM resumes WHERE author_id = ?)", userId);
            log.info("Deleted {} resume-company relationships",
deletedResumeCompanies);

            int deletedResumes = jdbcTemplate.update(
                "DELETE FROM resumes WHERE author_id = ?", userId);
            log.info("Deleted {} resumes", deletedResumes);

            int deletedUsers = jdbcTemplate.update(
                "DELETE FROM users WHERE id = ?", userId);
            log.info("Deleted {} user entities", deletedUsers);

        } catch (Exception e) {
            log.error("Error during native SQL deletion for user: {}",
userId, e);
            throw new RuntimeException("Failed to delete user data", e);
        }
    }
}
```

Файл CompanyAPIIntegratorService.java

Інтеграція з зовнішнім API для отримання інформації про компанії. Реалізація функціональних задач створення резюме та фільтрації за компанією.

```
package share.resume.com.services.integrators;

import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import share.resume.com.controllers.dto.CompanyResponseDto;

import java.util.*;

@Service
@RequiredArgsConstructor
@Slf4j
public class CompanyAPIIntegratorService {
    private final APIService apiService;

    @Value("${companies.integrator.url}")
    private String integratorUrl;

    @Value("${companies.integrator.token}")
    private String integratorToken;

    @Value("${companies.integrator.size.parameter}")
    private Integer sizeParameter;


    public List<CompanyResponseDto> getCompaniesByName(String companyName) {
        Map<String, Object> requestParams = new HashMap<>();
        requestParams.put("name", companyName);
        requestParams.put("size", sizeParameter);
        Map<String, String> requestHeaders = new HashMap<>();
        requestHeaders.put("Authorization", integratorToken);
        try {
            Map<String, Object> apiResponse =
apiService.sendGetRequest(integratorUrl, requestParams, requestHeaders);
            return parseCompanyResponse(apiResponse);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            return Collections.emptyList();
        }
    }

    private List<CompanyResponseDto> parseCompanyResponse(Map<String, Object>
response) {
        List<CompanyResponseDto> companiesDtos = new ArrayList<>();
        List<Map<String, Object>> companies = (List<Map<String, Object>>)
response.get("companies");
        for (Map<String, Object> company : companies) {
            Map<String, Object> about = (Map<String, Object>)
company.get("about");
            Map<String, Object> assets = (Map<String, Object>)
company.get("assets");
```

```
            if (about != null) {
                String companyName = (String) about.get("name");
                String logoUrl = null;
                if (assets != null) {
                    Map<String, Object> logo = (Map<String, Object>)
assets.get("logoSquare");
                    logoUrl = (logo != null) ? (String) logo.get("src") : "No
logo found";
                }
                if (companyName != null && !companyName.isBlank()) {
                    companiesDtos.add(new CompanyResponseDto(companyName,
logoUrl));
                }
            }
        }
        return companiesDtos;
    }
}
```

Файл AnonymizerIntegratorService.java

Реалізація функціональних задач завантаження документа з прихованими даними.

```
package share.resume.com.services.integrators;

import lombok.RequiredArgsConstructor;
import okhttp3.MediaType;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import share.resume.com.exceptions.HttpException;
import share.resume.com.util.CustomMultipartFile;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

@Service
@RequiredArgsConstructor
public class AnonymizerIntegratorService {
    private final APIService apiService;

    @Value("${anonymizer.integrator.url}")
    private String integratorUrl;

    @Value("${anonymizer.integrator.token}")
    private String integratorToken;

    public MultipartFile anonymize(MultipartFile file) {
        Map<String, String> headers = new HashMap<>();
        headers.put("authorization-token", integratorToken);
        headers.put("Content-Type", "multipart/form-data");

        Map<String, Object> body = new HashMap<>();
        body.put("document", file);

        Map<String, Object> response = apiService.sendPostRequest(
                integratorUrl,
                Collections.emptyMap(),
                headers,
                body,
                MediaType.parse("multipart/form-data")
        );

        String responseFile = (String) response.get("file");
        if (responseFile == null) {
            throw new HttpException("Anonymizer integrator service returned
null response");
        }

        return new CustomMultipartFile(
                "PRIVATE-" + file.getOriginalFilename(),
                "PRIVATE-" + file.getOriginalFilename(),
                file.getContentType(),
                responseFile.getBytes()
        );
    }
}
```

Файл MinioFileService.java

Робота з файловим сховищем MinIO. Реалізація функціональних задач, використовується для створення, редагування та отримання документів резюме.

```
package share.resume.com.services.files;

import io.minio.*;
import io.minio.http.Method;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import share.resume.com.exceptions.FileException;

import java.util.HashMap;
import java.util.Map;

@Service
@RequiredArgsConstructor
@Slf4j
public class MinioFileService implements FileService {
    private final MinioClient minioClient;

    @Override
    public void upload(String directory, MultipartFile file, String fileName)
{
        try {
            BucketExistsArgs bucketExistsArgs = BucketExistsArgs.builder()
                    .bucket(directory)
                    .build();
            if (!minioClient.bucketExists(bucketExistsArgs)) {
                MakeBucketArgs makeBucketArgs = MakeBucketArgs.builder()
                        .bucket(directory)
                        .build();
                minioClient.makeBucket(makeBucketArgs);
            }
            PutObjectArgs putObjectArgs = PutObjectArgs.builder()
                    .bucket(directory)
                    .stream(file.getInputStream(), file.getSize(), -1)
                    .object(fileName)
                    .build();
            minioClient.putObject(putObjectArgs);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            throw new FileException("Error during uploading file to MinIO");
        }
    }

    @Override
    public void delete(String directory, String filename) {
        try {
            RemoveObjectArgs removeObjectArgs = RemoveObjectArgs.builder()
                    .bucket(directory)
                    .object(filename)
                    .build();
            minioClient.removeObject(removeObjectArgs);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            throw new FileException("Error during deleting file from MinIO");
        }
    }

    @Override
    public String getAccessLink(String directory, String filename) {
        try {
            Map<String, String> reqParams = new HashMap<>();
            reqParams.put("response-content-type", "application/pdf");
            GetPresignedObjectUrlArgs getPresignedObjectUrlArgs =
GetPresignedObjectUrlArgs.builder()
                    .bucket(directory)
                    .object(filename)
                    .method(Method.GET)
                    .extraQueryParams(reqParams)
                    .build();
```

```
            return
minioClient.getPresignedObjectUrl(getPresignedObjectUrlArgs);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            throw new FileException("Error during retrieving access link from
MinIO");
        }
    }
}
```

Файл CompanyService.java

Робота з даними компаній. Реалізація функціональних задач створення й отримання резюме.

```
package share.resume.com.services;

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import share.resume.com.controllers.dto.CompanyResponseDto;
import share.resume.com.entities.CompanyEntity;
import share.resume.com.exceptions.EntityNotFoundException;
import share.resume.com.repositories.CompanyRepository;
import share.resume.com.services.integrators.CompanyAPIIntegratorService;

import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class CompanyService {
    private final CompanyAPIIntegratorService companyAPIIntegratorService;
    private final CompanyRepository companyRepository;

    @Transactional
    public List<CompanyResponseDto> getCompaniesByName(String name) {
        List<CompanyEntity> companies =
companyRepository.findAllByNameLike(name);
        if (companies.isEmpty()) {
            List<CompanyResponseDto> companiesDto =
companyAPIIntegratorService.getCompaniesByName(name);
            if (companiesDto.isEmpty()) {
                return Collections.emptyList();
            }
            companies.addAll(saveAll(companiesDto));
        }
        return companies.stream()
                .map(CompanyResponseDto::new)
                .toList();
    }

    private List<CompanyEntity> saveAll(List<CompanyResponseDto>
companiesDto) {
        List<CompanyEntity> companies = companiesDto.stream()
                .map(CompanyEntity::new)
                .toList();
        return companyRepository.saveAll(companies);
    }

    @Transactional
    public List<CompanyEntity> getAllByIds(Collection<UUID> ids) {
        List<CompanyEntity> companies = companyRepository.findAllById(ids);
        if (companies.size() != ids.size()) {
            throw new EntityNotFoundException(String.format("One of company
was not found"));
        }
        return companies;
    }

}
```

Файл resume-page.component.ts

Реалізація функціональних задач перегляд опублікованих резюме, фільтрація резюме, повідомлення про відсутність результатів, перегляд резюме у спливаючому вікні.

```
import { Component, DestroyRef, inject, OnInit, signal } from
'@angular/core';
import { ResumeFilterComponent } from './components/resume-filter/resume-
filter.component';
import { ResumeTableComponent } from './components/resume-table/resume-
table.component';
import { ResumeService } from './services/resume.service';
import { PublicResume, Resume } from './models/resume.model';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';
import { finalize } from 'rxjs';
import { ResumeFilters } from './models/resume-filters.model';
import { TranslateModule, TranslateService } from '@ngx-translate/core';

@Component({
  selector: 'app-resume-page',
  templateUrl: './resume-page.component.html',
  styleUrl: './resume-page.component.scss',
  standalone: true,
  imports: [ResumeFilterComponent, ResumeTableComponent, TranslateModule],
})
export class ResumePageComponent implements OnInit {
  private resumeService = inject(ResumeService);
  private destroyRef = inject(DestroyRef);
  private translateService = inject(TranslateService);
  private initialLoadComplete = false;
  private lastLoadedFilters: string = '';

  protected readonly Math = Math;

  resumes = signal<PublicResume[]>([]);
  totalCount = signal<number>(0);
  isLoading = signal<boolean>(false);
  error = signal<string | null>(null);
  filters = signal<ResumeFilters>({
    companyId: '',
    speciality: '',
    isHrScreeningPassed: null,
    yearOfExperienceRange: {
      min: null,
      max: null,
    },
    date: '',
    page: 1,
    pageSize: 10,
    orderBy: 'Date',
    sortOrder: 'Descending',
  });

  ngOnInit(): void {
    this.loadResumes(this.filters());
  }

  onFilterApplied(newFilters: ResumeFilters): void {

    this.filters.update(currentFilters => ({
      ...currentFilters,
      ...newFilters,
      page: 1,
    }));

    this.loadResumes(this.filters());
  }

  onPageChanged(pageEvent: { pageIndex: number, pageSize: number }): void {

    if (!this.initialLoadComplete) {
      return;
    }

    const currentFilters = this.filters();

    if ((pageEvent.pageIndex === 0 && pageEvent.pageSize === 10 &&
```

```
          currentFilters.page === 1 && currentFilters.pageSize === 10) ||
        (pageEvent.pageIndex + 1 === currentFilters.page &&
         pageEvent.pageSize === currentFilters.pageSize)) {
      return;
    }

    this.filters.update(currentFilters => ({
      ...currentFilters,
      page: pageEvent.pageIndex + 1,
      pageSize: pageEvent.pageSize,
    }));

    this.loadResumes(this.filters());
  }

  private loadResumes(filters: ResumeFilters): void {
    if (this.isLoading()) {
      return;
    }

    const filtersString = JSON.stringify(filters);

    if (this.initialLoadComplete && filtersString === this.lastLoadedFilters)
{
      return;
    }

    this.isLoading.set(true);
    this.error.set(null);

    this.lastLoadedFilters = filtersString;

    this.resumeService.getResumes(filters)
      .pipe(
        takeUntilDestroyed(this.destroyRef),
        finalize(() => {
          this.isLoading.set(false);
          this.initialLoadComplete = true;
        }),
      )
      .subscribe({
        next: (response) => {
          this.resumes.set(response.data);
          this.totalCount.set(response.totalCount);
        },
        error: (err) => {
          console.error('[ResumePageComponent] Error loading resumes:', err);
          this.error.set(this.translateService.instant('resume.loadError'));
        },
      });
  }
}
```

Файл user-resumes.service.ts

Реалізація функціональних задач створення резюме, перегляд власних резюме у профілі, зміна статусу резюме.

```
import { inject, Injectable } from '@angular/core';
import { ResumeFormData, CompanyStatusInfo } from '../models/resume-form-
data';
import { Observable } from 'rxjs';
import { CreateResumeModel, CompanyResumeInfo } from '../models/create-
resume.model';
import { ApiService } from '@app/core/services/api.service';
import { PrivateResume, ResumeStatusEnum } from '../models/resume.model';

@Injectable({
  providedIn: 'root',
})
export class UserResumesService {
  private apiService = inject(ApiService);
  private readonly apiEndpoint = '/resumes';
  constructor() {}

  addResume(resumeData: ResumeFormData): Observable<any> {
    const companiesInfo: CompanyResumeInfo[] =
resumeData.companies.map(company => ({
      id: company.companyId,
      isHrScreeningPassed: company.status === 'Approved'
```

```
    }));

    const apiRequest: CreateResumeModel = {
      companies: companiesInfo,
      yearsOfExperience: resumeData.yearsOfExperience,
      speciality: resumeData.specialization,
      document: resumeData.file,
    };

    const formData = this.apiService.createFormData(apiRequest);

    return this.apiService.post<FormData, any>(this.apiEndpoint, formData);
  }

  getResumes(): Observable<PrivateResume[]> {
    return this.apiService.get<PrivateResume[]>(this.apiEndpoint);
  }

  updateResumeStatus(resumeId: string, status: ResumeStatusEnum):
Observable<any> {
    const requestBody = {
      resumeEvent: status === ResumeStatusEnum.APPROVED ? 'APPROVED' :
'REJECTED'
    };

    return this.apiService.patch<any, any>(`${this.apiEndpoint}/${resumeId}`,
requestBody);
  }
}
```

Файл company-autocomplete.component.ts

Компонент автозаповнення для вибору компанії. Реалізація функціональних задач створення резюме.

```
import { Component, DestroyRef, ElementRef, Input, ViewChild, forwardRef,
inject, signal, computed, resource } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatAutocompleteModule, MatAutocomplete } from
'@angular/material/autocomplete';
import { MatInputModule } from '@angular/material/input';
import {
  ReactiveFormsModule,
  FormsModule,
  ControlValueAccessor,
  NG_VALUE_ACCESSOR,
  FormControl
} from '@angular/forms';
import { MatIconModule } from '@angular/material/icon';
import { IconComponent } from '../icon/icon.component';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';
import { CompanyService } from '../../core/services/company.service';
import { Company } from '../../core/models/company.model';
import { distinctUntilChanged, firstValueFrom } from 'rxjs';
import { waitResource } from '@app/core/utils/wait-resource';
import { TranslateModule } from '@ngx-translate/core';

@Component({
  selector: 'app-company-autocomplete',
  standalone: true,
  imports: [
    CommonModule,
    MatFormFieldModule,
    MatAutocompleteModule,
    MatInputModule,
    ReactiveFormsModule,
    FormsModule,
    MatIconModule,
    IconComponent,
    TranslateModule,
  ],
  templateUrl: './company-autocomplete.component.html',
  styleUrls: ['./company-autocomplete.component.scss'],
  providers: [{
    provide: NG_VALUE_ACCESSOR,
    useExisting: forwardRef(() => CompanyAutocompleteComponent),
    multi: true,
  },
  CompanyService
],
```

```
})
export class CompanyAutocompleteComponent implements ControlValueAccessor {
  @Input() required = false;
  @Input() errorMessage = 'Please select a company';
  @Input() label?: string;

  @ViewChild('autocompleteTrigger') autocompleteTrigger!: ElementRef;
  @ViewChild('auto') autocomplete!: MatAutocomplete;

  private readonly destroyRef = inject(DestroyRef);
  private readonly companyService = inject(CompanyService);

  public isOpen = signal<boolean>(false);
  public searchTerm = signal<string>('');
  public selectedCompany = signal<Company | null>(null);
  public touched = signal<boolean>(false);

  public searchControl = new FormControl<string>('');

  public companiesResource = resource({
    request: () => {
      const term = this.searchTerm();
      return term && term.length > 2 ? term : undefined;
    },
    loader: async ({ request, abortSignal }) => {
      if (!request) return [];

      try {
        await waitResource(500, abortSignal);

        return await
firstValueFrom(this.companyService.searchCcompanies(request));
      } catch (error) {
        if (error instanceof Error && error.message === 'Operation aborted')
{
          return [];
        }

        console.error('Error fetching companies:', error);
        return [];
      }
    }
  });

  public hasError = computed(() =>
    this.required && this.touched() && !this.selectedCompany()
  );

  private onChange: (value: Company | null) => void = () => {};
  private onTouched: () => void = () => {};

  constructor() {
    // Set up the value changes from the searchControl without debounce
    // since debouncing is now handled in the resource
    this.searchControl.valueChanges
      .pipe(
        takeUntilDestroyed(this.destroyRef),
        distinctUntilChanged()
      )
      .subscribe(value => {
        if (value !== null) {
          this.searchTerm.set(value);
        }
      });
  }

  public writeValue(value: Company | null): void {
    this.selectedCompany.set(value);
    if (value) {
      this.searchControl.setValue(value.name, { emitEvent: false });
    } else {
      this.searchControl.setValue('', { emitEvent: false });
    }
  }

  public registerOnChange(fn: (value: Company | null) => void): void {
    this.onChange = fn;
  }

  public registerOnTouched(fn: () => void): void {
    this.onTouched = fn;
```

```
  }

  public setDisabledState(isDisabled: boolean): void {
    if (isDisabled) {
      this.searchControl.disable();
    } else {
      this.searchControl.enable();
    }
  }

  public hasValue(): boolean {
    return !!this.selectedCompany();
  }

  public clearSelection(event: MouseEvent): void {
    event.stopPropagation();

    this.selectedCompany.set(null);
    this.searchControl.setValue('');

    this.onChange(null);
    this.markAsTouched();
  }

  public onSearchInputChange(event: Event): void {
    const value = (event.target as HTMLInputElement).value;
    this.searchTerm.set(value);

    // If the search term doesn't match the selected company name, clear
selection
    if (this.selectedCompany() && this.selectedCompany()!.name !== value) {
      this.selectedCompany.set(null);
      this.onChange(null);
    }
  }

  public selectCompany(company: Company): void {
    this.selectedCompany.set(company);
    this.searchControl.setValue(company.name);
    this.onChange(company);
    this.markAsTouched();
  }

  public markAsTouched(): void {
    this.touched.set(true);
    this.onTouched();
  }

  public onPanelOpened(): void {
    this.isOpen.set(true);
  }

  public onPanelClosed(): void {
    this.isOpen.set(false);
    this.markAsTouched();
  }

  public refreshCompanies(): void {
    this.companiesResource.reload();
  }
}
```

Файл auth.component.ts

Реалізація функціональних задач Реєстрація користувача, Перенаправлення після реєстрації, Авторизація користувача.

```
import { Component, inject, OnInit, ViewEncapsulation } from '@angular/core';
import {
  FormBuilder,
  FormGroup,
  ReactiveFormsModule,
  ValidatorFn,
} from '@angular/forms';
import { ActivatedRoute, Router, RouterLink } from '@angular/router';

import { take } from 'rxjs';
import { InputComponent } from '../../reusable/input/input.component';
import { ButtonComponent } from '../../reusable/button/button.component';
```

```
import { IconComponent } from '../../reusable/icon/icon.component';
import { AuthService } from '../../core/services/auth.service';
import { ToasterService } from '../../core/services/toaster.service';
import { forgotPasswordConfig, loginConfig, registerConfig } from
'../../core/constants/auth-page-configs.constants';
import { AuthPageConfig } from '../../core/models/auth-page-config.model';
import { UserRoleEnum } from '../../core/enums/user-role.enum';
import { passwordConfirmationValidator } from '../../core/utils/password-
confirmation.validator';

@Component({
  selector: 'app-auth',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    InputComponent,
    ButtonComponent,
    IconComponent,
    RouterLink,
  ],
  templateUrl: './auth.component.html',
  styleUrl: './auth.component.scss',
  encapsulation: ViewEncapsulation.None,
})
export class AuthComponent implements OnInit {
  private fb = inject(FormBuilder);
  private route = inject(ActivatedRoute);
  private authService = inject(AuthService);
  private router = inject(Router);
  private toasterService = inject(ToasterService);

  public config!: AuthPageConfig;
  public authForm!: FormGroup;

  public ngOnInit(): void {
    this.route.data.subscribe(data => {
      this.config = data['config'];
      this.initializeForm();
    });
  }

  private initializeForm(): void {
    if (!this.config) {
      return;
    }

    const formControls: Record<string, [string, ValidatorFn[]]> = {};

    this.config?.fields?.forEach(field => {
      formControls[field.name] = ['', field.validators || []];
    });
    this.authForm = this.fb.group(formControls);

    // Add password confirmation validator for register form
    if (this.config === registerConfig) {
      this.authForm.addValidators(passwordConfirmationValidator());
    }
  }

  public onSubmit(): void {
    if (!this.authForm.valid) {
      return;
    }

    switch (this.config) {
      case loginConfig:
        this.handleLogin();
        break;

      case registerConfig:
        this.handleRegister();
        break;

      case forgotPasswordConfig:
        this.handleForgotPassword();
        break;

      default:
        console.warn(
          'No matching config logic found. Form data:',
          this.authForm.value,
```

```
        );
        break;
    }
  }

  private handleLogin(): void {
    const { email, password } = this.authForm.value;

    this.authService.login(email, password).pipe(take(1)).subscribe({
      next: (response) => {
        const userRole = this.authService.userRole;

        // Check if user is admin and redirect accordingly
        if (userRole === UserRoleEnum.ADMIN) {
          this.router.navigate(['/admin']);
        } else {
          const redirectUrl = this.authService.redirectUrl;
          if (redirectUrl && !redirectUrl.includes('/admin')) {
            this.router.navigateByUrl(redirectUrl);
          } else {
            this.router.navigate(['/']);
          }
        }

        // Clear any stored redirectUrl
        this.authService.setRedirectUrl(null);
      },
    });
  }

  private handleRegister(): void {
    const { email, password } = this.authForm.value;

    const registerData = {
      email,
      password,
    };

    this.authService.register(registerData).pipe(take(1)).subscribe({
      next: () => {
        this.toasterService.showSuccess('Registration successful! Please log
in with your credentials.');
        this.router.navigate(['/login']);
      },
    });
  }

  private handleForgotPassword(): void {
    const { email } = this.authForm.value;

    this.authService.resetPassword(email).pipe(take(1)).subscribe({
      next: () => {
        this.toasterService.showSuccess('Password reset instructions have
been sent to your email');
        this.router.navigate(['/login']);
      },
      error: (error) => {
        this.toasterService.showError(error.message || 'Failed to send reset
instructions');
      },
    });
  }
}
```

Файл admin-resume-detail-page.component.ts

Реалізація функціональних задач перегляд деталей резюме адміністратором, зміна статусу резюме, завантаження документа з прихованими даними.

```
import { Component, OnInit, inject, signal, computed, DestroyRef } from
'@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { DatePipe, NgClass } from '@angular/common';
import { ButtonComponent } from '../../../reusable/button/button.component';
import { DomSanitizer, SafeResourceUrl } from '@angular/platform-browser';
import { PrivateResume, ResumeStatusEnum } from
'../../resume/models/resume.model';
import { AdminResumeStateService } from '../services/admin-resume-
state.service';
import { UserResumesService } from '../../resume/services/user-
resumes.service';
```

```
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

@Component({
  selector: 'app-admin-resume-detail-page',
  templateUrl: './admin-resume-detail-page.component.html',
  styleUrl: './admin-resume-detail-page.component.scss',
  standalone: true,
  imports: [
    ButtonComponent,
    DatePipe,
    NgClass
  ],
})
export class AdminResumeDetailPageComponent implements OnInit {
  private route = inject(ActivatedRoute);
  private router = inject(Router);
  private sanitizer = inject(DomSanitizer);
  private adminResumeStateService = inject(AdminResumeStateService);
  private userResumesService = inject(UserResumesService);
  private destroyRef = inject(DestroyRef);

  resume = signal<PrivateResume | null>(null);
  error = signal<string | null>(null);
  isUpdating = signal<boolean>(false);

  privateDocumentUrl = computed(() => {
    const resume = this.resume();
    return resume ? resume.getPrivateDocumentUrl() : null;
  });

  publicDocumentUrl = computed(() => {
    const resume = this.resume();
    return resume ? resume.getPublicDocumentUrl() : null;
  });

  safePrivateDocUrl = computed<SafeResourceUrl>(() => {
    return this.privateDocumentUrl()
      ?
this.sanitizer.bypassSecurityTrustResourceUrl(this.privateDocumentUrl()!)
      : this.sanitizer.bypassSecurityTrustResourceUrl('');
  });

  safePublicDocUrl = computed<SafeResourceUrl>(() => {
    return this.publicDocumentUrl()
      ?
this.sanitizer.bypassSecurityTrustResourceUrl(this.publicDocumentUrl()!)
      : this.sanitizer.bypassSecurityTrustResourceUrl('');
  });

  resumeStatus = computed(() => {
    return this.resume()?.resumeStatus ||
ResumeStatusEnum.WAITING_FOR_APPROVE;
  });

  ngOnInit(): void {
    const resumeId = this.route.snapshot.paramMap.get('id');
    if (!resumeId) {
      this.error.set('Resume ID not found');
      return;
    }

    const selectedResume = this.adminResumeStateService.selectedResume();

    if (selectedResume && selectedResume.id === resumeId) {
      this.resume.set(selectedResume);
    } else {
      this.error.set('Resume not found');
      this.navigateBack();
    }
  }

  navigateBack(): void {
    this.router.navigate(['/admin']);
  }

  approveResume(): void {
    if (this.resume()) {
      this.isUpdating.set(true);

      this.userResumesService.updateResumeStatus(
        this.resume()!.id,
```

```
        ResumeStatusEnum.APPROVED
      )
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe({
        next: () => {
          const currentResume = this.resume()!;

          const updatedResume = new PrivateResume(
            currentResume.id,
            currentResume.documents,
            currentResume.companies,
            currentResume.speciality,
            currentResume.yearsOfExperience,
            currentResume.createdAt,
            ResumeStatusEnum.APPROVED,
            currentResume.hidden
          );

          this.resume.set(updatedResume);
          this.adminResumeStateService.setSelectedResume(updatedResume);
          this.isUpdating.set(false);
        },
        error: (err) => {
          this.error.set('Failed to approve resume');
          this.isUpdating.set(false);
          console.error('Error approving resume:', err);
        }
      });
    }
  }

  rejectResume(): void {
    if (this.resume()) {
      this.isUpdating.set(true);

      this.userResumesService.updateResumeStatus(
        this.resume()!.id,
        ResumeStatusEnum.REJECTED
      )
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe({
        next: () => {
          const currentResume = this.resume()!;

          const updatedResume = new PrivateResume(
            currentResume.id,
            currentResume.documents,
            currentResume.companies,
            currentResume.speciality,
            currentResume.yearsOfExperience,
            currentResume.createdAt,
            ResumeStatusEnum.REJECTED,
            currentResume.hidden
          );

          this.resume.set(updatedResume);
          this.adminResumeStateService.setSelectedResume(updatedResume);
          this.isUpdating.set(false);
        },
        error: (err) => {
          this.error.set('Failed to reject resume');
          this.isUpdating.set(false);
          console.error('Error rejecting resume:', err);
        }
      });
    }
  }

  uploadEditedDocument(): void {
    console.log('Upload edited document functionality');
  }

  getFormattedDate(date: Date | null | undefined): string {
    if (!date) return 'N/A';
    const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec'];
    return `${months[date.getMonth()]} ${date.getDate()},
${date.getFullYear()}`;
  }

  getStatusClass(): string {
```

```
    switch (this.resumeStatus()) {
      case ResumeStatusEnum.APPROVED:
        return 'status-approved';
      case ResumeStatusEnum.REJECTED:
        return 'status-rejected';
      case ResumeStatusEnum.WAITING_FOR_APPROVE:
        return 'status-waiting';
      default:
        return '';
    }
  }

  formatStatusText(): string {
    switch (this.resumeStatus()) {
      case ResumeStatusEnum.APPROVED:
        return 'Approved';
      case ResumeStatusEnum.REJECTED:
        return 'Rejected';
      case ResumeStatusEnum.WAITING_FOR_APPROVE:
        return 'Waiting for Approval';
      default:
        return this.resumeStatus() ? String(this.resumeStatus()) : 'Unknown';
    }
  }
}
```

Файл admin-resume-state.service.ts

```
import { Injectable, signal } from '@angular/core';
import { PrivateResume } from '../../resume/models/resume.model';

@Injectable({
  providedIn: 'root'
})
export class AdminResumeStateService {
  private readonly STORAGE_KEY = 'adminSelectedResume';
  private selectedResumeSignal = signal<PrivateResume |
null>(this.getStoredResume());

  get selectedResume() {
    return this.selectedResumeSignal;
  }

  setSelectedResume(resume: PrivateResume) {
    this.selectedResumeSignal.set(resume);
    this.storeResume(resume);
  }

  clearSelectedResume() {
    this.selectedResumeSignal.set(null);
    sessionStorage.removeItem(this.STORAGE_KEY);
  }

  private storeResume(resume: PrivateResume): void {
    try {
      const resumeJson = resume.toJson();
      const resumeString = JSON.stringify(resumeJson);
      sessionStorage.setItem(this.STORAGE_KEY, resumeString);
    } catch (e) {
      console.error('Error storing admin resume in session storage:', e);
    }
  }

  private getStoredResume(): PrivateResume | null {
    try {
      const storedResume = sessionStorage.getItem(this.STORAGE_KEY);
      if (!storedResume) {
        return null;
      }

      const resumeData = JSON.parse(storedResume);
      return PrivateResume.fromJson(resumeData);
    } catch (e) {
      console.error('Error parsing stored admin resume:', e);
      sessionStorage.removeItem(this.STORAGE_KEY);
      return null;
    }
  }
}
```

Файл resume-detail-comments.component.ts

Реалізація функціональних задач коментування резюме, відповіді на коментарі, голосування за коментарі, оновлення рейтингу коментарів, сортування коментарів.

```
import { Component, DestroyRef, Input, OnChanges, SimpleChanges, inject,
signal } from '@angular/core';
import { FormControl, FormGroup, FormsModule, ReactiveFormsModule, Validators
} from '@angular/forms';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';
import { HttpErrorResponse } from '@angular/common/http';
import { DatePipe, NgClass, NgFor, NgIf } from '@angular/common';
import { ButtonComponent } from
'../../../../reusable/button/button.component';
import { CommentsService } from '../../services/comments.service';
import { Comment, CommentCreateRequest, CommentVoteRequest } from
'../../models/comment.model';
import { CommentItemComponent } from './comment-item.component';

interface CommentForm {
  text: FormControl<string>;
}

@Component({
  selector: 'app-resume-detail-comments',
  templateUrl: './resume-detail-comments.component.html',
  styleUrl: './resume-detail-comments.component.scss',
  standalone: true,
  imports: [
    ReactiveFormsModule,
    FormsModule,
    DatePipe,
    ButtonComponent,
    NgFor,
    NgIf,
    NgClass,
    CommentItemComponent
  ]
})
export class ResumeDetailCommentsComponent implements OnChanges {
  private commentsService = inject(CommentsService);
  private destroyRef = inject(DestroyRef);

  @Input() resumeId: string | null = null;

  comments = signal<Comment[]>([]);
  isLoadingComments = signal<boolean>(false);
  commentsError = signal<string | null>(null);

  commentForm = new FormGroup<CommentForm>({
    text: new FormControl('', { nonNullable: true, validators:
[Validators.required] }),
  });

  replyingToComment = signal<Comment | null>(null);

  sortOptions = ['Most Helpful', 'Newest', 'Oldest'];
  selectedSort = signal<string>('Most Helpful');

  ngOnChanges(changes: SimpleChanges): void {
    if (changes['resumeId'] && this.resumeId) {
      this.loadComments(this.resumeId);
    }
  }

  loadComments(resumeId: string): void {
    this.isLoadingComments.set(true);
    this.commentsError.set(null);

    this.commentsService.getCommentsByResumeId(resumeId)
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe({
        next: (response) => {
          const flatComments = Array.isArray(response) ? response :
(response?.data || []);

          const commentMap = new Map<string, Comment>();
```

```
          const rootComments: Comment[] = [];

          flatComments.forEach(comment => {
            // Ensure the comment has a replies array
            comment.replies = [];
            commentMap.set(comment.id, comment);
          });

          flatComments.forEach(comment => {
            if (comment.parentCommentId) {
              const parentComment = commentMap.get(comment.parentCommentId);
              if (parentComment) {
                if (!parentComment.replies) {
                  parentComment.replies = [];
                }
                parentComment.replies.push(comment);
              } else {
                rootComments.push(comment);
              }
            } else {
              rootComments.push(comment);
            }
          });

          this.comments.set(rootComments);
          this.isLoadingComments.set(false);

          this.updateSort(this.selectedSort());
        },
        error: (error: HttpErrorResponse) => {
          console.error('Error loading comments:', error);
          this.commentsError.set('Failed to load comments. Please try
again.');
          this.isLoadingComments.set(false);
        }
      });
  }

  submitComment(): void {
    if (this.commentForm.invalid || !this.resumeId) {
      return;
    }

    const newComment: CommentCreateRequest = {
      resumeId: this.resumeId,
      text: this.commentForm.controls.text.value,
      parentCommentId: this.replyingToComment()?.id
    };

    this.commentsService.createComment(newComment)
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe({
        next: () => {
          // Reset form
          this.commentForm.reset();
          this.cancelReply();

          // Refresh comments
          if (this.resumeId) {
            this.loadComments(this.resumeId);
          }
        },
        error: (error: HttpErrorResponse) => {
          console.error('Error posting comment:', error);
        }
      });
  }

  replyToComment(comment: Comment): void {
    this.replyingToComment.set(comment);
  }

  cancelReply(): void {
    this.replyingToComment.set(null);
    this.commentForm.reset();
  }

  // Handler for vote events from comment-item component
  handleVote(event: {comment: Comment, voteState: 'UP' | 'DOWN'}): void {
    const { comment, voteState } = event;
    this.commentsService.voteOnComment({
```

```
      commentId: comment.id,
      voteState: voteState
    })
    .pipe(takeUntilDestroyed(this.destroyRef))
    .subscribe({
      next: () => {
        // Refresh comments to get updated vote counts
        if (this.resumeId) {
          this.loadComments(this.resumeId);
        }
      },
      error: (error: HttpErrorResponse) => {
        console.error('Error voting on comment:', error);
      }
    });
  }

  // Handler for reply submission from nested comments
  handleSubmitReply(event: {parentId: string, text: string}): void {
    if (!this.resumeId) {
      return;
    }

    const newComment: CommentCreateRequest = {
      resumeId: this.resumeId,
      text: event.text,
      parentCommentId: event.parentId
    };

    this.commentsService.createComment(newComment)
      .pipe(takeUntilDestroyed(this.destroyRef))
      .subscribe({
        next: () => {
          this.cancelReply();

          // Refresh comments
          if (this.resumeId) {
            this.loadComments(this.resumeId);
          }
        },
        error: (error: HttpErrorResponse) => {
          console.error('Error posting reply:', error);
        }
      });
  }

  updateSort(sort: string): void {
    this.selectedSort.set(sort);

    // Reorder comments based on sort option
    this.comments.update(rootComments => {
      const sortFunction = (a: Comment, b: Comment) => {
        if (this.selectedSort() === 'Most Helpful') {
          return b.reactionsRate - a.reactionsRate;
        } else if (this.selectedSort() === 'Newest') {
          return new Date(b.createdAt).getTime() - new
Date(a.createdAt).getTime();
        } else { // Oldest
          return new Date(a.createdAt).getTime() - new
Date(b.createdAt).getTime();
        }
      };

      // Sort root comments
      const sortedRoots = [...rootComments].sort(sortFunction);

      // Recursively sort child comments
      const sortReplies = (replies: Comment[]) => {
        if (!replies || replies.length === 0) return [];

        const sortedReplies = [...replies].sort(sortFunction);
        sortedReplies.forEach(reply => {
          if (reply.replies && reply.replies.length > 0) {
            reply.replies = sortReplies(reply.replies);
          }
        });

        return sortedReplies;
      };

      // Apply sorting to all replies
```

```
      sortedRoots.forEach(comment => {
        if (comment.replies && comment.replies.length > 0) {
          comment.replies = sortReplies(comment.replies);
        }
      });

      return sortedRoots;
    });
  }
}
```

Файл main.py

Реалізація функціональної задачі завантаження документа з прихованими даними.

```
from typing import Annotated

from fastapi import (
    Depends,
    FastAPI,
    HTTPException,
    Header,
    UploadFile,
    status,
    Response,
)
import fitz
from presidio_analyzer import AnalyzerEngine

from settings import Settings

settings = Settings()  # type: ignore
app = FastAPI(
    debug=settings.DEBUG,
)

def check_authorization(
    authorization_token: Annotated[str, Header()],
) -> None:
    if authorization_token != settings.API_KEY:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Authorization token is invalid.",
        )

def get_pii(usertext):
    analyzer = AnalyzerEngine()
    detected_pii = analyzer.analyze(text=usertext, language="en")
    return detected_pii

def get_generators(text):
    detected_pii = get_pii(text)
    for pii in detected_pii:
        pii_dict = pii.to_dict()
        pii_text = text[pii_dict["start"] : pii_dict["end"]]
        yield pii_text

@app.post("/")
async def anonymize_document(
    document: UploadFile,
    _is_client_authorized: Annotated[None, Depends(check_authorization)],
):
    doc = fitz.open(stream=await document.read())

    for page in doc:
        page.wrap_contents()
        words_list = page.get_text("words", flags=fitz.TEXT_INHIBIT_SPACES)
        text = ""
        for word in words_list:
            text += word[4] + " "
        for data in get_generators(text):
            areas = page.search_for(data)
            [page.add_redact_annot(area, fill=(0, 0, 0)) for area in areas]
        page.apply_redactions()

    return Response(content=doc.write(), media_type=document.content_type)
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ РЕЗЮМЕ

Програма та методика тестування

КПІ.ІП-1201.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Григорій АВЧАРОВ

Київ – 2025

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є розроблений веб-застосунок, що дозволяє користувачам обмінюватись резюме без персональних даних та обмінюватись досвідом за домогою коментарів. Тестування буде проводитись на операційній системі Windows у браузерах Google Chrome 136.0.7103.114 та Microsoft Edge 136.0.3240.92.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності веб-застосунка з останніми версіями сучасних браузерів Chrome та Microsoft Edge;
- перевірка сумісності застосунку з операційною системою Windows;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись наступні допоміжні засоби:

- Статичний аналіз коду(codequality.browserstack.com);

Порядок проведення тестування буде наступним:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на виведення повідомлень чи помилок, коли це потрібно;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування якості коду;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ РЕЗЮМЕ

Керівництво користувача

КПІ.ІП-1201.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Григорій АВЧАРОВ

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«ShareCV» - це веб-застосунок для резюме без персональних даних. Метою розробки є покращення процесу аналізу резюме, за допомогою якого шукачі роботи зможуть підвищити якість та ефективність своїх резюме, шляхом надання доступу до прикладів без персональних даних, створення середовища для обговорення та обміну досвідом.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність пристрою з встановленим браузером, що підтримує сучасні технології JavaScript ES6 та HTML5;

– наявність доступу до Інтернету;

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний гітхаб-репозиторій. Для цього спершу необхідно склонувати репозиторій на персональний комп’ютер, та задомогою середовища Docker запустити веб-застосунок.

2.3 Перевірка коректної роботи

По завершенню запуску додатка перейшовши в браузер за адресою http://localhost:80/ можливо побачити головну сторінку застосунку. У разі, якщо головна сторінка не відображається, то запуск відбувся не успішно.

3 ВИКОНАННЯ ПРОГРАМИ

3.1 Для неавторизованого користувача

При запуску програмного застосунку користувачу буде відображено головну сторінку, на якій він може авторизуватися у систему й ознайомитись з основним функціоналом (рис 3.1).

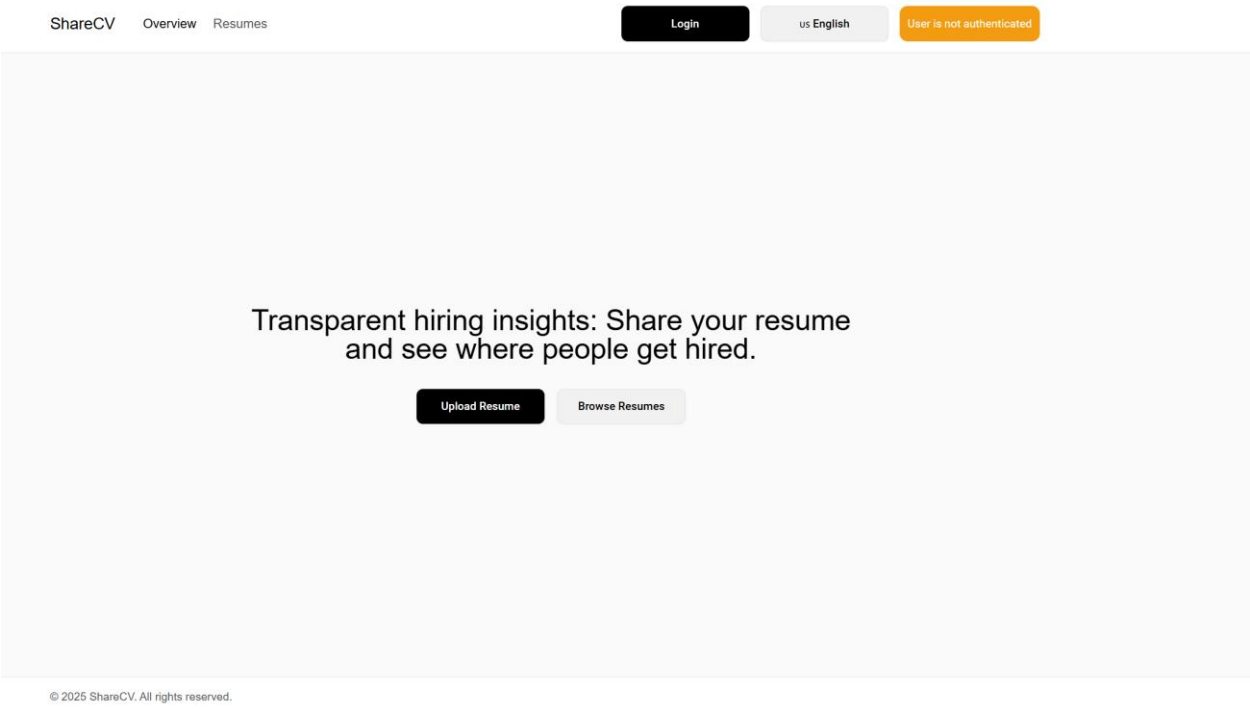


Рисунок 3.1 – Початкова сторінка застосунку

Користувач має змогу ввести адресу перейти на сторінку реєстрації та авторизації(рис 3.2).

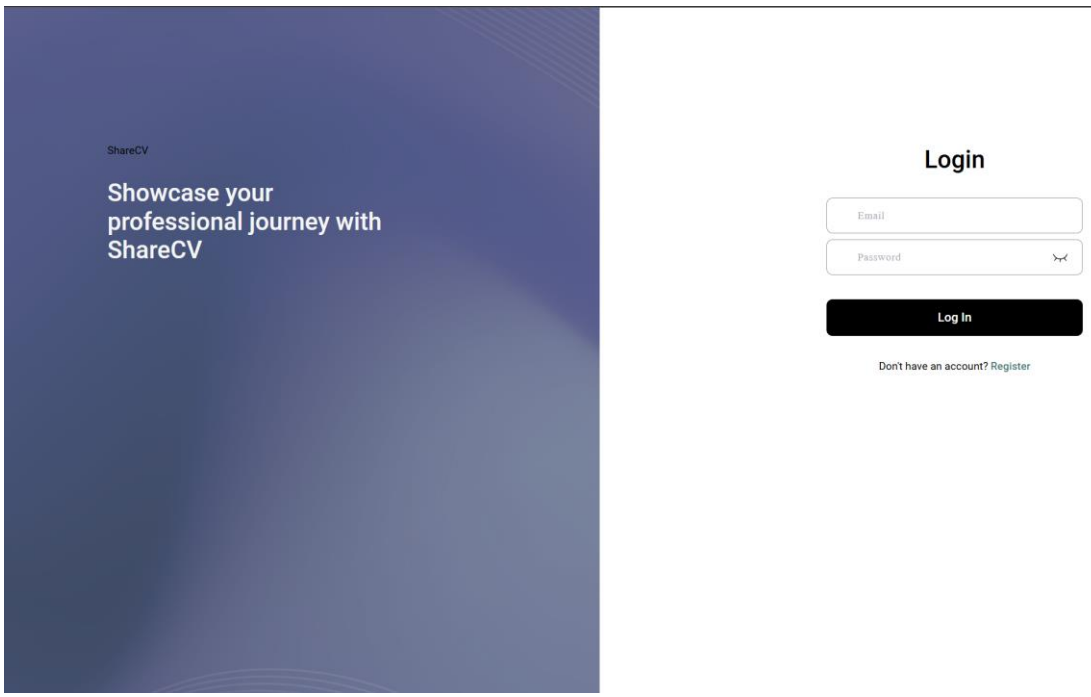


Рисунок 3.2 – Сторінка авторизації

Якщо користувач ще не має облікового запису, він має можливість натиснути кнопку “Don’t have account? Register” та ввести дані для реєстрації.

Паролі мають співпадати і вказаний логін не повинен бути зареєстрованим в системі (рис 3.3).

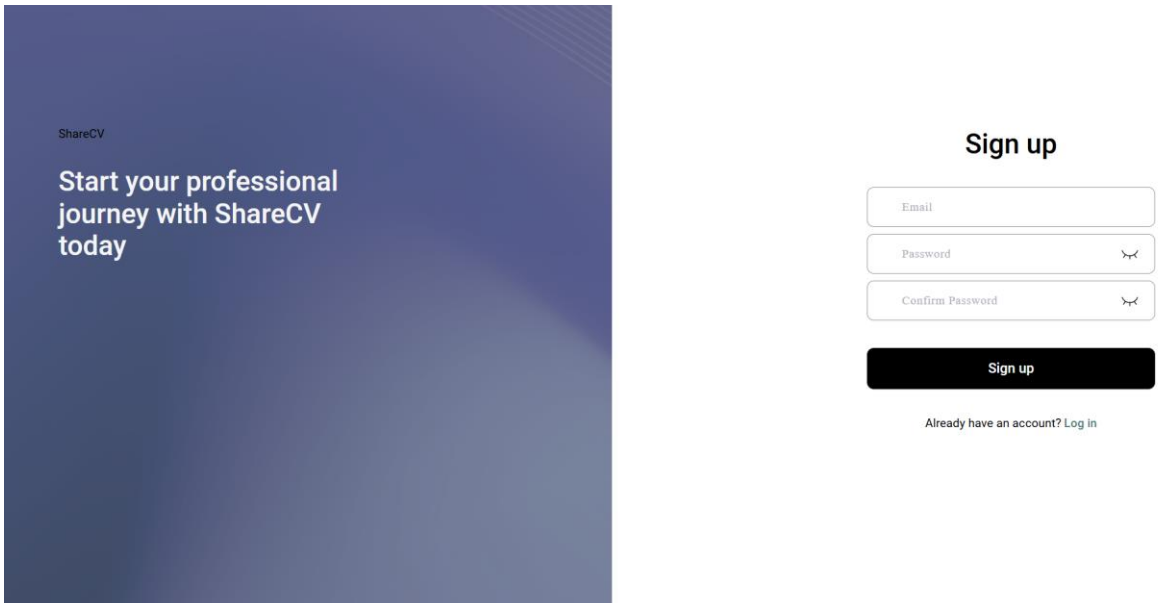


Рисунок 3.3 – Сторінка реєстрації

Після реєстрацій користувач входить зі своїми даними у застосунок, при успішному вході його буде перенаправлено на головну сторінку для подальших дій (рис 3.4).

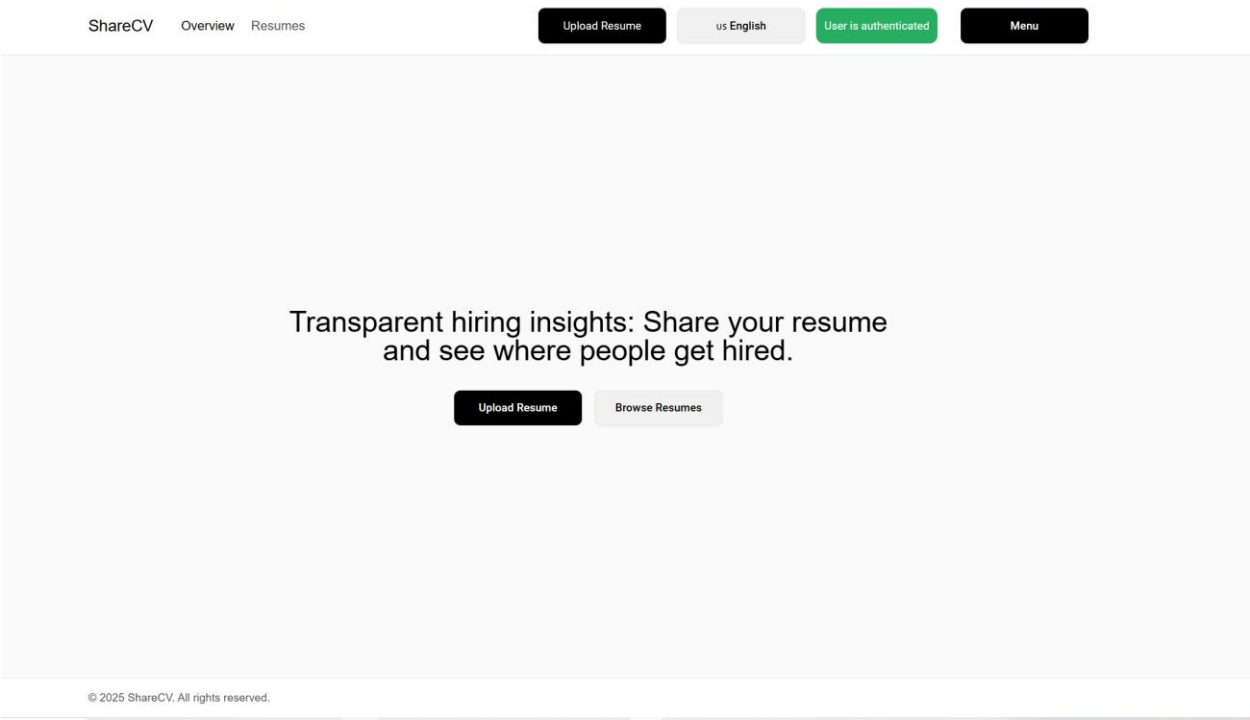


Рисунок 3.4 – Головна сторінка після реєстрації

3.2 Для авторизованого користувача

Після авторизації користувач має можливість відправити своє резюме до обробки натиснувши кнопку “Upload Resume”. Після натискання на кнопку відкривається вікно для завантаження резюме рис(3.5).

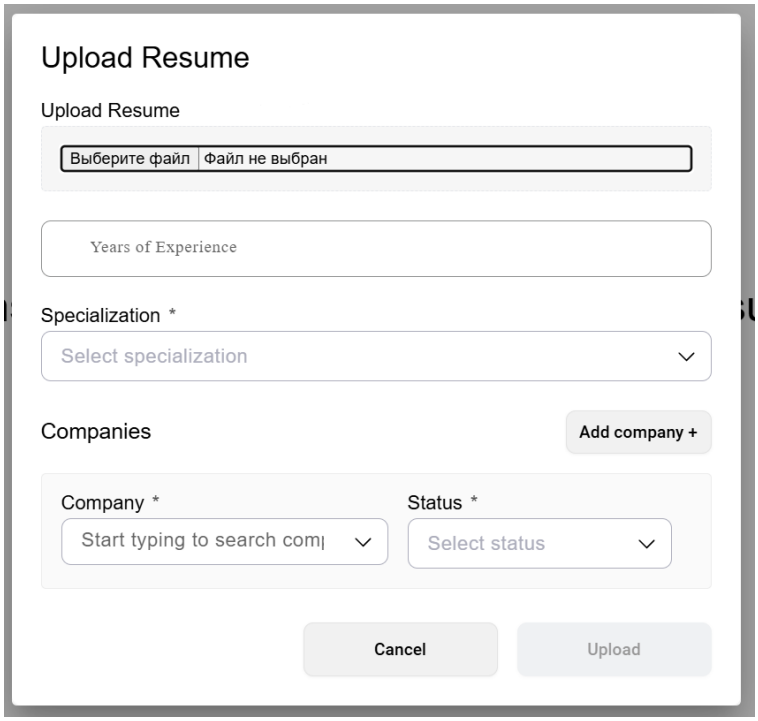


Рисунок 3.5 – Вікно для завантаження резюме

Після завантаження резюме натиснувши на кнопку “Profile” користувач може подивитись свої подані резюме та резюме, інших користувачів, що він додав до обраних (рис 3.6).

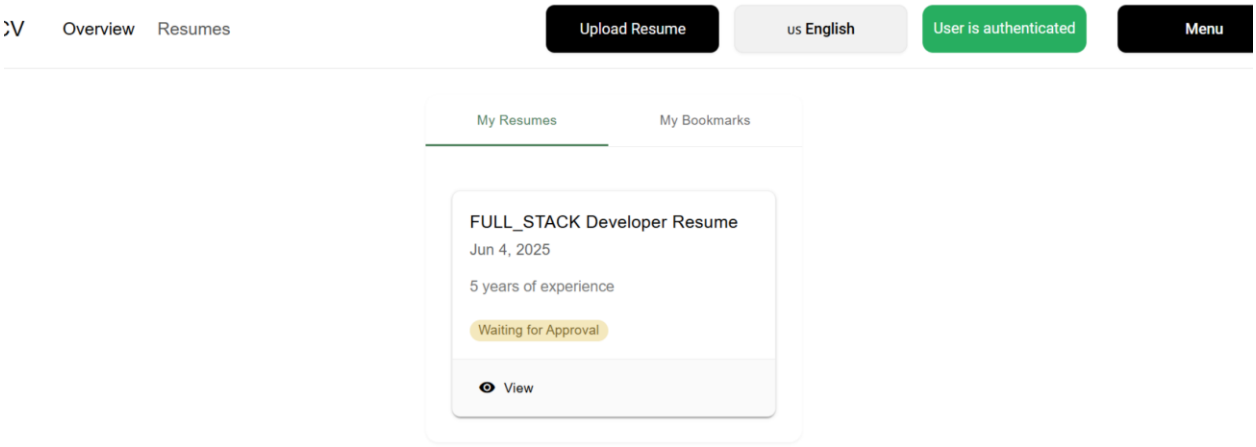


Рисунок 3.6 – Сторінка з резюме користувача та обраними

Після перевірки усі резюме потрапляють на сторінку “Resumes”, на якій користувач може змінити фільтри та перейти детально до кожного резюме (рис 3.7).

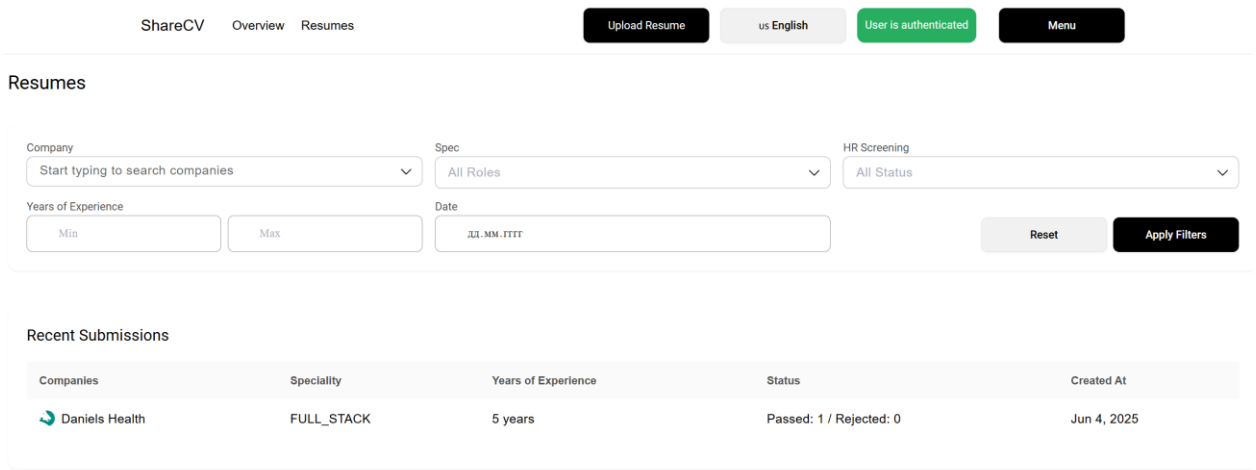


Рисунок 3.7 – Сторінка опублікованих резюме

Натиснувши на рядок у таблиці з опублікованим резюме користувач потрапляє на детальну сторінку резюме, де може переглянути усі деталі, компанії в які резюме пройшло й не пройшло відбір, та секцію коментарів, де користувач може віддати свій голос за будь-який коментар(тільки один голос), чи написати свій коментар, а також зберегти резюме у обране (рис 3.8).

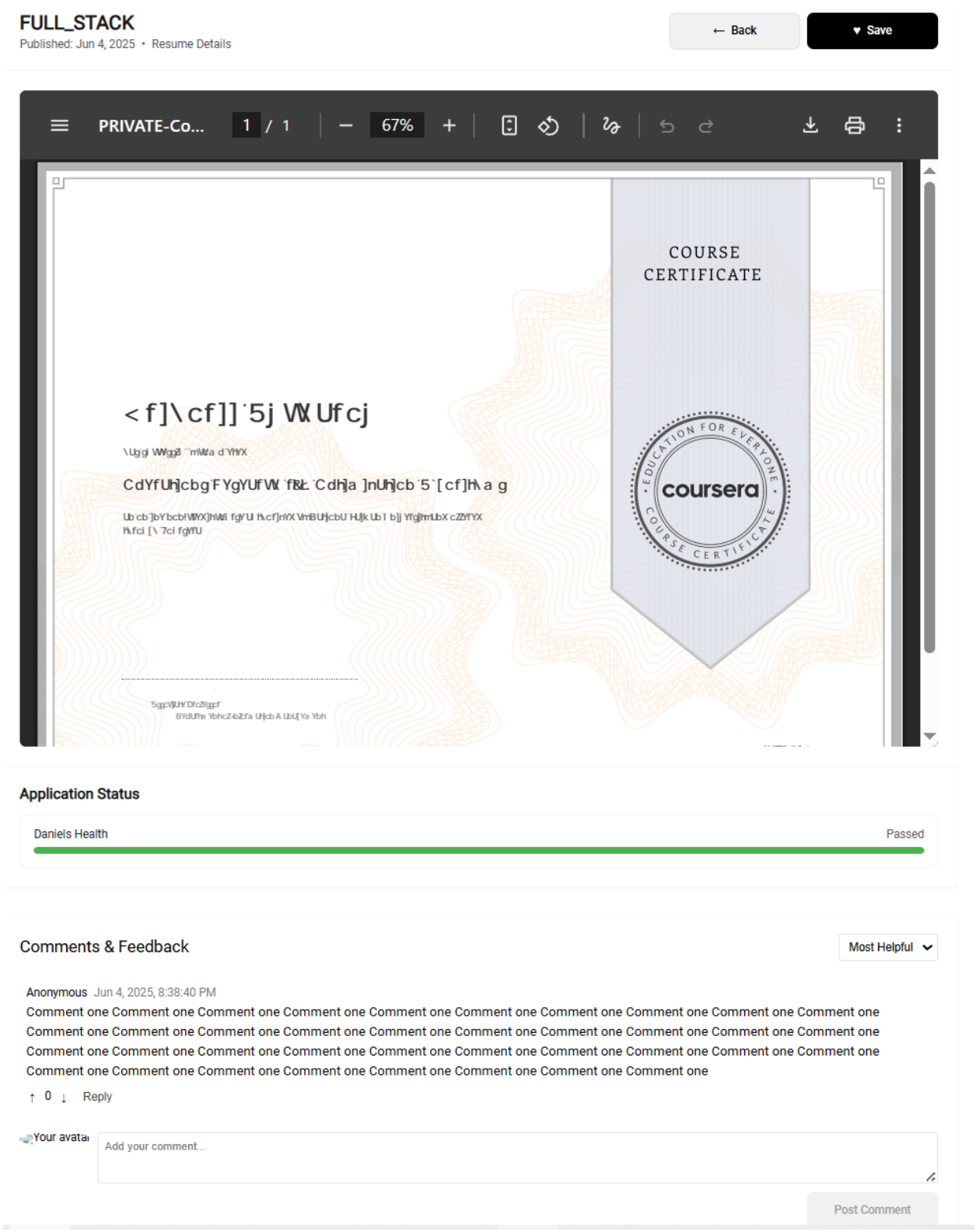


Рисунок 3.8 – Детальна сторінка опублікованого резюме

Якщо користувач бажає видалити усі дані свого облікового запису, то він має можливість це зробити за допомогою кнопки “Delete account” (рис 3.9).

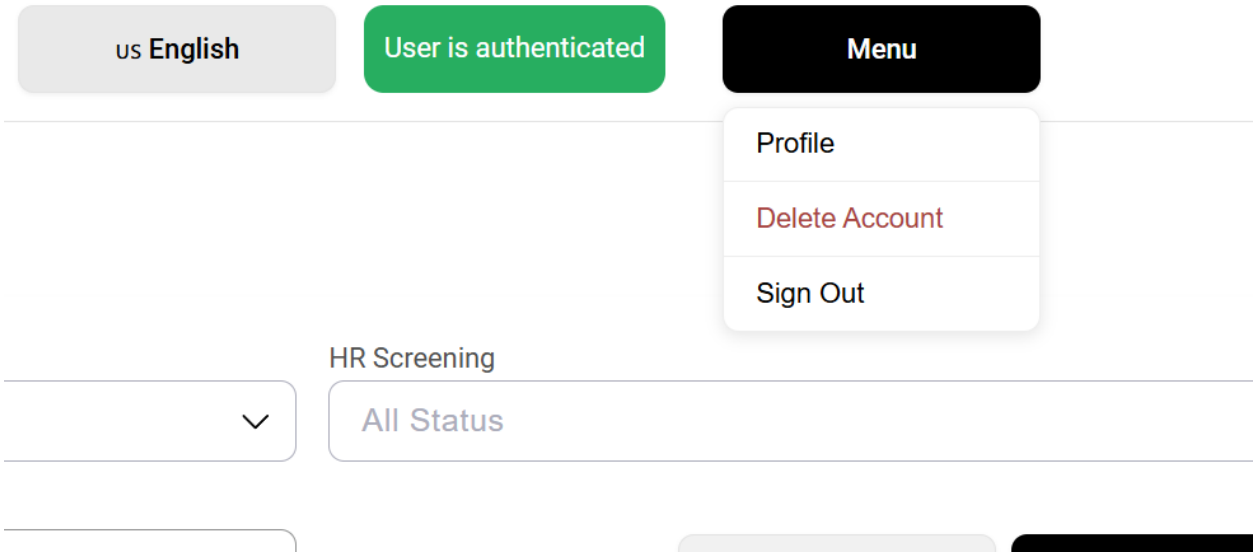


Рисунок 3.9 – Видалення облікового запису

3.3 Для адміністратора

Після успішної авторизації користувач з роллю адміністратора перенаправляється на сторінку з поданими резюме (рис 3.10).

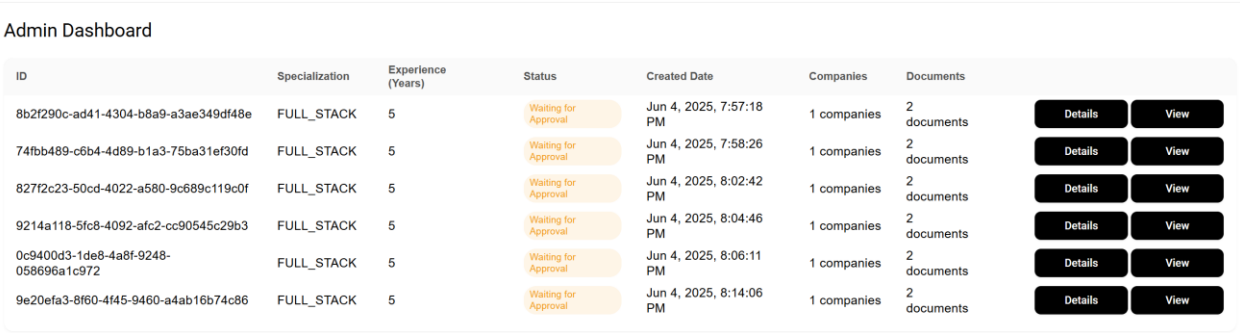


Рисунок 3.10 – Сторінка поданих резюме

Натиснувши на кнопку “Details” у таблиці з поданими резюме адміністратору відкривається детальна сторінка резюме, де адміністратор може детально проаналізувати подане резюме (рис 3.11).

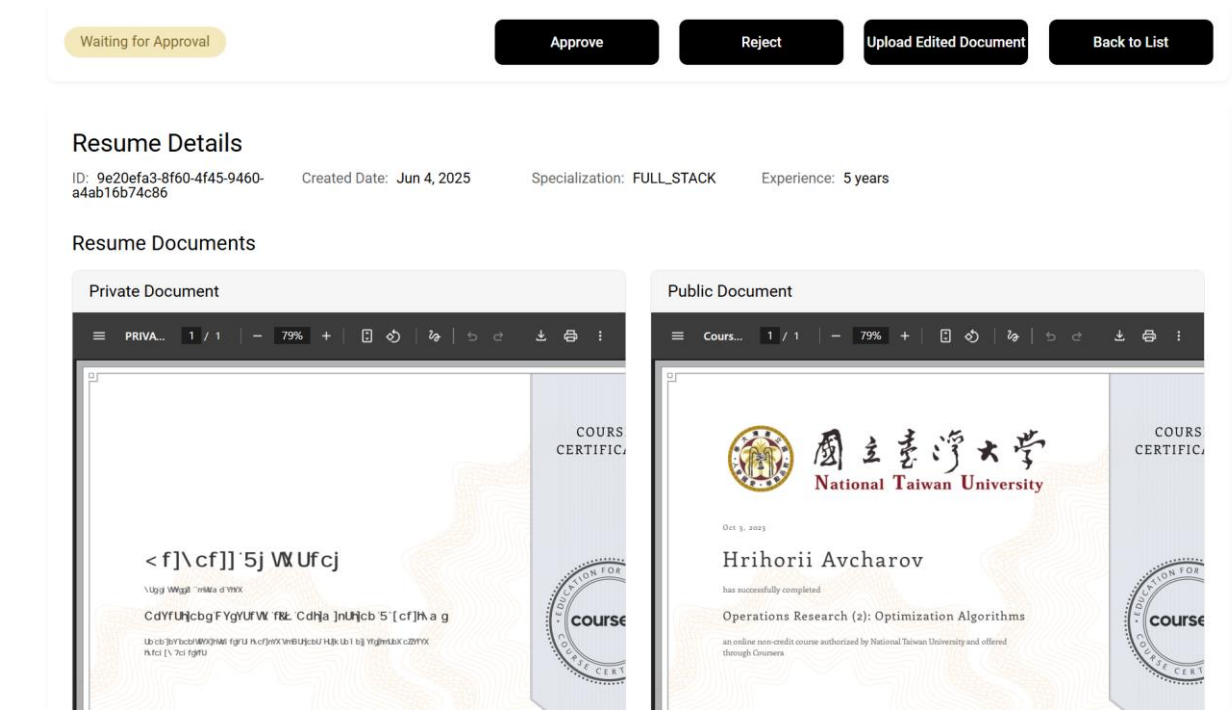


Рисунок 3.11 – Детальна сторінка поданого резюме

Натиснувши на кнопку “Approve” або “Reject” статус резюме буде змінено відповідно до “Approved” або “Rejected” в залежності від вибраного варіанту. Також при натисканні на кнопку “Upload edited document” адміністратор може завантажити відредаговане резюме, якщо він знайшов помилки в прибиранні даних (рис 3.12).

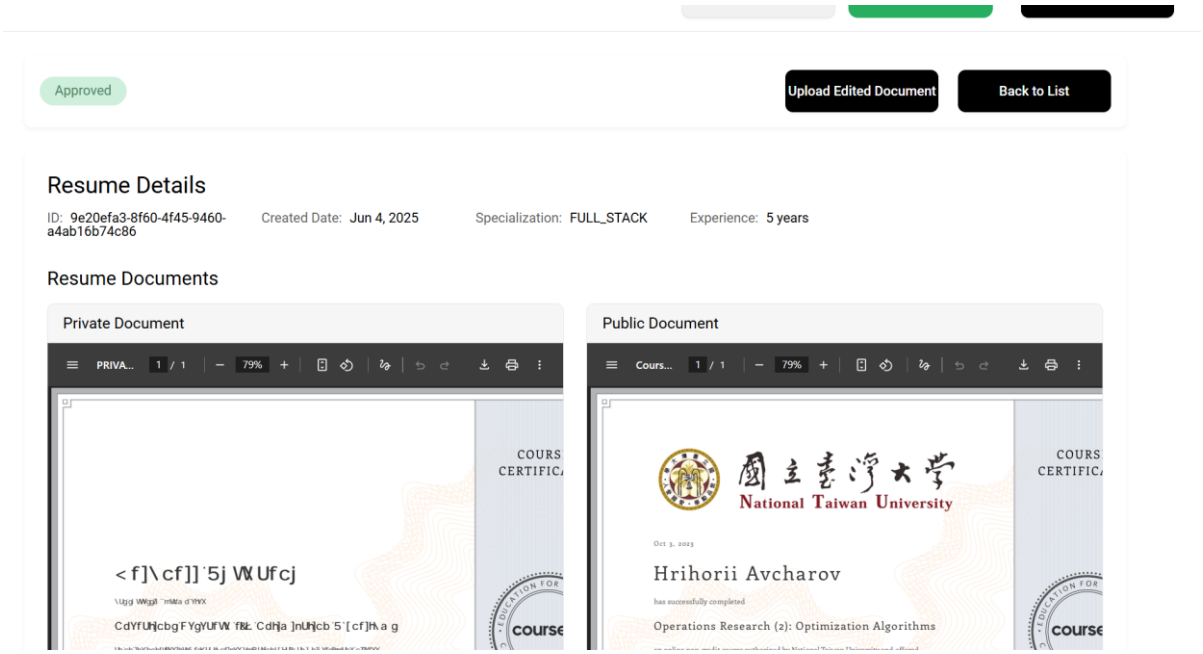


Рисунок 3.12 – Сторінка детального резюме після зміни статусу

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ РЕЗЮМЕ

Графічний матеріал

КПІ.ІП-1201.045440.06.99

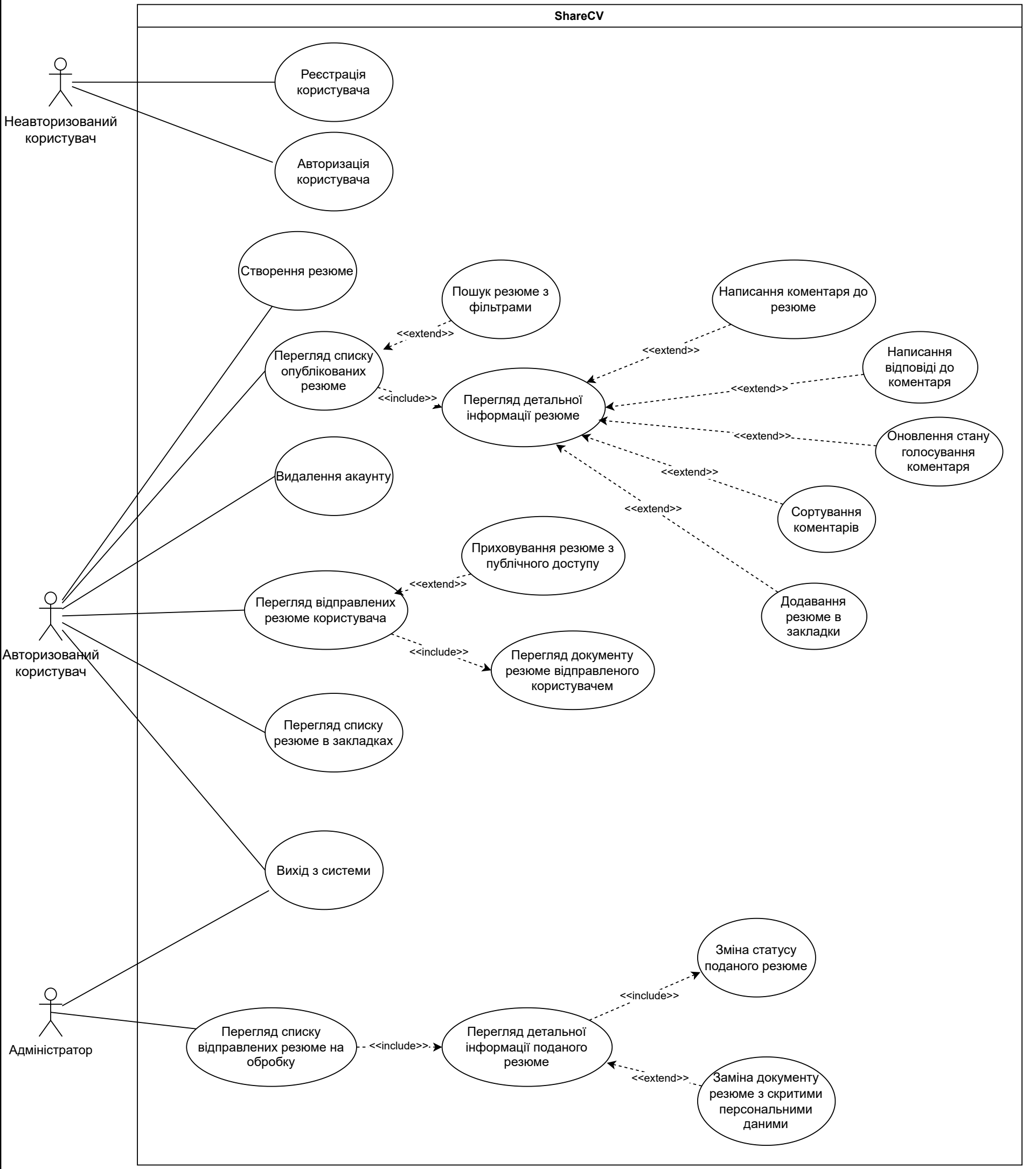
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

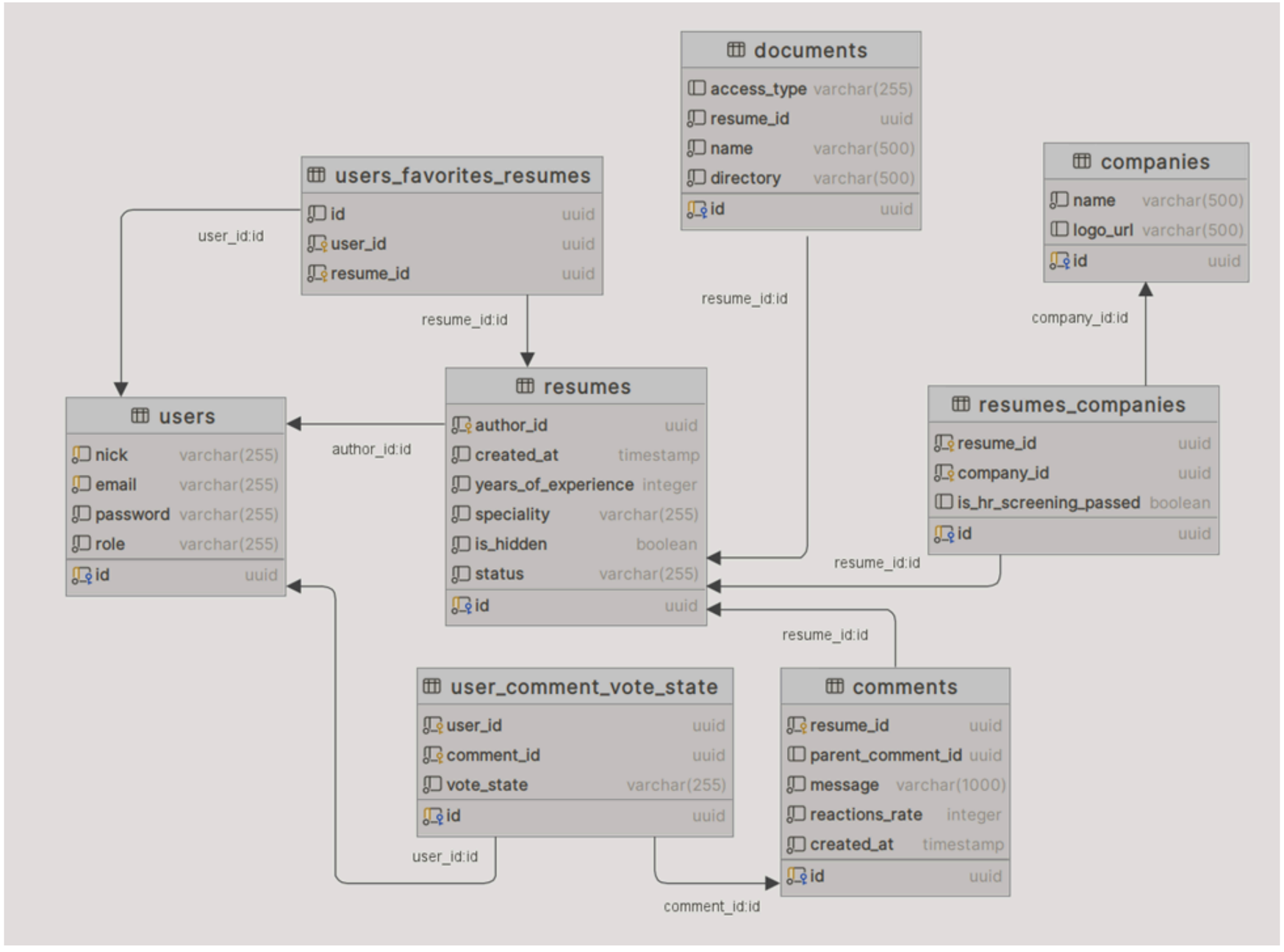
Виконавець:
_____ Григорій АВЧАРОВ

Київ – 2025

КПІ.ІП-1201.045440.06.99.ССВ



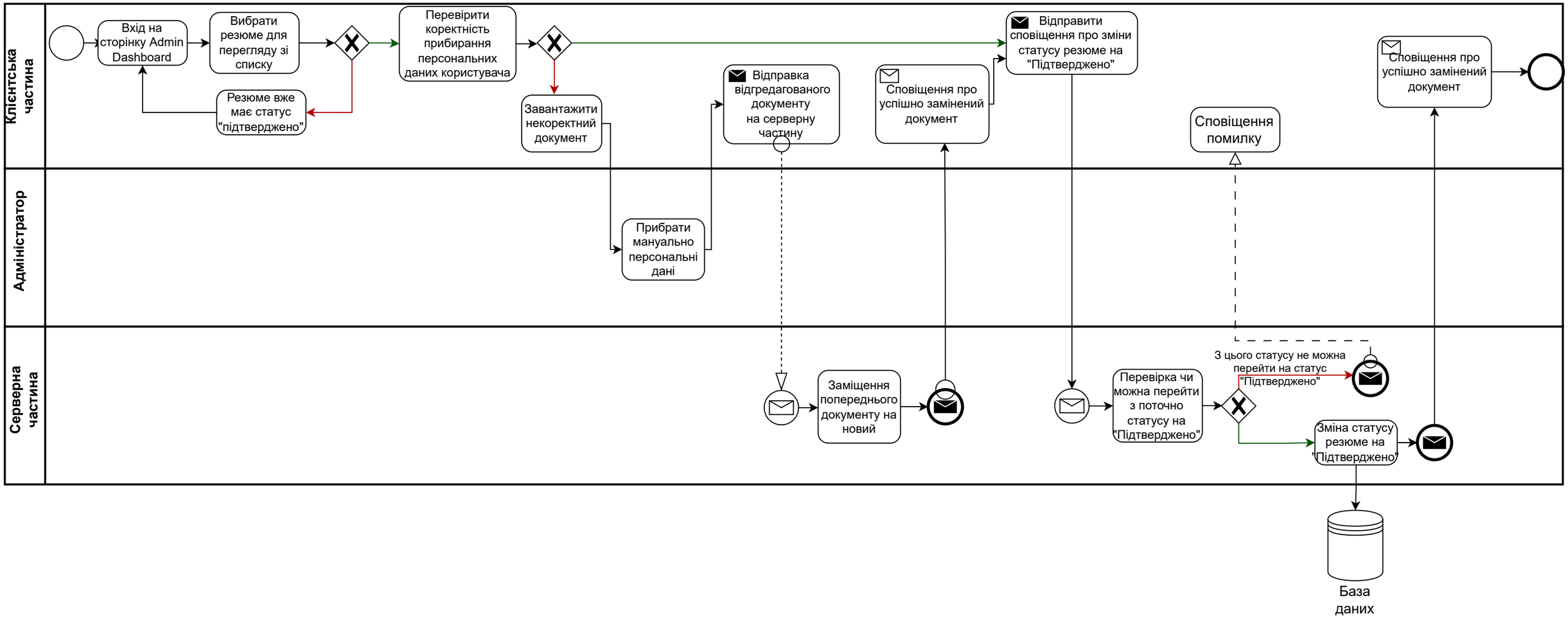
					КПІ.ІП-1201.045440.06.99.ССВ
Зм.	Арк.	№ докум.	Підп.	Дата	
Розробив		Авчаров Г.О.			Діаграма варіантів використань
Перевірив		Фіногенов О.Д.			Лит. / Маса / Масштаб
Т. контр.					Аркуш 1 / Аркушів 1
Н. контр.		Головченко М.М			Вебзастосунок для покращення ефективності резюме
Затвердив		Жаріков Е.В.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-12



					КПІ.ІП-1201.045440.06.99.СБД	
					Схема бази даних	Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Авчаров Г.О.				Аркуш 1 / Аркушів 1
Перевірив		Фіногенов О.Д.				
Т. контр.						
Н. контр.		Головченко М.М			Вебзастосунок для покращення ефективності резюме	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-12
Затвердив		Жаріков Е.В.				

КПІ.ІП-1201.045440.06.99.СБД

ДІАГРАМА БІЗНЕС ПРОЦЕСУ «МОДЕРАЦІЯ РЕЗЮМЕ АДМІНІСТРАТОРОМ»



Демонстраційний плакат до дипломного проекту

Вебзастосунок для покращення ефективності резюме

Виконав студент гр. ІП-12 Авчаров Г.О.

Керівник Фіногенов О.Д.