Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка WEB-платформи для пошуку адвокатів та online консультацій»

Виконала: студентка __3__ курсу
групи ___1ПІ-19мс_____
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Галушко Н.Д.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Ракитянська. Г. Б.
(прізвище та ініціали)
Рецензент:. асистент. каф. КН Кирилащук Т. Г.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – бакалавр
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"25" березня 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Галушко Наталії Дмитрівні

1. Тема роботи – розробка WEB-платформи для пошуку адвокатів та online консультацій.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від "24" березня 2022 року №_66._

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: середовище розробки PHP Storm, мова розробки PHP, операційна система – Mac OS, система управління базою даних MySQL, базові алгоритми для пошуку адвокатів та спілкуванню з клієнтами.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; аналіз даних; розробка інтерфейсу веб-системи; розробка методу пошуку адвокатів; розробка моделі веб-системи; розробка алгоритмів веб-системи; розробка програмного модуля для пошуку адвокатів; розробка веб-системи; тестування веб-системи; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: графічний інтерфейс веб-системи; метод пошуку адвокатів; модель роботи веб-системи; блок-схеми алгоритмів роботи веб-системи; тестування веб-системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г.Б., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання 25 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз веб-систем для пошуку юристів та вибір чітких цілей для поставленої задачі	26.03.2022 – 01.04.2022	Вик.
2	Розробка алгоритмів веб-системи	04.04.2022 – 15.04.2022	Вик.
3	Розробка модуля пошуку адвокатів	18.04.2022 – 22.04.2022	Вик.
4	Розробка модуля для спілкування з клієнтами	25.04.2022 – 02.05.2022	Вик.
5	Розробка веб-системи з вбудованими модулями	03.05.2022 – 23.05.2022	Вик.
5	Тестування веб-системи	24.05.2022 – 30.05.2022	Вик.
6	Оформлення матеріалів до захисту БДР	31.05.2022 – 10.06.2022	Вик.

Студент _____ Галушко Н.Д.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Ракитянська Г.Б.
(підпис) (прізвище та ініціали)

Анотація

Бакалаврська дипломна робота складається з 51 сторінки формату А4, на яких є 25 рисунків, 3 таблиць, список використаних джерел містить 18 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз web-систем для пошуку адвокатів і online консультацій. Сформульовано мету досліджень – удосконалення процесу пошуку юристів шляхом розробки модуля пошуку, який містить розширений фільтр, що враховує усі важливі для користувача критерії.

Запропоновано метод пошуку адвокатів, який дозволяє врахувати необхідні параметри. Також запропоновано метод для проведення online консультацій, який забезпечує швидкий обмін повідомленнями між користувачами Розроблено модель веб системи, яка містить вбудовані модулі для пошуку адвокатів та модуль для проведення онлайн консультацій. Розроблено зручний графічний інтерфейс для веб-середовища. Налагоджено взаємодію між модулями пошуку адвокатів, модулем для проведення онлайн консультацій та графічним інтерфейсом.

Розроблено алгоритм роботи веб-системи на основі розроблених методів та моделей для пошуку адвокатів та онлайн консультацій.

Отримані в бакалаврській роботі результати можуть використовуватись приватними адвокатами та юридичними фірмами.

Ключові слова: веб-платформа, пошук адвокатів, юридичні онлайн консультації.

Abstract

The bachelor's thesis consists of 51 A4 pages, which contain 25 figures, 3 tables, the list of sources used contains 18 titles.

In the bachelor's thesis a detailed analysis of web-systems for finding lawyers and online consultations was conducted. The purpose of the research is formulated - to improve the process of finding lawyers by developing a search module that contains an advanced filter that takes into account all the important criteria for the user.

A method of finding lawyers has been proposed, which allows to take into account the necessary parameters. A method for conducting online consultations is also proposed, which provides fast exchange of messages between users. A model of a web system has been developed, which contains built-in modules for finding lawyers and a module for conducting online consultations. Developed a user-friendly graphical interface for the web environment. The interaction between the modules for finding lawyers, the module for online consultations and the graphical interface has been established.

An algorithm for the operation of the web system based on the developed methods and models for finding lawyers and online consultations has been developed.

The results obtained in the bachelor's thesis can be used by private lawyers and law firms.

Keywords: web system, search for lawyers, online legal advice.

ЗМІСТ

ВСТУП

Чи можливо зараз уявити сферу бізнесу, якої б не торкнулись IT-технології? Відповідь, швидше за все, негативна. Автоматизація, штучний інтелект вже впроваджені у більшість галузей і продовжують стрімко розвиватись.

Технологічні тренди повпливали і на таку консервативну сферу як юриспруденція. Більшість працівників сфери є свідками переходу від традиційної моделі бізнесу (traditional low) до нової моделі (new law) [1]. Рано чи пізно така трансформація торкнеться кожного представника юридичного бізнесу, в іншому випадку представники сфери не будуть здатні витримати конкуренцію на ринку.

Сучасні веб-системи для пошуку юристів здебільшого існують у вигляді каталогів з досьє юристів. Їх основним недоліком є відсутність можливості фільтрації по важливим професійним характеристикам, наприклад, таких як досвід в апеляційних інстанціях, досвід у касаційній інстанції, досвід у міжнародних установах, наявність рекомендацій. Відповідно користувачам таких веб-систем необхідно витратити багато часу на перегляд досьє та пошук необхідного спеціаліста, та навіть у разі перегляду багатьох профілів є вірогідність не знайти необхідної інформації про досвід юриста. Також великим недоліком існуючих веб-ресурсів є відсутність можливості проведення online консультацій Для розв'язання даної проблеми доцільною є розробка внутрішнього модуля для пошуку юриста, який містить розширений фільтр, а також розробка модуля для зручної online комунікації спеціалістів з клієнтами . Тому актуальною є розробка веб-системи для пошуку юристів з модулями пошуку юристів, що містить розширений фільтр, та модулем для online консультацій.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є удосконалення процесу пошуку юристів шляхом розробки модуля пошуку, який містить розширений фільтр, що враховує усі важливі для користувача критерії. Основними задачами роботи є:

- розробити метод для пошуку адвокатів;
- розробити метод для онлайн консультацій;
- розробити модель веб-системи;
- розробити базу даних відповідно до вимог веб-системи;
- розробити модуль пошуку адвокатів;
- розробити модуль для налагодження процесу онлайн консультацій;
- розробити графічний інтерфейс для веб-середовища;
- налаштувати взаємодію між модулями пошуку адвокатів, online консультацій та графічним інтерфейсом;
- провести тестування модуля пошуку адвокатів з використанням графічного інтерфейсу.

Об’єкт дослідження – процеси розробки програмних засобів для web-платформ, призначених для пошуку адвокатів та online консультацій.

Предмет дослідження – методи та засоби розробки програмних засобів web-платформи для пошуку адвокатів з модулем пошуку адвокатів, який містить розширений фільтр.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

- методи побудови веб-систем для побудови стабільної веб-системи;
- методи передачі репрезентативного стану для передачі даних між клієнтом та сервером;
- методи автоматизації процесу пошуку адвокатів;
- методи зберігання персональних даних для захисту даних користувачів веб-системи.

Наукова новизна отриманих результатів.

- Подальшого розвитку отримав метод автоматизації процесу пошуку адвокатів, який, на відміну від існуючих, дозволяє відфільтрувати досьє спеціалістів з урахуванням багатьох професійних критеріїв.
- Подальшого розвитку отримав метод для автоматизації процесу спілкування юристів з клієнтами, який на відміну від існуючих автоматично надсилає листи та реалізує можливість проведення консультацій в режим online.

Практична цінність отриманих результатів. Практична цінність полягає у кінцевій реалізації веб-системи з інтегрованою системою пошуку адвокатів, що дозволяє врахувати усі необхідні критерії спеціаліста.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій роботі автору належать такі результати: модуль для пошуку адвокатів; модуль налагодження online консультацій.

Публікації. Основні результати дослідження опубліковані в науковій роботі – в тезах доповіді на науково-технічній конференції підрозділів Вінницького національного технічного університету факультету інформаційних технологій та комп’ютерної інженерії (2022).

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 18 літературних джерел.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану веб-систем для пошуку юристів

З розвитком інформаційних технологій пошук юристів шляхом відвідування багатьох юридичних компаній стає недоцільним у зв’язку з часом, який затрачається на пошуки. Люди, яким потрібен юридичний спеціаліст, переходять до використання різних веб-ресурсів, які дають можливість швидкого необхідного спеціаліста. Подібний підхід надає кілька значних переваг:

– можна переглянути досьє багатьох спеціалістів;
– можна поспілкуватись зі спеціалістами онлайн;
– не потрібно відвідувати юридичні фірми.

Цифрові технології стрімко розвиваються і захоплюють все більше сфер, зокрема і сферу юриспруденції. Керуючись потребою ринку окремі компанії впроваджують власні рішення для автоматизації процесів їхньої сфери діяльності. До основних переваг даних систем можна віднести:

– впровадження власного графічного інтерфейсу;
– відсутність пропозицій конкурентів;
– можливість розширення функціоналу для пошуку спеціалістів.

Незважаючи на всі переваги власних систем для автоматизації процесів у сфері юриспруденції, більшість компаній віддають перевагу загальнодоступним веб-системам. Основною причиною даного рішення є вартість розробки власної системи. Однак, загальнодоступні системи для пошуку спеціалістів також мають свої переваги:

– простота розміщення досьє спеціалістами;
– можливість просування та збільшення відвідуваності шляхом SEO оптимізації;
– можливість надавати послуги як індивідуальний спеціаліст, а не працівник юридичної фірми.

Виходячи з даного порівняння зрозуміло, що більш доцільним рішенням є використання відкритих систем, які будуть постійно підтримуватися провайдером та дозволять привабити більшу кількість клієнтів.

1.2 Порівняльний аналіз аналогів

Існує досить невелике різноманіття веб-систем для пошуку спеціалістів у сфері юриспруденції. До найбільш відомих рішень відносять:

– Legarithm;
– Legist;
– Protocol.ua;
– Lawyer.ua.

Legarithm (рис. 1.1) – веб-система, яка дозволяє замовити юридичні послуги та замовити консультацію по питанням, що стосуються сфери юриспруденції [2].

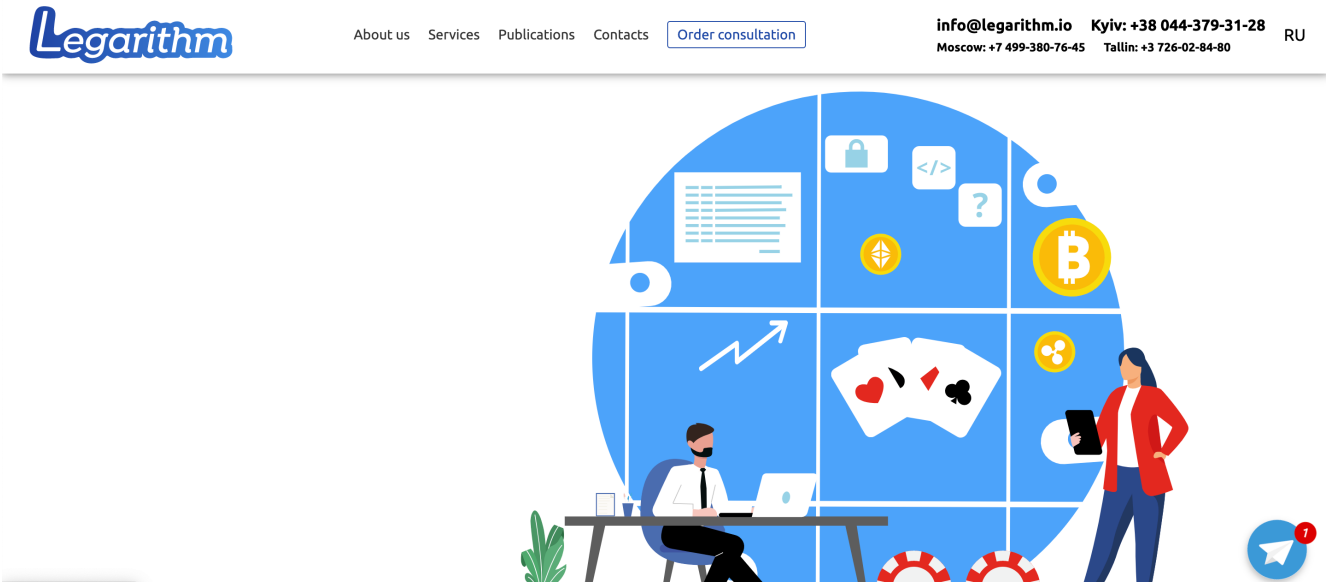

Рисунок 1.1 – Головна сторінка сайту Legarithm

Legist (рис. 1.2) –веб-система компанії, яка надає юридичні послуги фізичним та юридичним особам. Дана система націлена на забезпечення швидкого надання послуг online консультацій [3].

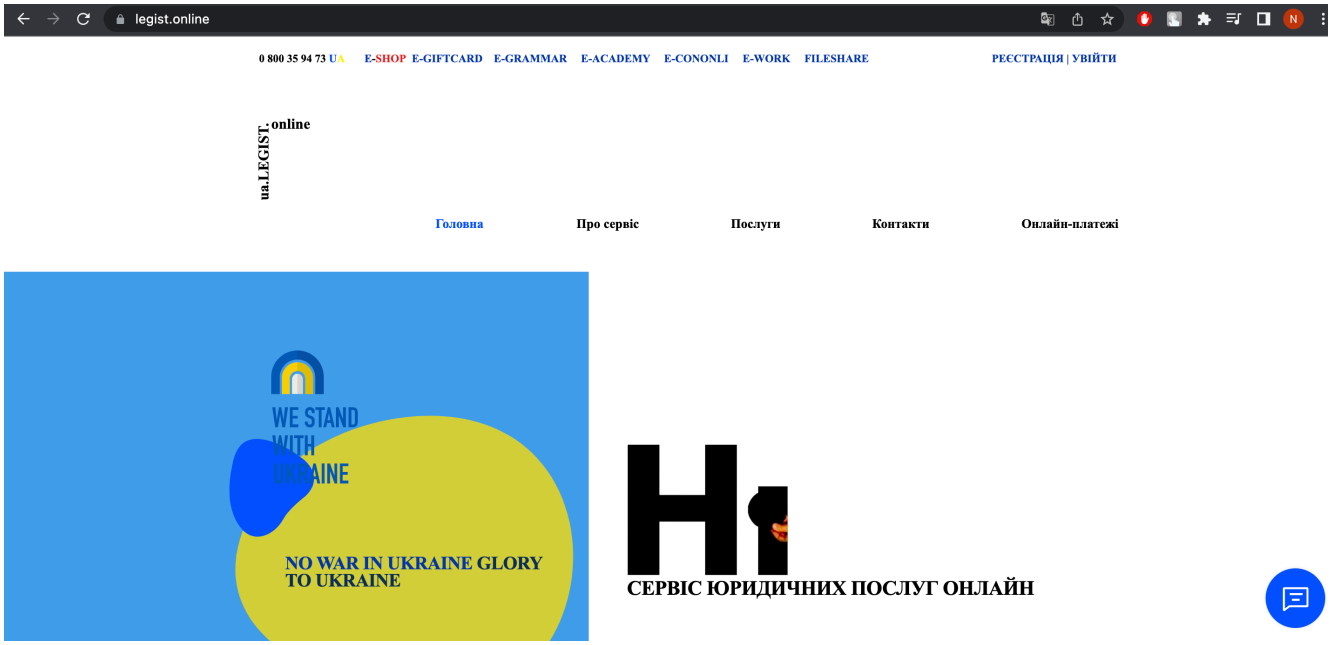

Рисунок 1.2 – Головна сторінка сайту Legist

Protocol.ua (рис. 1.3) – веб-система для пошуку юристів та онлайн консультацій. Дана система надає всі стандартні функції для пошуку спеціаліста та систему замовлення консультації. Особливістю ресурсу є можливість провести відеодзвінок з юристом [4].

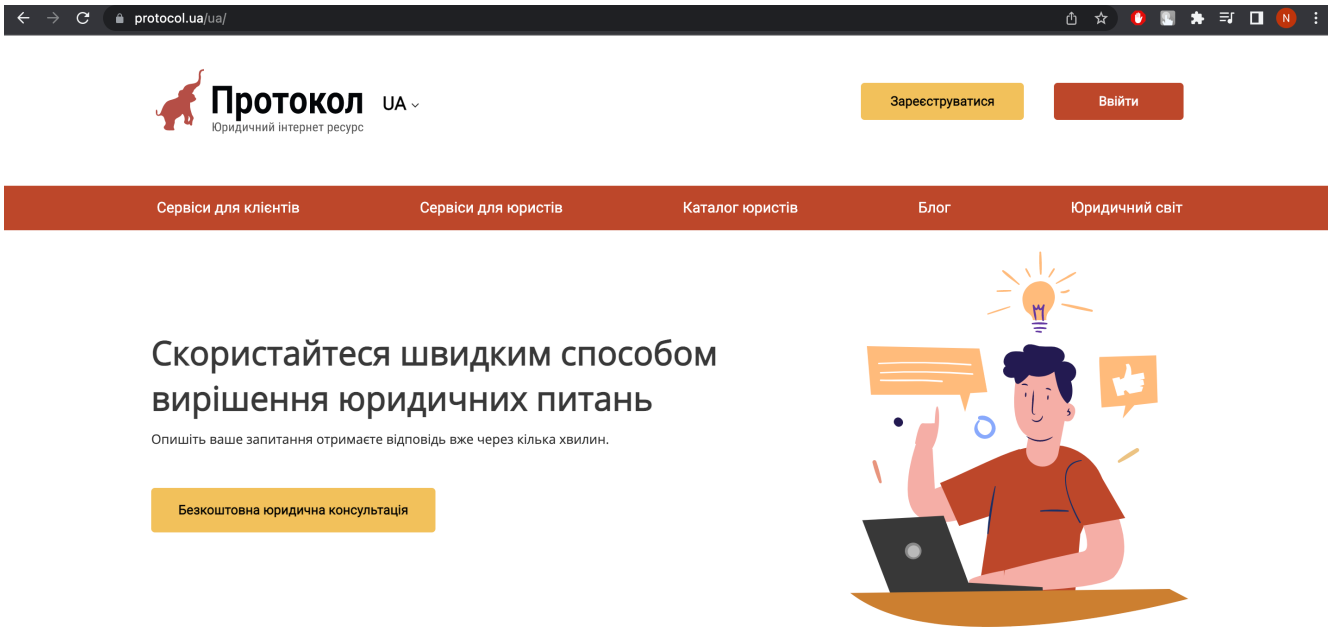

Рисунок 1.3 – Головна сторінка сайту Protocol.ua

Lawyer.ua (рис. 1.4) – найвідоміша веб-система для пошуку юристів. Дана система передбачає зручний інтерфейс пошуку спеціалістів та замовлення юридичних послуг. Існує можливість відфільтрувати досьє спеціалістів по основним критеріям [5].

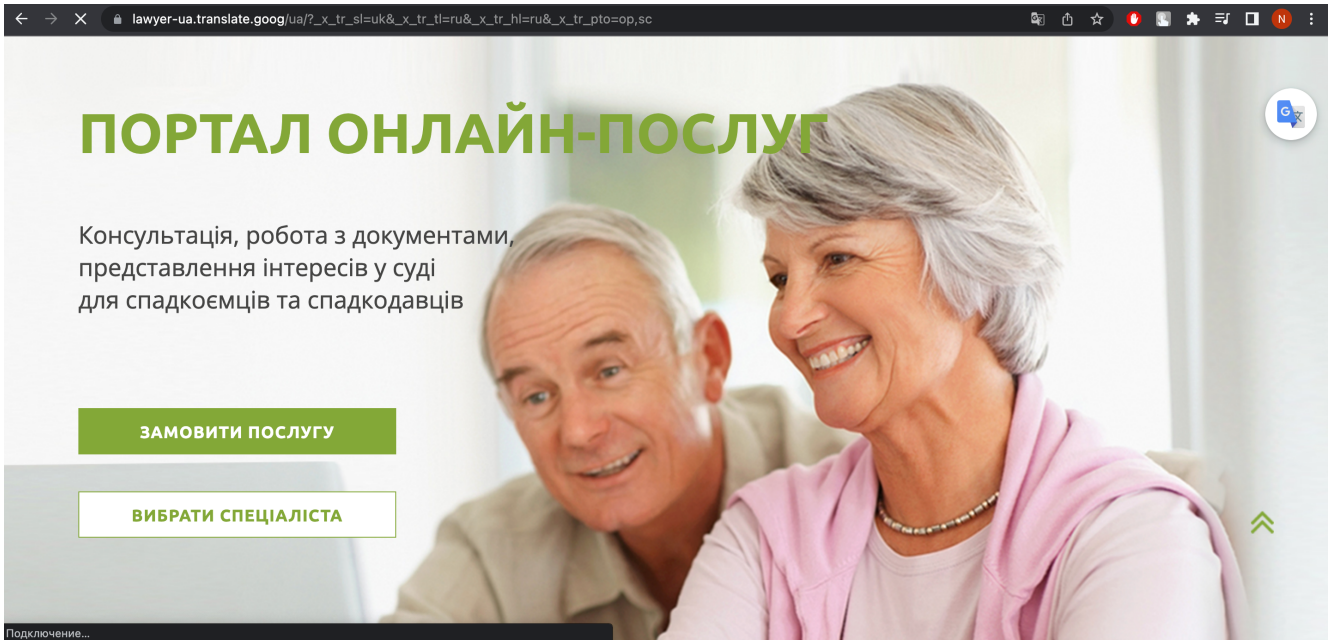

Рисунок 1.4 – Головна сторінка сайту Lawyer.ua

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика веб-систем

Критерії	Legarithm	Legist	Protocol.ua	Lawyer.ua	Власний модуль
Адаптивний інтерфейс	+	+	+	+	+
Можливість розміщувати досьє для усіх бажаючих спеціалістів	-	-	+	+	+
Наявність модуля пошуку юристів	-	-	+	+	+

Наявність розширених фільтрів для пошуку юристів	-	-	-	-	+
Наявність можливості проведення консультацій online	–	+	+	+	+
Загальна оцінка	20%	40%	80%	80%	100%

Відповідно до таблиці порівняльних характеристик розробка власної веб-системи для пошуку адвокатів є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити новий функціонал для автоматизації процесів сфери юриспруденції.

1.3 Аналіз методів розв’язання поставленої задачі

Для розв’язання поставленої задачі існує декілька підходів.

Один із них це створення окремих модулів для пошуку юристів з розширеним фільтром та online консультацій, які будуть інтегруватись у вже існуючі веб-системи. Такий варіант реалізується найшвидше, проте містить певні недоліки: додаткові зусилля зі сторони розробників вже існуючих веб-ресурсів, можлива несумісність у версіях технологій, що застосовані при розробці модулів та існуючих веб-ресурсів.

Кращою альтернативою є розробка власного сервісу для інтеграції модулів для пошуку юристів з розширеним фільтром та online консультацій . Даний підхід дає можливість зосередитись на розробці саме необхідного модуля без повномасштабної розробки веб-системи. Користувачі даного сервісу матимуть можливість інтегрувати повноцінне рішення у свої системи. Це значно зменшить ресурси необхідні для інтеграції у порівнянні з попереднім рішенням. Також дане рішення передбачає розробку внутрішньої системи авторизації для захисту даних веб-систем, які забажають інтегрувати даний модуль. Однак, подібний сервіс може потребувати значних апаратних ресурсів у випадку інтеграції багатьма сервісами.

Найбільш поширеним рішенням є розробка власної веб-системи з вбудованими модулями пошуку юристів з розширеним фільтром та online консультацій. У даному випадку вирішується проблема з конфліктом у версіях технологій з тими, що застосовані в існуючих веб-ресурсах. Також результатом даного методу розробки буде унікальна веб-система, яка буде містити унікальний функціонал. Даний підхід теж має свій недолік оскільки потребує повноцінної розробки системи.

Врахувавши переваги та недоліки кожного методу було вирішено використати метод розробки власної веб-системи із влаштованим модулями для пошуку юристів, що містить розширений фільтр, та для online консультацій адже він дозволить надати повноцінний програмний продукт без необхідності в потужному апаратному забезпеченні.

1.4 Постановка задач для веб-системи для пошуку юристів

Проаналізувавши переваги та недоліки існуючих веб-систем для пошуку юристів, було сформульовано такі задачі бакалаврської дипломної роботи:

- розробити метод автоматизації процесу пошуку юристів, який дозволить відфільтрувати досьє спеціалістів по професійним критеріям, щоб відсіяти спеціалістів, які не підходять користувачу;
- визначити найбільш ефективний підхід для створення досьє юристами;
- розробити зручний та адаптивний графічний інтерфейс веб-систем;
- розробити веб-систему для пошуку юристів з влаштованим модулем пошуку спеціалістів, який містить розширений фільтр;
- розробити веб-систему для пошуку юристів з влаштованим модулем для проведення online консультацій;
- провести тестування веб-системи згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан веб-систем для пошуку юристів на сьогоднішній день. Було проведено порівняння відомих веб-систем для пошуку юристів та проведення online консультацій, таких як: Legarithm, Legist, Protocol.ua та Lawyer.ua. У результаті порівняння було виявлено, що досліджувані системи набули значної популярності серед учасників ринку праці завдяки своїй зручності та доступності. Проаналізувавши переваги та недоліки відомих веб-систем, було виявлено, що вони не надають можливості в достатній мірі автоматизувати процес пошуку спеціалістів сфери юриспруденції та проведення консультацій в режимі онлайн. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власної веб-системи.

2 РОЗРОБКА МЕТОДУ ПОШУКУ АДВОКАТІВ, МОДЕЛІ РОБОТИ ТА АЛГОРИТМІВ ВЕБ-СИСТЕМИ

2.1 Аналіз даних

Одним з найважливіших аспектів веб-системи є дані, які надають інформаційної цінності та дозволяють поширити ідеї серед користувачів. Основним постачальником даних у веб-системах для автоматизації процесів сфери юриспруденції виступають спеціалісти, які створюють та розміщують досьє. У бакалаврській дипломній роботі працівники юридичної сфери також виступають постачальниками даних для модулю пошуку спеціалістів. Таким чином, можна зрозуміти, що основна частина інформації формується на даних, які отримуються від спеціалістів юридичної сфери.

У бакалаврській дипломній роботі клієнтська частина отримує дані з серверної за допомогою API веб-системи. Клієнтська частина вказує юристам, які дані та в якому форматі вони повинні надати. На серверній частині вони валідуються, обробляються та зберігаються в базі даних в потрібному форматі. Отримання даних, що були збережені раніше, відбувається шляхом надсилання запитів на сервер. Базуючись на параметрах, які передаються з клієнтської частини, функції на серверній частині відшукують необхідні дані, обробляють їх та повертають у модулі зі сторони клієнта.

Так як веб-система розрахована на велику кількість спеціалістів юридичною сфери та велику кількість користувачів одним з найважливіших аспектів є швидкість отримання даних з бази даних та їх обробка. При великій кількості досьє юристів пошук в базі даних може тривати кілька хвилин часу, що також дуже погано впливатиме на індексацію ресурсу пошуковими системами. Для уникнення такої ситуації було прийнято рішення використати кешування даних. Кеш даних – це високошвидкісний рівень зберігання, на якому необхідний набір даних має тимчасовий характер. За допомогою кешування стає можливим ефективне

повторне використання раніше отриманих або обчислених даних [6]. Таким чином користувач зможе отримати необхідні дані максимально швидко.

2.2 Розробка структури інтерфейсу веб-системи

Розробляючи інтерфейс веб-системи було вирішено зробити його максимально простим для використання та інтуїтивно зрозумілим.

Для зручності навігації було добавлено головне меню (рис. 2.1), де користувач може обрати сторінку для відображення. Для швидкого пошуку адвокатів по географічному регіону було розроблено додаткове контекстне меню (рис. 2.2).

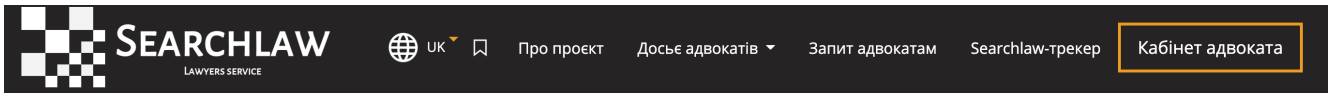

Рисунок 2.1 – Головне меню

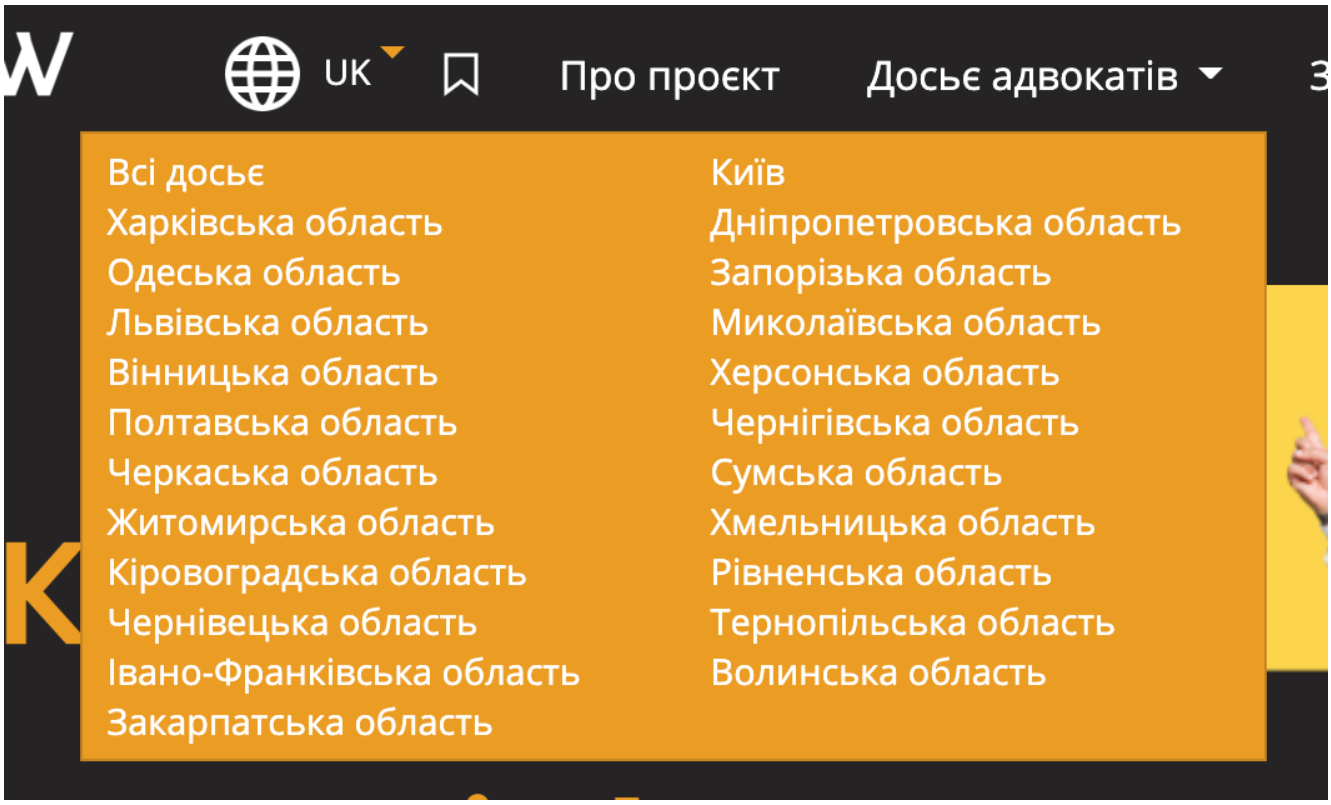

Рисунок 2.2 – Контекстне меню

Для зручного пошуку юристів було вирішено розмістити фільтри на одній сторінці зі списком досьє юристів (рис. 2.3). Таким чином користувачу не доведеться прикладати значних зусиль для пошуку бажаної інформації.

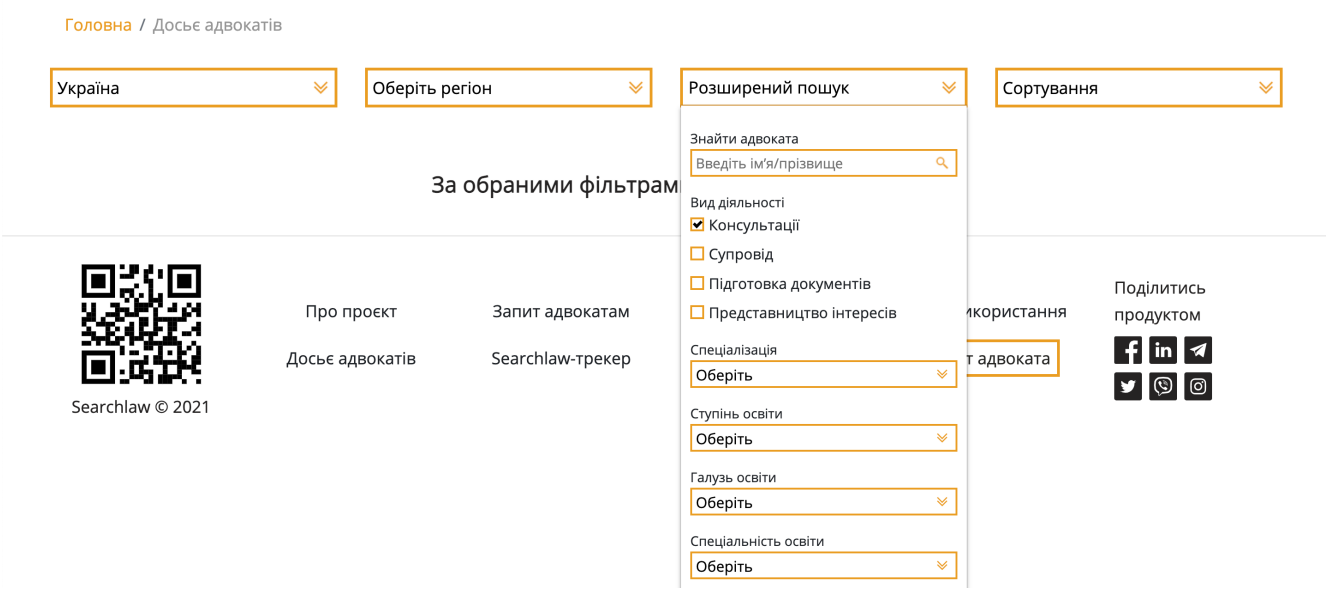

Рисунок 2.3 – Сторінка досьє

Для зручного надсилання запиту адвокату була розроблена форма, яка розміщена на окремій сторінці (рис. 2.4).

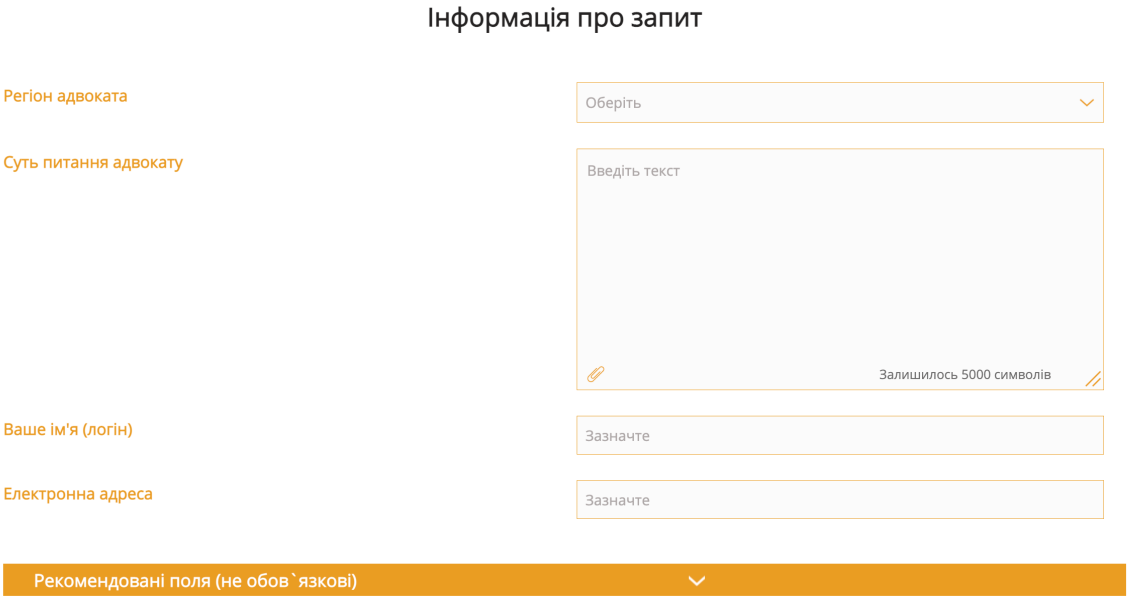

Рисунок 2.4 – Сторінка для надсилання запиту адвокату

Також був розроблений зручний інтерфейс для створення досьє адвоката

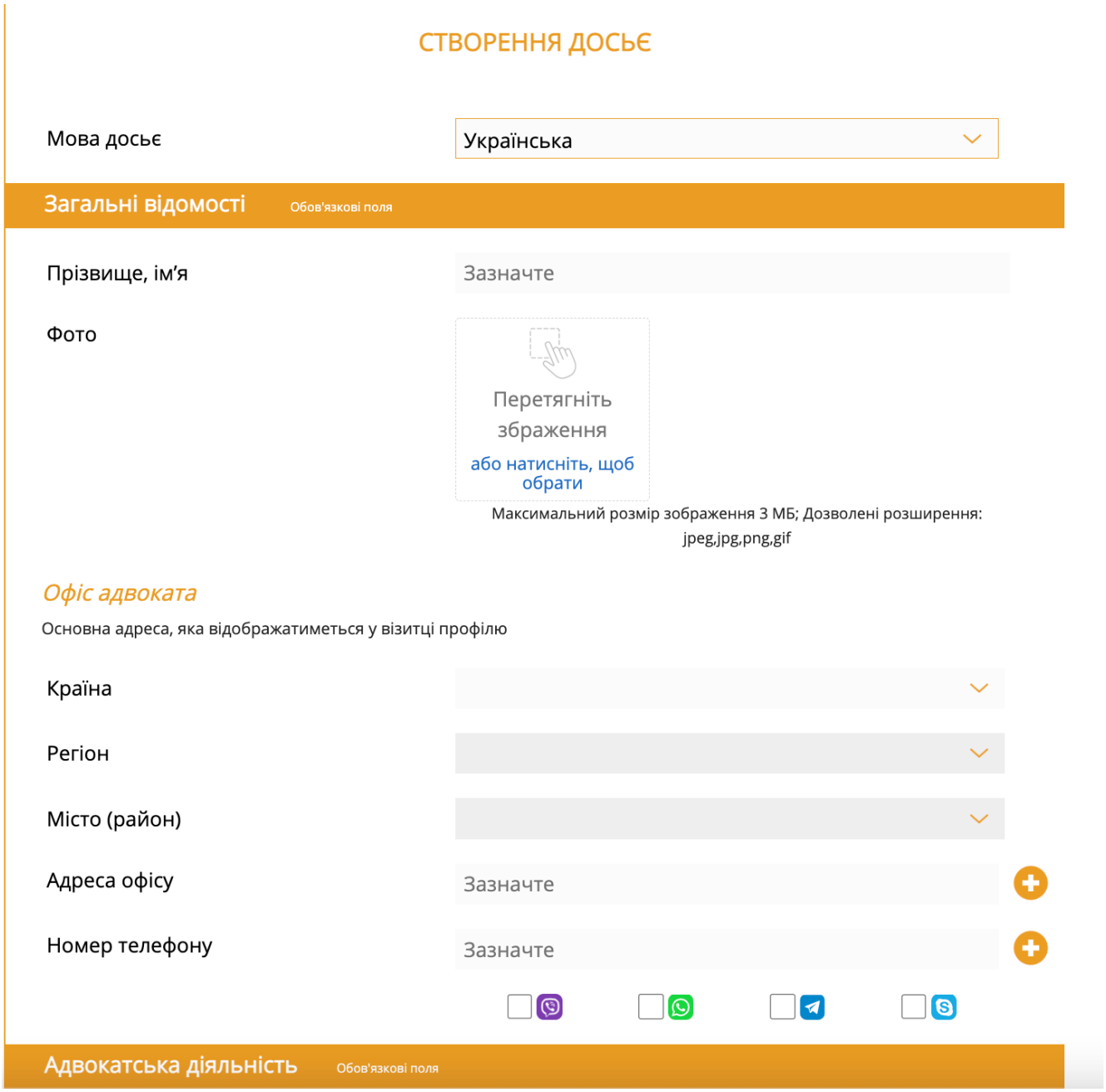

Рисунок 2.5 – Сторінка досьє

Розробка графічного інтерфейсу веб-системи була проведена у середовищі Figma.

2.3 Розробка методу пошуку адвокатів

Для автоматизації процесу пошуку адвокатів було вирішено розробити метод пошуку адвокатів (рис. 2.6). Даний метод передбачає, що адвокат створить досьє, а клієнт знайде його у разі відповідності критеріям та зможе почати комунікацію зі спеціалістом.

Рисунок 2.6 – Метод пошуку досьє

2.4 Розробка моделі роботи веб-системи

Для кращого розуміння роботи веб-системи для пошуку адвокатів було вирішено розробити модель роботи системи з використанням UML діаграми діяльності (рис. 2.7). Дані діаграми найкращим чином дозволяють описати роботу програми [7]. Згідно вказаної діаграми першим ділом юрист повинен обрати регіон в якому веде свою діяльність. Наступним кроком є створення досьє, яке в подальшому відправляється на перевірку адміністратору. Якщо досьє проходить перевірку, то адвокат отримує повідомлення про його активізацію. Після цього якщо користувач шукатиме досьє, він побачить створене досьє в результатах

пошуку, у разі відповідності регіону або іншим критеріям. Далі користувач матиме можливість переглянути досьє або ж почати комунікацію з адвокатом.

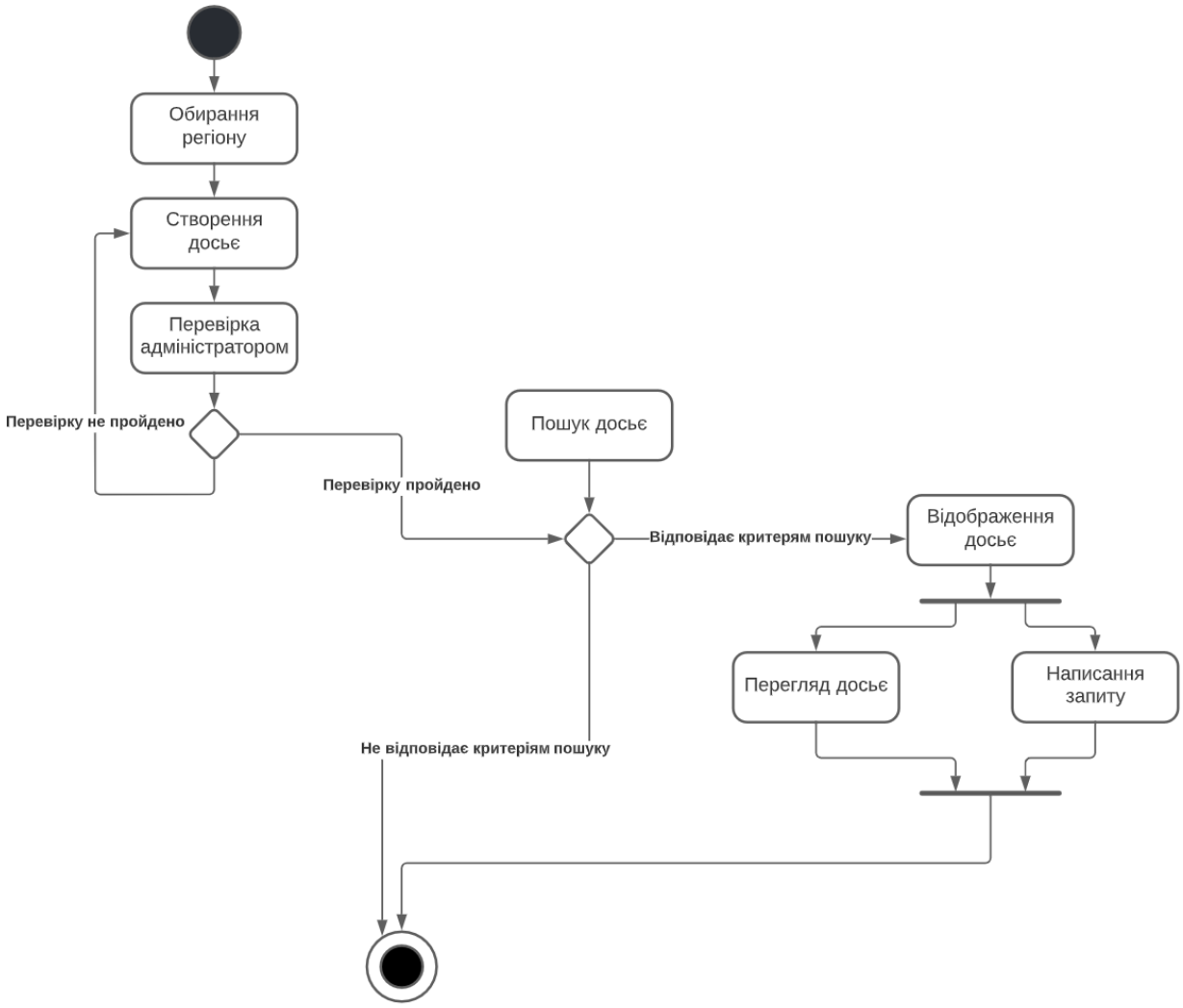

Рисунок 2.7 – Діаграма діяльності веб-системи

2.5 Розробка алгоритмів роботи веб-системи

Для розробки веб-системи для пошуку юристів необхідно розробити загальний алгоритм, згідно якого буде працювати система. Згідно даного алгоритму першим ділом юрист повинен обрати регіон діяльності із списку запропонованих та створити досьє. Далі досьє надсилається на перевірку адміністратору системи. У разі проходженні перевірки досьє публікується на сайті та стає доступним для пошуку. Якщо користувач шукатиме адвоката по критеріям,

яким відповідає створене юристом досьє – він побачить його в результатах пошуку. Користувач матиме можливість переглянути досьє або написати заявку даному спеціалісту та почати з ним комунікацію. При надсиланні заявки адвокату автоматично надсилається завчасно сформований адміністратором лист про отримання нового запиту. Запит адвокат може побачити в особистому кабінеті. Якщо адвокат приймає запит – він продовжує комунікацію з користувачем в режимі online. Повідомлення з веб-системи дублюються на пошту. У разі відхилення запиту користувач автоматично отримує відповідне повідомлення на електронну пошту. Таким чином працює загальний алгоритм веб-ресурсу в основі якого алгоритми для автоматизації пошуку адвокатів та процесу комунікації.

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних веб-системи. Також було розглянуто можливість покращення продуктивності веб-системи шляхом кешування даних, що отримуються з бази даних. Було розроблено основний алгоритм роботи веб-системи в основі якого алгоритми пошуку адвоката за потребами користувача та комунікація юристів з користувачами. Також було розроблено метод пошуку юристів, який надає змогу реалізувати процес автоматизованого пошуку спеціалістів з врахуванням критеріїв користувача. Для кращого розбору роботи веб-системи було розроблено модель системи за допомогою UML діаграми діяльності.

3 РОЗРОБКА ВЕБ-СИСТЕМИ ДЛЯ ПОШУКУ АДВОКАТІВ З ВЛАШТОВАНИМ МОДУЛЕМ ПОШУКУ АДВОКАТІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Одною з найважливіших частин розробки програмного забезпечення є вибір правильних технологій. Даний вибір може значною мірою вплинути на час роботи та якість вихідного результату. Вибір технологій повинен залежати від загального призначення вихідного продукту та можливості його майбутньої підтримки, оскільки існує велика імовірність, що з часом доведеться оновлювати версії використаних технологій.

PHP - це популярна мова програмування, особливо серед веб-розробників. Початкова ідея PHP полягала у розробці набору інструментів спрощення процесу створення динамічних веб-сторінок. Незважаючи на те, що сучасний PHP є мовою загального призначення, частіше його використовують як серверний інструмент для генерації HTML-коду, який потім інтерпретується веб-браузером [8].

Сьогодні виділяють три основні області використання PHP:

1. Для написання скриптів та повноцінних веб-застосунків, що виконуються на серверній стороні. Це найпопулярніша сфера застосування, оскільки мова спочатку створювалася саме для веб-розробок. Для повноцінної роботи веб-програми, написаної на PHP, необхідні сервер, парсер (CGI-додаток) та клієнтське програмне забезпечення (веб-браузер), яке відображає результат виконання коду.

2. Для створення сценаріїв, які виконуються в командному рядку. Такі міні-програми можуть працювати на будь-якому ПК. Для їх виконання потрібно лише парсер. Оскільки PHP містить потужні інструменти для роботи з рядками, такі сценарії найчастіше створюють для обробки текстових даних.

3. Для написання графічних інтерфейсів. PHP має багато відгалужень, створених для реалізації різних завдань. Одним із таких відгалужень є PHP-GTK. Його зазвичай використовують програмісти, які звикли до синтаксису PHP [9].

Популярність PHP обумовлена такими перевагами:

1) Простий та інтуїтивно зрозумілий синтаксис. PHP швидко освоюють навіть програмісти-новачки. Він увібрав усі найкращі особливості таких популярних мов, як C, Java та Perl. PHP-код легко читається незалежно від способу використання (для написання невеликих скриптів або створення потужних додатків з використанням об'єктно-орієнтованого підходу до реалізації програми).
2) Кросплатформенність та гнучкість. PHP сумісний із усіма популярними платформами (Linux, Windows, MacOS). Написані на ньому програми успішно працюють на різному серверному ПЗ (IIS, Nginx, Apache та багатьох інших).
3) Відмінна масштабованість. PHP дозволяє досягти максимальної продуктивності додатків, написаних на ньому, зі зростанням апаратних ресурсів. Веб-програми, розподілені на кілька серверів, здатні справлятися із суттєвими навантаженнями (великим трафіком).
4) Активний розвиток та вдосконалення. Спільнота розробників постійно працює над впровадженням додаткового функціоналу, що розширює можливості мови, спрощенням синтаксису та покращенням захисту від можливих атак.
5) Широкі перспективи подальшого розвитку. Більшість CMS були створені на чистому PHP та фреймворках. Цим обумовлені популярність та затребуваність PHP програмістів.

Laravel – один з найпопулярніших PHP-фреймворків для написання веб-додатків, створений на основі Symfony, як альтернатива CodeIgniter – фреймворку, що використовує архітектурну модель Model View Controller (MVC) з відкритим вихідним кодом. Laravel є найсильнішим суперником в екосистемі PHP просто тому, що він включає функції, необхідні для створення сучасних, підтримуваних,

розподілених веб-додатків в реальному часі. Крім того, у нього є велика відеотека Laracasts, що містить понад 900 посібників [10].

Переваги PHP фреймворку Laravel:

1. Структура Model - View – Controller, яка дозволяє ізолювати один від одного компоненти для виконання різних завдань [11].
2. У Laravel використовується Eloquent ORM, яка спрощує роботу з базою даних [12].
3. У Laravel використовується легковажний і високопродуктивний (завдяки кешування) шаблонизатор Blade, за допомогою якого можна легко стандартизувати та використовувати надалі шаблон [13].
4. Laravel надає вбудовані бібліотеки та модулі, які допомагають покращити вашу веб-програму. Кожен модуль інтегрований із менеджером залежностей Composer, що спрощує оновлення [14].
5. Спрощує розгортання та оновлення веб-програми, позбавляючи вас від помилок та конфліктів, особливо якщо над проектом працює команда розробників.

Для фронт-енд частини використано фреймворк Vue.js.

Основними перевагами фреймворку Vue.js є швидкість рендерингу, реактивність, HTML-шаблони, наявність великої кількості документації та активна спільнота розробників.

Для роботи з базою даних було використано MySQL.

Крім універсальності та поширеності СУБД MySQL має цілий комплекс важливих переваг перед іншими системами. Зокрема, слід зазначити такі якості як:

1. Простота у використанні. MySQL досить легко встановлюється, а наявність безлічі плагінів та допоміжних програм спрощує роботу з базами даних.
2. Великий функціонал. Система MySQL має практично весь необхідний інструментарій, який може знадобитися в реалізації практично будь-якого проекту.

3. Безпека. Система спочатку створена в такий спосіб, що багато вбудованих функцій безпеки у ній працюють за умовчанням.
4. Масштабованість. Будучи вельми універсальною СУБД, MySQL однаково легко можна використовувати для роботи і з малими, і з великими обсягами даних.
5. Швидкість. Висока продуктивність системи забезпечується за рахунок спрощення деяких стандартів, що використовуються в ній [15].

Враховуючи всі описані вище переваги для розробки веб-системи для пошуку адвокатів та онлайн консультацій було обрано такі інструменти: мова програмування PHP, фреймворк Laravel, Vue.js та СУБД MySQL.

3.2 Розробка бази даних

Перед розробкою веб-системи необхідно спроєктувати та створити базу даних. Для створення бази даних був використаний механізм міграцій, який пристутній в фреймворку Laravel.

Міграції - це щось подібне до системи контролю версій для бази даних. Вони дозволяють команді програмістів змінювати структуру БД, водночас залишаючись у курсі змін інших учасників [16]. Міграції зазвичай йдуть пліч-о-пліч з конструктором таблиць для більш простого поводження з архітектурою вашого додатку.

Приклад коду створення таблиці шляхом використання міграцій наведено нижче:

```
class CreateDossiersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('dossiers', function (Blueprint $table) {
            $table->bigIncrements('id');
```

```
            $table->bigInteger('user_id')->unsigned()->nullable();
            $table->string('locale');
            $table->string('number')->nullable();
            $table->tinyInteger('status')->default(0);
            $table->bigInteger('region_id')->unsigned();
            $table->bigInteger('city_id')->unsigned()->nullable();
            $table->json('license');
            $table->json('overview')->nullable();
            $table->json('contacts')->nullable();
            $table->json('education')->nullable();
            $table->json('experience_other')->nullable();
            $table->json('training')->nullable();
            $table->json('publications')->nullable();
            $table->json('recommendations')->nullable();
            $table->json('honors')->nullable();
            $table->json('services')->nullable();
            $table->string('qr')->nullable();
            $table->timestamp('admin_seen_at')->nullable();
            $table->bigInteger('seen_counter')->unsigned()->default(0);
            $table->bigInteger('bookmarks_counter')->unsigned()-
>default(0);
            $table->timestamps();
            $table->foreign('user_id')
                ->references('id')->on('users')
                ->onDelete('cascade');
            $table->foreign('region_id')
                ->references('id')->on('regions')
                ->onDelete('cascade');
            $table->foreign('city_id')
                ->references('id')->on('cities')
                ->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('resumes');
    }
}
```

Діаграма бази даних зображена на рисунку 3.1.

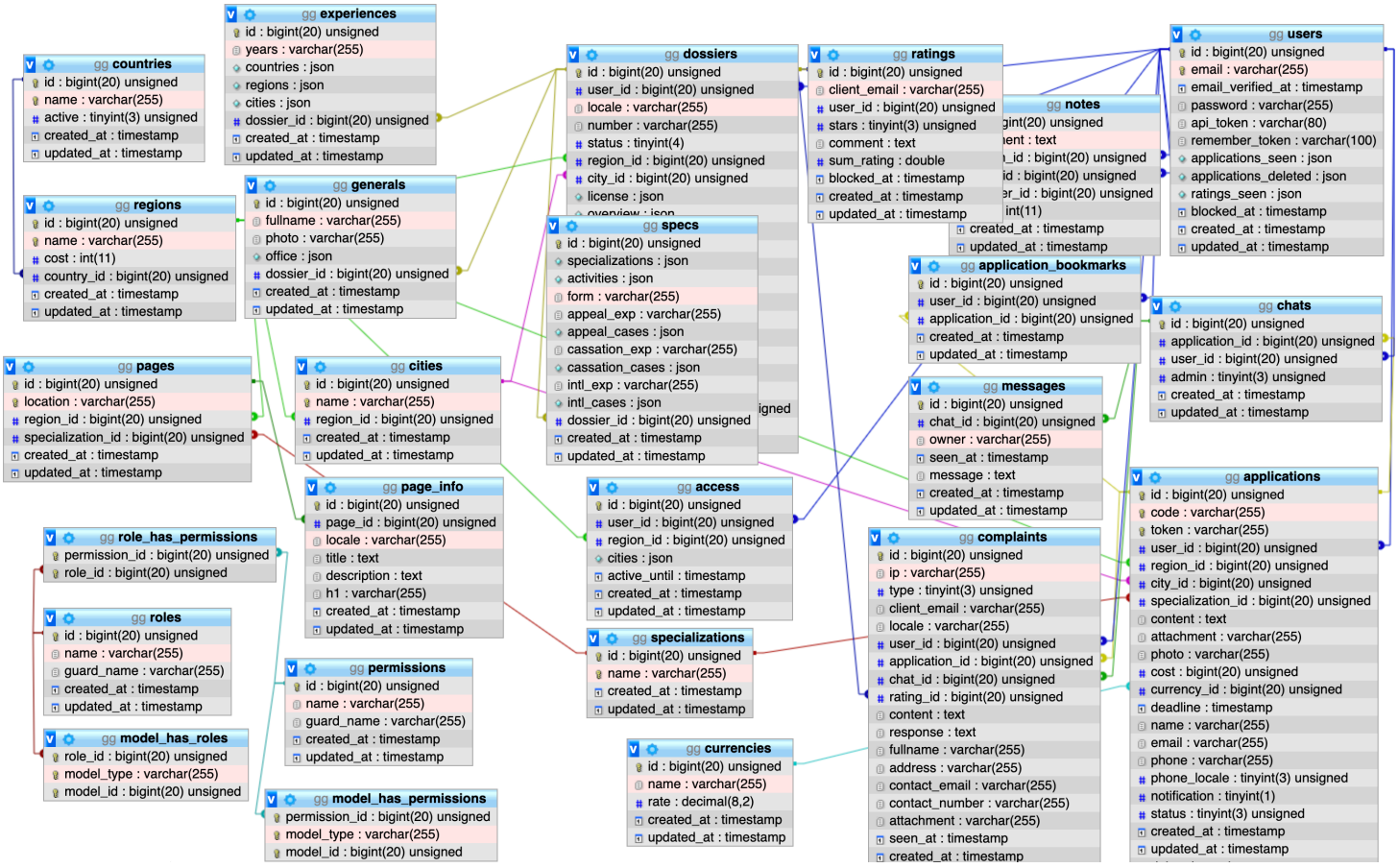

Рисунок 3.1 – Діаграма бази даних

3.3 Розробка модуля пошуку адвокатів

Однією з найважливіших частин веб-системи є модуль пошуку адвокатів з розширеним фільтром, що дозволяє врахувати багато важливих професійних критеріїв та пришвидшити пошук потрібного спеціаліста.

Для початку підключимо створимо клас DossierController, у якому буде функція, що здійснюватиме пошук адвокатів за заданими параметрами, підключимо у ньому необхідні для роботи класи.

```
use App\Http\Resources\DossierResource;
use App\Models\Dossier;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
```

Напишемо функцію для отримання досьє адвокатів з бази даних у вигляді масиву. Функцію напишемо в окремому класі DossierResource, який відповідатиме лише за отримання даних адвокатів з бази. Це дозволить уникнути дублювання коду та покращити його читабельність.

Частину функції наведено нижче:

```
if (!$regions->isEmpty()) {
    return [
        'id' => $this->id,
        'rating' => $this->getRatingAttribute(),
        'general' => $this->general()->first(),
        'experience' => $experience,
        'countries_id' => $countries,
        'user_id' => $user->id,
        'regions_id' => $regions->all(),
        'locale' => $this->locale,
        'user_dossiers_count' => $user->dossiers()->count(),
        'city' => $this->city ? $this->city->name : null,
        'region' => $this->region->name,
        'country' => $this->region->country->name,
        'spec' => $this->spec,
        'education_degree' => $degrees,
        'education_field' => $fields,
        'education_specialty' => $specialties,
        'ac_degree' => $ac_degrees,
        'teaching' => $teaching,
        'cassation' => $cassation_exp,
        'appeal_exp' => $appeal_exp,
        'cases' => $appeal_cases,
        'patent' => $patent,
        'publication' => $publications,
        'recommendation'=>$recommendations,
        'honor'=>$honors,
        'cost' => $cost,
        'count_doss' => $count_doss,
        'languages' => $languages,
        'seen_counter' => $this->seen_counter,
        'in_bookmarks' => session()->has('bookmark') && in_array($this->id, session( key: 'bookmark'))
    ];
}
```

У функції, яка буде фільтрувати досьє адвокатів, напишемо правила валідації параметрів, за якими буде здійснюватися пошук. Це дозволить запобігти виникненню помилок в роботі веб-системи у разі якщо користувачі будуть вводити некоректні дані.

```
$validated = $request->validate([
    'q' => 'nullable|string',
    'activities' => 'nullable|array',
    'years' => 'nullable|string',
    'specialization' => 'nullable|string',
    'degrees'=>'nullable|string',
    'fields'=>'nullable|string',
    'appeal'=>'nullable|string',
    'cases'=>'nullable|string',
    'cassation'=>'nullable|string',
    'patent'=>'nullable|string',
    'honor'=>'nullable|string',
    'publication'=>'nullable|string',
    'recommendation'=>'nullable|string',
    'specialties'=>'nullable|string',
    'ac_degrees' => 'nullable|string',
    'teaching' => 'nullable|string',
    'sort_by' => 'string',
    'regions_id' => 'nullable',
    'countries_id' => 'nullable',
    'locale' => 'nullable',
    'rating' => 'int|nullable',
    'cost_from' => 'int|nullable',
    'cost_to' => 'int|nullable',
    'languages' => 'int|nullable',
    'per_page' => 'int'
]);
```

Отримаємо досьє використовуючи раніше написану функцію, яка повертає дані з бази даних.

```
$dossiers = DossierResource::collection(Dossier::where('status', Dossier::STATUS_ACTIVE)
    ->get())->toArray($request);

$dossiers = collect($dossiers)->filter(function ($value) { return !is_null($value); });
```

Перевіримо чи є досьє, які відповідають параметрам пошуку.

```
if (isset($validated['countries_id'])) {
    $needle = $validated['countries_id'];
    $dossiers = $dossiers->filter( function($dossier) use ($needle) {
        $result = array_search($needle, $dossier['countries_id']);
        return is_int($result);
    });
}
```

Уникнемо дублювання досьє адвокатів.

```
$dossiers = $dossiers->unique('user_id');
```

Врахуємо пагінацію та повернемо результат пошуку у форматі JSON.

```
$dossiers = $this->paginate($dossiers, $perPage = $validated['per_page'], $page = null,
    $options = ['path' => $request->url()]);

return response()->json($dossiers);
```

Напишемо функціонал на Vue.js, який буде викликати раніше створену функцію, та виводити дані користувачу.

```
axios.post( url: `/api/dossier`, data: {
    activities: this.activities,
    cost_from: this.cost_from,
    cost_to: this.cost_to,
    years: $('#select-years').children("option:selected").val(),
    degrees: this.degree_name ? this.degree_name : $('#select-degree').
    children("option:selected").val(),
    fields: this.field_name ? this.field_name : $('#select-field').children("option:selected").val()
    specialties: this.specialty_name ? this.specialty_name : $('#select-specialty').
    children("option:selected").val(),
    languages: this.language_name ? this.language_name : $('#select-language').
    children("option:selected").val(),
    ac_degrees: this.ac_degree_name ? this.ac_degree_name : $('#select-acdegree').
    children("option:selected").val(),
    specialization: this.category_specialization_name ? this.category_specialization_name :
        $('#select-specialization').children("option:selected").val(),
    rating: $('#select-rating').children("option:selected").val(),
    teaching: $('#select-teaching').children("option:selected").val(),
    cases:  $('#select-case').children("option:selected").val(),
    patent:  $('#select-patent').children("option:selected").val(),
    publication:  $('#select-publication').children("option:selected").val(),
    recommendation:  $('#select-recommendation').children("option:selected").val(),
    cassation:  $('#select-cassation').children("option:selected").val(),
    honor:  $('#select-honor').children("option:selected").val(),
    countries_id: $('#country-select-cabinet-general').children("option:selected").val(),
    per_page: this.dossiersToShow
})
    .then((response) => {
        this.dossiers = response.data.data;
        this.loaded = true;
    })
```

3.4 Розробка модуля онлайн консультацій

Ще однією важливою складовою веб-системи є реалізація можливості спілкуватися зі спеціалістами онлайн. Для зручності листування повідомленням надаються статуси (прочитано або не прочитано), які зберігаються в базу даних.

Створимо клас ChatsController, який відповідатиме за обмін повідомленнями. Підключимо необхідні для роботи класи.

```
use App\Models\Chat;
use App\Models\Message;
use App\Notifications\MessageFromClient;
use App\Notifications\MessageFromLawyer;
use App\Notifications\ReplyWithoutMessage;
use Carbon\Carbon;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Notification;
```

Напишемо функцію, яка створюватиме новий чат.

```
public function startChat(Request $request)
{
    $request->validate([
        'reply-message' => 'string|required'
    ]);
    $chat = new Chat([
        'user_id' => $request->input( key: 'user'),
        'application_id' => $request->input( key: 'application'),
    ]);
    $chat->save();
    $chat->messages()->create([
        'message' => $request->input( key: 'reply-message'),
        'owner' => 'user',
        'seen_at' => Carbon::now()
    ]);
    if($chat->application->notification == 1){
        Notification::route( channel: 'mail', $chat->application->email)
            ->notify(new MessageFromLawyer($chat));
    }


    return redirect()->route( route: 'cabinet.request', app()->getLocale())->with('notification', [
        'type' => 'success',
        'title' => __( key: 'modals.result_success'),
        'content' =>  __( key: 'modals.application_response_message')
    ]);
}
```

Для уникнення небажаного доступу до переписок напишемо функцію для перевірки прав користувачів.

```
public function checkAccess($chat, $user_id){
    if($chat->application){
        if(!($user_id != $chat->user_id || $user_id != $chat->application->user_id)){
            abort( code: 401);
        }
    }
    else if($user_id != $chat->user_id) {
        abort( code: 401);
    }
}
```

Напишемо функцію для отримання повідомлень з чату.

```
public function fetchMessages($id)
{
    $api_user = Auth::guard( name: 'api')->id();
    $chat = Chat::findOrFail([$id])->first();

    $this->checkAccess($chat, $api_user);

    return $chat->messages()->get();
}
```

Реалізуємо можливість відгуку на досьє адвоката без необхідності набирати повідомлення. Адвокат отримає автоматично надіслане стандартне повідомлення.

```
public function replyWithoutMessage(Request $request){

    $chat = new Chat([
        'user_id' => $request->input( key: 'user_id'),
        'application_id' => $request->input( key: 'application_id'),
    ]);
    $chat->save();
    $chat->messages()->create([
        'message' => __( key: 'mail.responded'),
        'owner' => 'user',
        'seen_at' => Carbon::now()
    ]);
    if($chat->application->notification == 1){
        Notification::route( channel: 'mail', $chat->application->email)
            ->notify(new ReplyWithoutMessage ($chat->application->id));

    }
```

Напишемо функцію для надсилання повідомлень у веб-системі, яка також надсилатиме сповіщення на електронну пошту користувача.

```
public function sendMessage(Request $request)
{
    $api_user = Auth::guard( name: 'api')->id();
    $chat = Chat::findOrFail([$request->input( key: 'chat_id')])->first();

    $this->checkAccess($chat, $api_user);

    if ($chat !== null) {
        $chat->messages()->create($request->all());
        if ($request->input( key: 'owner') == 'application') {
            $chat->save();
        }

        if($request->input( key: 'owner') == 'application'){
            $chat->user->notify(new MessageFromClient($chat));
        } else {
            Notification::route( channel: 'mail', $chat->application->email)
                ->notify(new MessageFromLawyer($chat));
        }
    }
}
```

Сформуємо лист, який отримає користувач на електронну пошту, у класі MessageFromLawyer.

```
public function toMail($notifiable)
{
    return (new MailMessage)
                ->subject(__( key: 'mail.from_client_subject'))
                ->line(__( key: 'mail.from_client_content'))
                ->line(__( key: 'mail.admin_press'))
                ->action(__( key: 'mail.link_go'), route( name: 'cabinet.mail', app()->getLocale()));
}
```

Реалізуємо зміну статусів повідомлень (прочитані або не прочитані).

```
public function updateSeenStatus($id)
{
    $api_user = Auth::guard( name: 'api')->id();
    $chat = Chat::findOrFail([$id])->first();

    $this->checkAccess($chat, $api_user);

    $chat->messages()->whereNull('seen_at')->update(['seen_at' => \Illuminate\Support\Carbon::now()]);
}
```

Напишемо функцію для видалення повідомлень.

```
public function deleteMessage($message_id){

    Message::find($message_id)->delete();

}
```

Клієнтську частину, яка буде виводити дані користувачам, та викликати створені раніше функції, реалізуємо в компонентах Vue.js.

Напишемо функцію, яка буде отримувати повідомлення.

```
fetchMessages() {
    axios.get( url: `/api/messages/${this.data.chat.id}`, this.api_config)
        .then(response => {
            this.messages = response.data;
        })
},
```

Напишемо функцію для видалення повідомлень.

```
deleteMessage(id){
    axios.post( url: `/api/messages/delete/${id}`,  data: {

    }, this.api_config);
    document.getElementById(id).style.display = 'none' ;
},
```

Реалізуємо на Vue.js функції перевірки доступу до чату, підрахунок нових повідомлень, зміну статусу повідомдень.

```
    methods: {
        setChatToShow(id) {
            this.chatToShow = id
        },
        updateSeenStatus(chat) {
            axios.post( url: `/api/chat/${chat.chat.id}/viewed`, data: {}, config: {
                headers: {
                    Authorization: `Bearer ` + chat.api_token,
                    Accept: 'application/json',
                }
            })
        },
        accessReport(id) {
            $('#modal-application-id').val(id)
        },
        updateMessagesCount(count){
            var counter = document.getElementById( elementId: 'messages_counter');
            counter.innerText = counter.innerText - count;

        }
    },
}
```

Реалізуємо на Vue.js відправку повідомлень. Напишемо функцію, яка надсилатиме на сервер POST запит та створюватиме нове повідомлення.

```
sendMessage() {
    if(this.newMessage === ''){
        return;
    }
    this.messages.push({
        owner: 'user',
        message: this.newMessage
    })
    axios.post( url: `/api/message/store`, data: {
        chat_id: this.item.chat.id,
        message: this.newMessage,
        owner: 'user'
    }, this.api_config)
        .catch(error => {
            this.messages.pop()
            alert(this.$t('validation.chat.session_expire'))
        })
        .then(() => {
        this.fetchMessages();
    });
    this.newMessage = ''
},
```

3.5 Налаштування розгортки веб-системи

Для пришвидшення та полегшення процесу розгортання веб-системи було вирішено використовувати Docker.

Docker (Докер) — програмне забезпечення з відкритим вихідним кодом, яке використовується для розробки, тестування, доставки та запуску веб-застосунків у середовищах з підтримкою контейнеризації [17]. Він необхідний для більш ефективного використання системи та ресурсів, швидкого розгортання готових програмних продуктів, а також для їх масштабування та перенесення в інші середовища з гарантованим збереженням стабільної роботи.

Також завдяки Docker було налаштовано сервер бази даних. Нижче наведено код конфігураційного файлу.

```
version: "3.7"
networks:
    app-network:
        driver: bridge
services:
    app:
        container_name: app
        build: docker/php
        working_dir: /app
        volumes:
            - ./:/app
        networks:
            - app-network

    nginx:
        image: nginx:latest
        volumes:
            - ./:/app
            - ./docker/nginx/vhost.conf:/etc/nginx/conf.d/default.conf
            - ./docker/nginx/nginx.conf:/etc/nginx/nginx.conf
        ports:
            - "80:80"
        networks:
            - app-network

    redis:
        image: redis
        restart: always
        networks:
```

```
            - app-network

    mysql:
        image: mysql:5.7
        environment:
            MYSQL_ROOT_PASSWORD: root
            MYSQL_DATABASE: respo_tools
            MYSQL_USER: admin
            MYSQL_PASSWORD: admin
        volumes:
            - ./docker/mysql/dumps:/docker-entrypoint-initdb.d
            - ./docker/mysql/data:/var/lib/mysql
        restart: always
        networks:
            - app-network
    adminer:
        image: adminer
        restart: always
        ports:
            - 8083:8080
        networks:
            - app-network

volumes:
    mongodbdata:
        driver: local
```

3.6 Висновки

У третьому розділі було обґрунтовано вибір мов програмування та технологій, які будуть використовуватися при розробці програмного продукту та наведено основні їх переваги. У результаті аналізу було обрано мову програмування PHP, фреймворк Laravel та JS фреймворк Vue.js для реалізації фронт-енд частини.

Було проведено розробку модулів для пошуку адвокатів та онлайн консультацій, описано налаштування для розгортки веб-системи.

4 ТЕСТУВАННЯ ВЕБ-СИСТЕМИ

4.1 Тестування веб-системи

Тестування веб-системи – це процес завдяки якому можна визначити, наскільки зручною і привабливою вона є для відвідувачів, наскільки швидко і легко на ній можна відшукати потрібну інформацію, отримати необхідні послуги [18]. Однак, найважливішою частиною даного тестування є перевірка відповідності бізнес логіці, яка повинна виконуватись веб-системою.

В першу чергу перевіримо головну сторінку сайту (рис. 4.1).

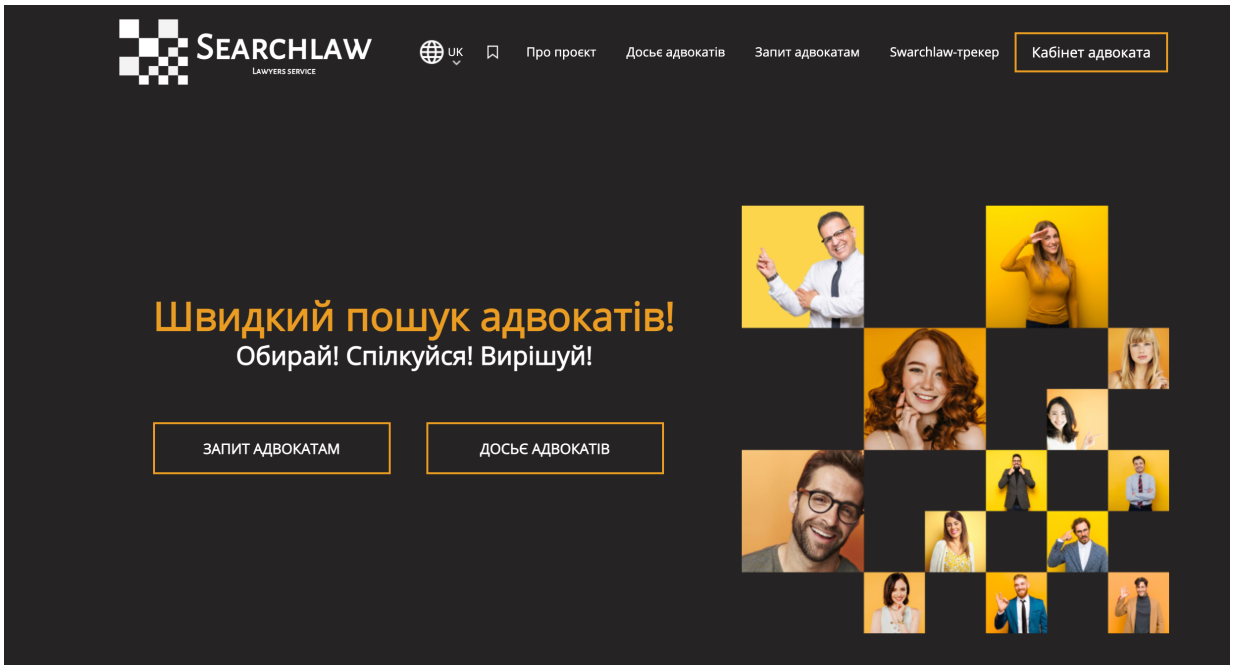

Рисунок 4.1 – Головна сторінка веб-системи

Перевіримо працездатність авторизації, зайдемо в кабінет адвоката.

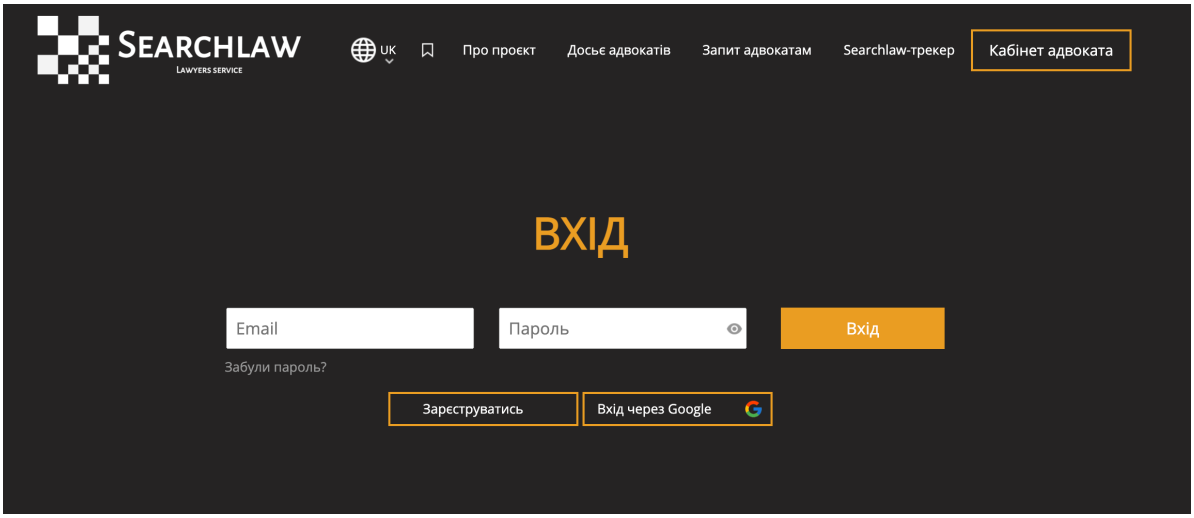

Рисунок 4.2 – Сторінка авторизації

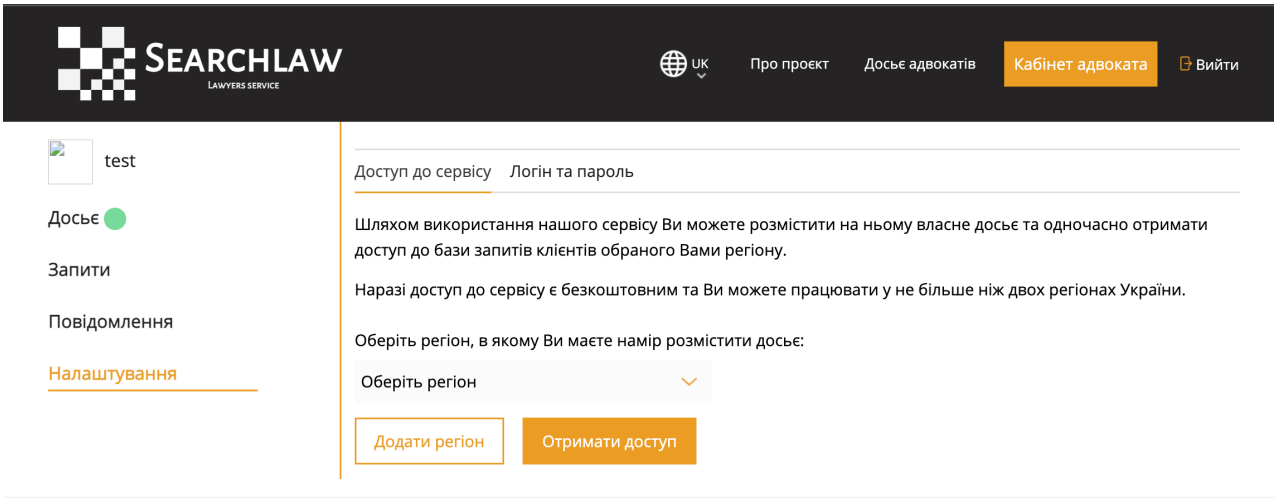

Рисунок 4.3 – Кабінет адвоката

Перш ніж створити досьє адвокат повинен отримати доступ до регіонів, з яких він хоче отримувати заявки від потенційних клієнтів. Безкоштовно можна отримати доступ до двох регіонів України. Доступ надається на місяць з можливістю поновлення. Без доступу до регіонів досьє створити неможливо, відповідно, неможливо і отримувати заявки від клієнтів.

Протестуємо отримання доступу до регіонів.

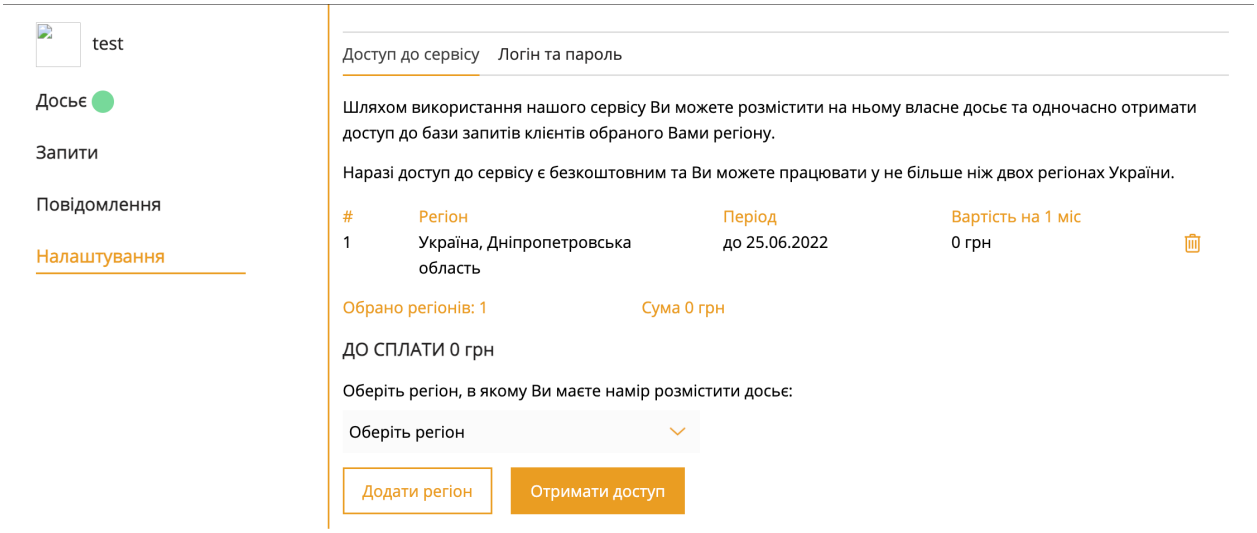

Рисунок 4.3 – Отримання доступу до регіону

Протестуємо створення досьє.

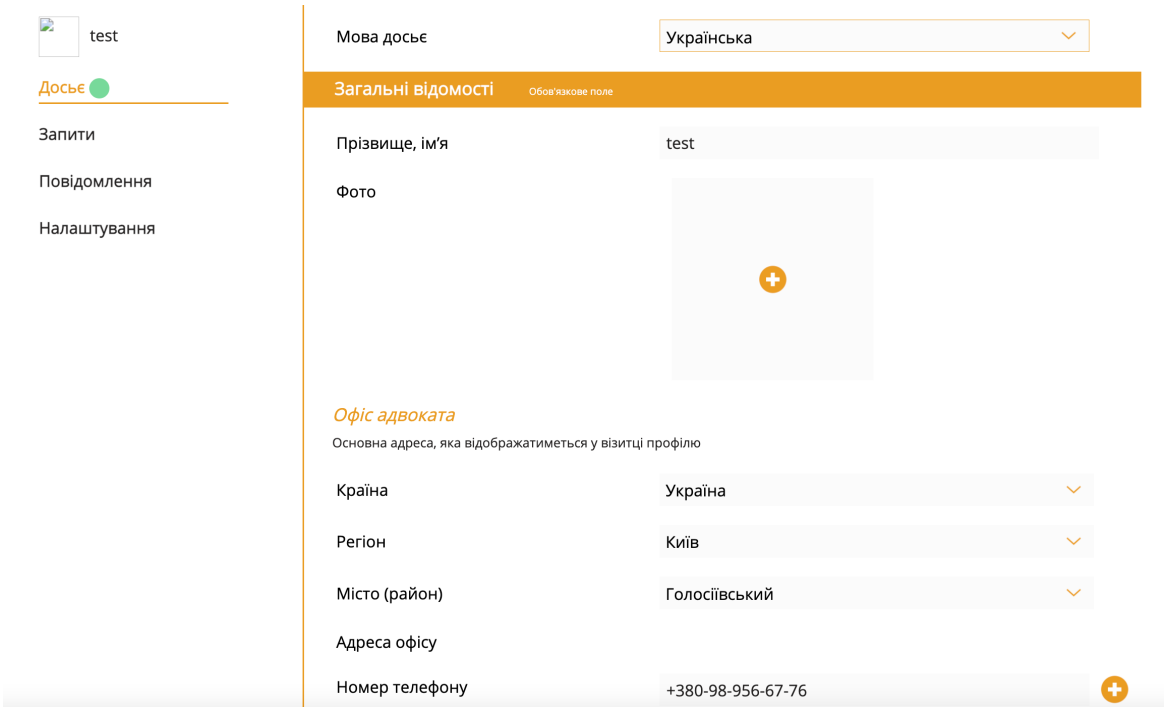

Рисунок 4.4 – Сторінка створення досьє адвоката

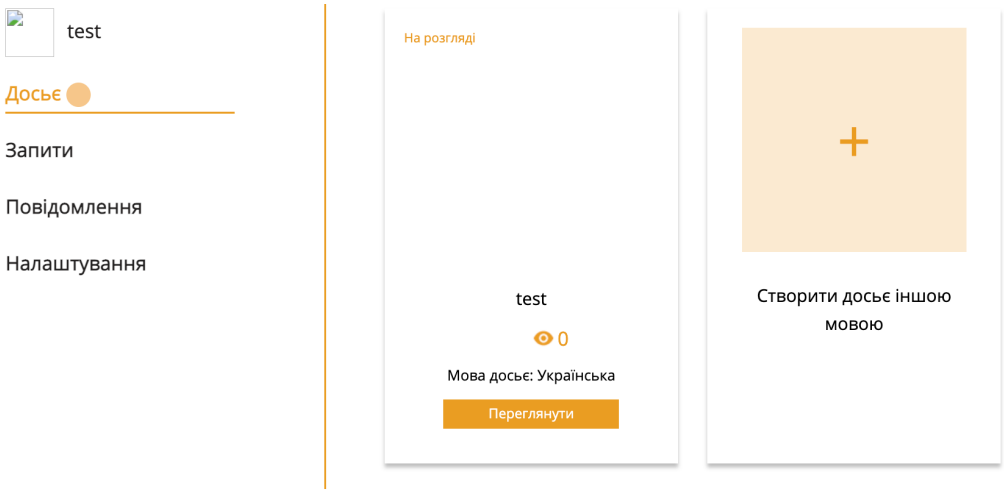

Рисунок 4.5 – Збережене досьє

Після створення досьє відправляється на розгляд адміністратором. Протестуємо чи з’явилось досьє в панелі адміністратора.

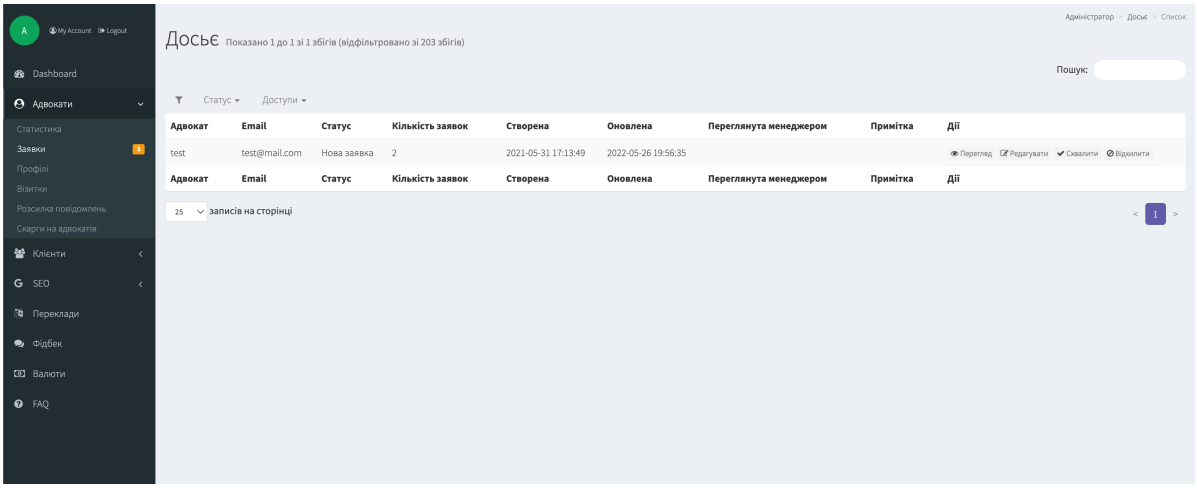

Рисунок 4.6 – Досьє в панелі адміністратора

Протестуємо пошук адвокатів та фільтрацію.

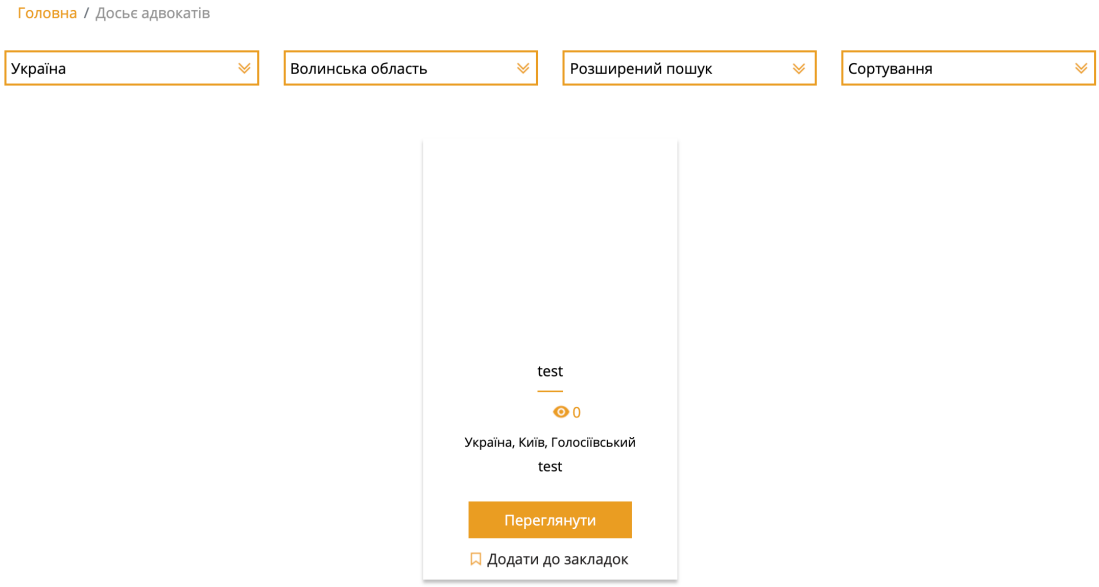

Рисунок 4.7 – Пошук адвокатів

Протестуємо надсилання запиту адвокатам.

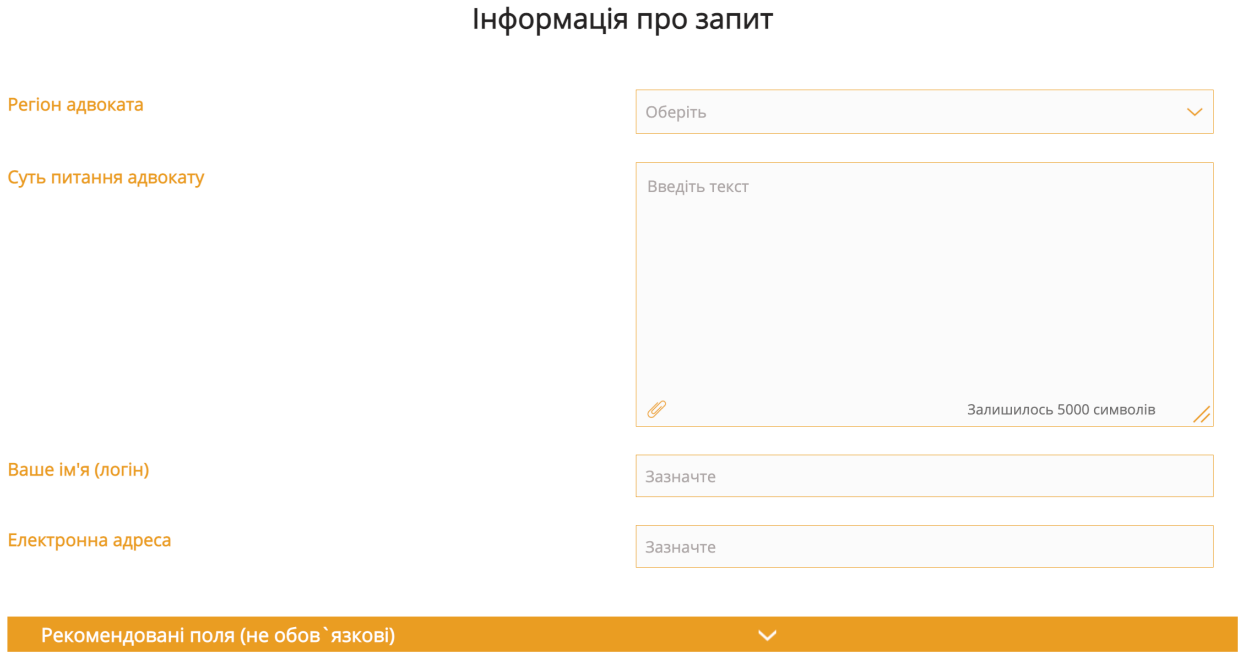

Рисунок 4.8 – Сторінка надсилання запиту адвокатам

Перевіримо чи з’явився запит в кабінеті адвоката.

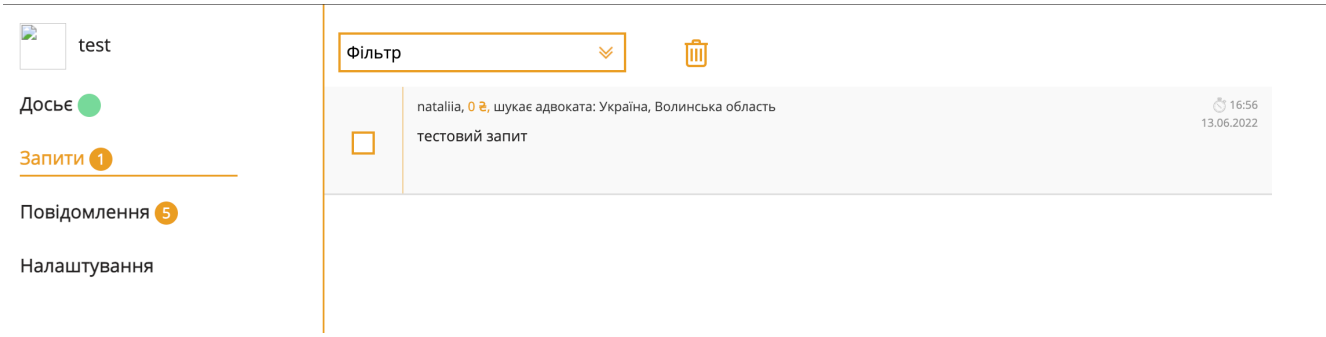

Рисунок 4.9 – Сторінка надсилання запиту адвокатам

Протестуємо відгук на запит клієнта.

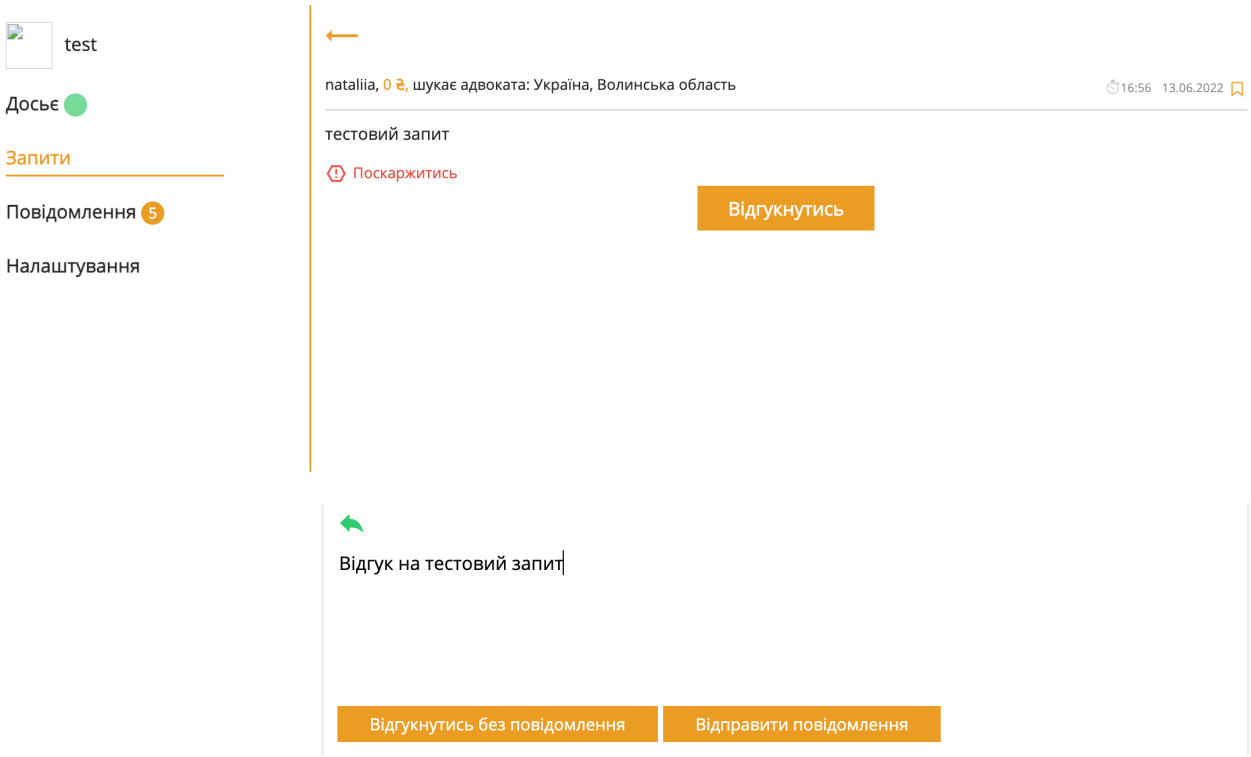

Рисунок 4.10 – Відповідь на запит клієнта

Протестуємо надсилання запиту на перевірку статусу заявки.

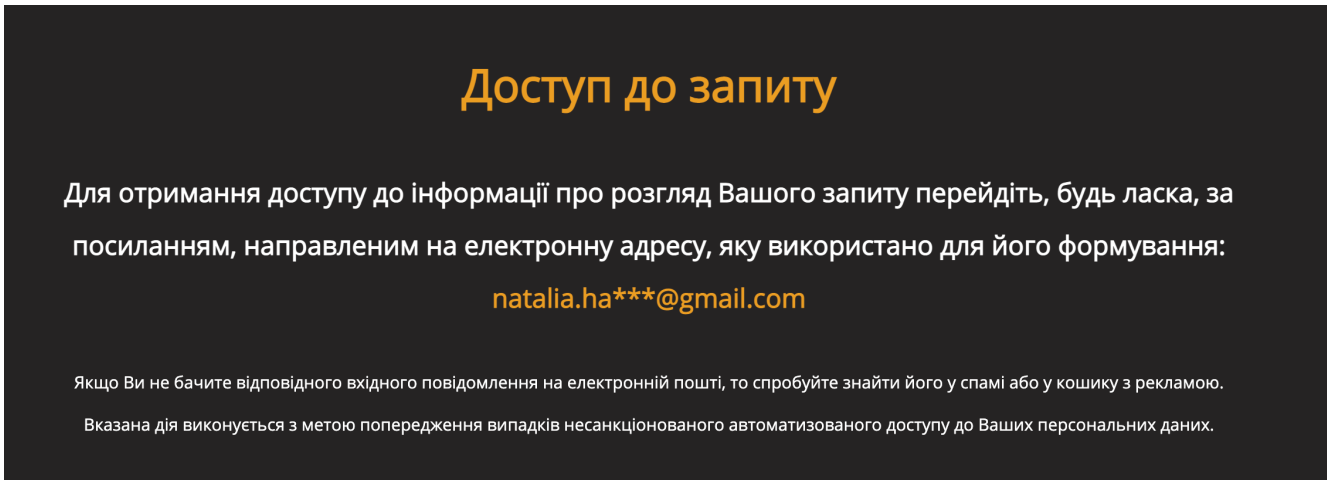

Рисунок 4.11 – Запит на перевірку статусу заявки

Протестуємо трекер статусу заявки.

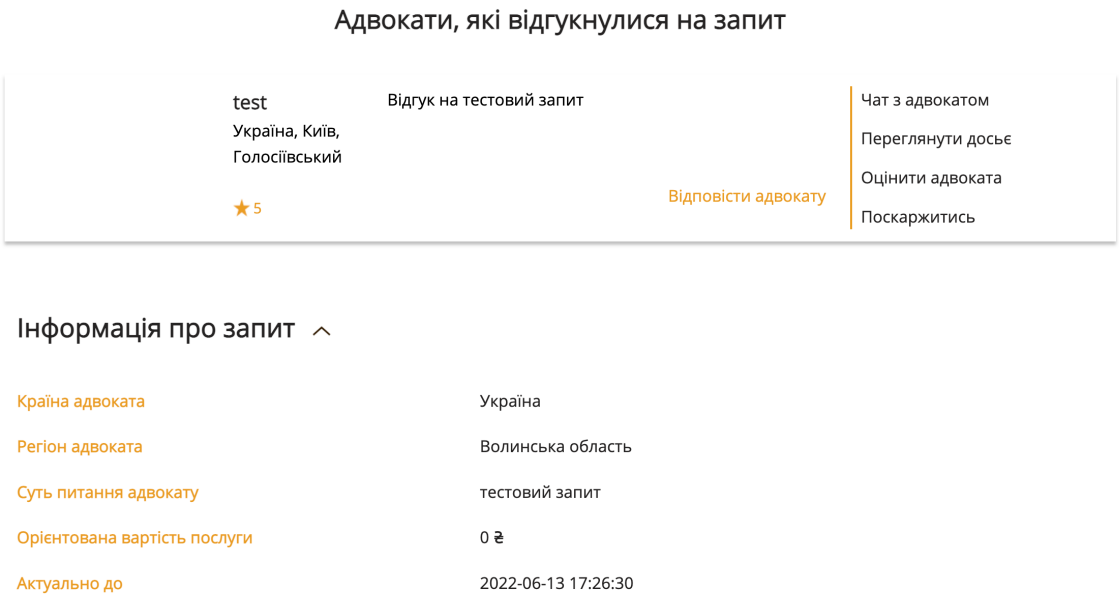

Рисунок 4.12 – Перевірка статусу заявки

Протестуємо надання відповіді адвокату.

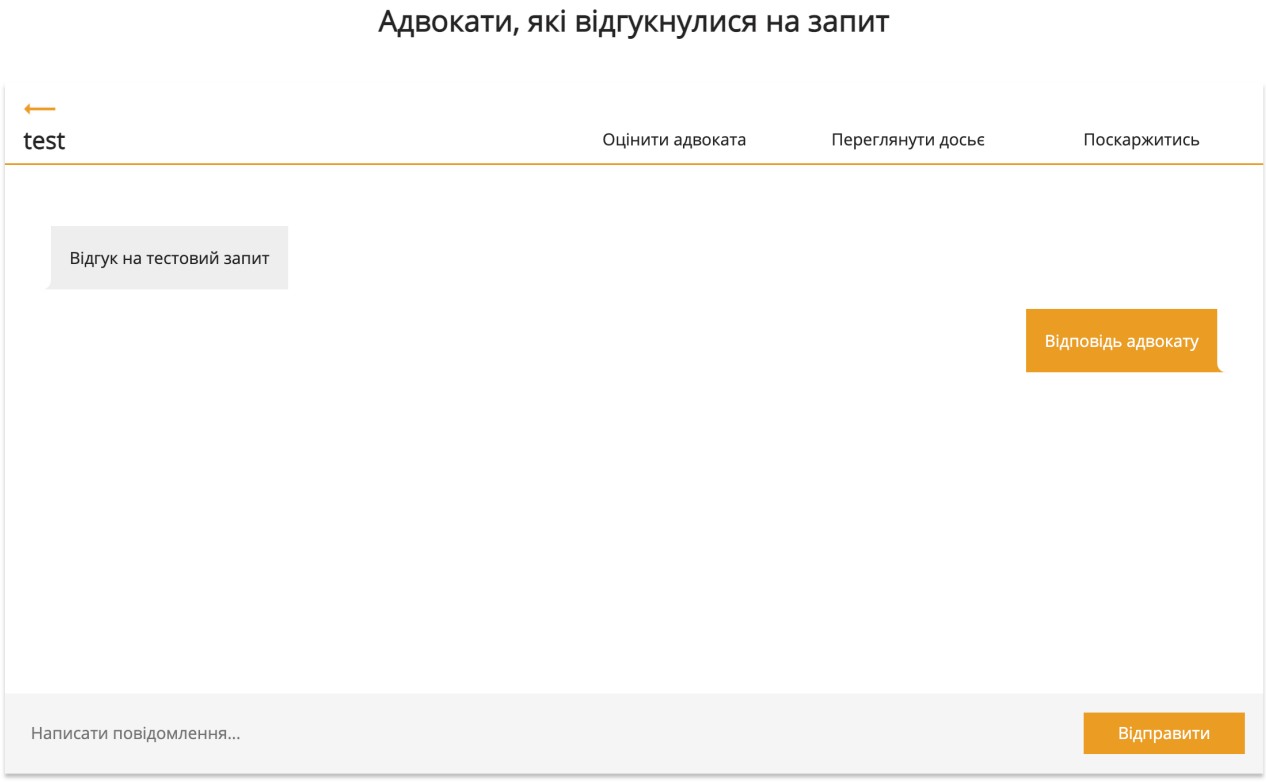

Рисунок 4.13 – Відповідь адвокату

У результаті тестування додатку було доведено, що його функціонал працює відповідно до технічного завдання.

4.2 Створення інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц
Об’єм оперативної пам’яті	1 ГБ для 32-розрядної системи і 2 ГБ для 64-разрядної системи
Місце на жорсткому диску	10 ГБ
Операційна система	Будь яка

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-разрядної системи
Розмір жорсткого диску	15 ГБ
Операційна система	Будь яка

Для того щоб запустити веб-систему користувачу необхідно мати встановлений Docker. Цей інструмент дозволяє гарантувати, що у будь якій системі проект буде працювати без помилок, та будуть встановлені всі необхідні компоненти [12]. Маючи вставлений Docker, для розгортання проекту користувачу потрібно буде лише запустити відповідну команду.

Після розгортання проекту користувач може авторизуватись, а може продовжити користування веб-системою без авторизації, якщо він не є адвокатом. Доступ до кабінету адвоката без авторизації отримати неможливо. У разі, якщо користувач забув пароль до особистого кабінету, або до кабінету адвоката, є можливість його відновити. Отримати доступ до панелі адміністратора можна лише перейшовши за спеціальним посиланням.

4.3 Висновки

У четвертому розділі було проведено тестування веб-системи для пошуку адвокатів та онлайн консультацій. Була простестована працездатнівсть сторінок системи, пошук, фільтрація, створення досьє. Адаптивність системи протестовано шляхом використання розширення екрану телефона iPhone 13. Також було розроблено інструкцію користувача по встановленню та початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено веб-систему для пошуку адвокатів з вбудованими модулями пошуку адвокатів та онлайн консультацій . Для розробки було використано середовище програмної розробки PHP Storm.

Під час виконання бакалаврської дипломної роботи було проаналізовано стан проблеми на сьогоднішній день. Було розглянуто основні аналоги програмного продукту. Було визначено їхні недоліки та порівняно з власним програмним продуктом. Базуючись на порівнянні було встановлено основні задачі бакалаврської дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування PHP та фреймворку Laravel. Також було розглянуто переваги використання сервера баз даних MySQL для збереження даних користувачів.

У бакалаврській дипломній роботі було зроблено наступне:

- розроблено базу даних відповідно до вимог веб-системи;
- розроблено модуль пошуку адвокатів;
- розроблено модуль для online консультацій;
- розроблено графічний інтерфейс для веб-середовища;
- налаштовано взаємодію між модулями пошуку адвокатів, online консультацій та графічним інтерфейсом;
- проведено тестування веб-системи згідно поставлених задач.

Був розроблений загальний алгоритм роботи веб-системи та модуля пошуку адвокатів. Також було розроблено метод пошуку адвокатів та модель роботи веб-системи.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифрова юриспруденція: як нові технології змінюють сферу права [Електронний ресурс] – Режим доступу до ресурсу: https://mind.ua/openmind/20197609-cifrova-yurisprudenciya-yak-novi-tehnologiyi-zminyuyut-sferu-praval

2. Legarithm [Електронний ресурс] – Режим доступу до ресурсу: https://legarithm.io/en/

3. Legist [Електронний ресурс] – Режим доступу до ресурсу: https://www.legist.online/

4. Protocol.ua [Електронний ресурс] – Режим доступу до ресурсу: https://protocol.ua/

5. Lawyer.ua [Електронний ресурс] – Режим доступу до ресурсу: https://lawyer.ua/

6. Кеш [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/

7. UML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/3404140-advantages-of-php/

8. PHP [Електронний ресурс] – Режим доступу до ресурсу: http://programming.in.ua/web-design/allphp/30-about-php.html/

9. Переваги PHP [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unified_Modeling_Language/

10. Переваги Laravel в розробці комерційних веб-додатків [Електронний ресурс] – Режим доступу до ресурсу: https://solardigital.com.ua/blog/razrabotka-na-laravel/

11. Що таке MVC [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/3886982-what-is-mvc/

12. Eloquent. Початок роботи [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.su/docs/5.4/eloquent

13. Шаблонізатор Blade [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.su/docs/8.x/blade/

14. Що таке Composer [Електронний ресурс] – Режим доступу до ресурсу: https://webkyrs.info/page/chto-takoe-composer-i-zachem-on-nuzhen/

15. Система керування базами даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: http://www.znannya.org/?view=mysql/

16. Міграції і початкові дані [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.su/docs/5.0/migrations/

17. Що таке Docker [Електронний ресурс] – Режим доступу до ресурсу: https://blog.ithillel.ua/articles/shcho-take-docker-prostimi-slovami-pro-konteynerizatsiyu

18. Тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalearning.com.ua/theory/about_qa/shpargalka-z-testuvannya/

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"31" березня 2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка web-платформи для пошуку адвокатів та онлайн консультацій»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доц. Г.Б. Ракитянська

"31" березня 2022 р.

Виконала:

_____студентка гр. 1ПІ-19мс Н.Д. Галушко

"31" березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка web-системи для пошуку адвокатів та онлайн консультацій ».

Галузь застосування – веб-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 12 від «7 » лютого 2022 р.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є удосконалення процесу пошуку юристів шляхом розробки модуля пошуку, який містить розширений фільтр, що враховує усі важливі для користувача критерії.

Призначення роботи – автоматизації процесу пошуку адвокатів та надання можливості проводити консультації в режимі online.

4. Вихідні дані для проведення НДР

1. Мартин Фаулер UML. Основы. 3-е издание. / Мартин Фаулер, Кендалл Скотт – Символ-Плюс, 2015. – 192 ст.
2. Matt A. Weisfeld The Object-Oriented Thought Process: навч. посіб. / Addison-Wesley Professional; 4th edition, 2013. – 336 ст.
3. Kevin Tatroe Programming PHP: Creating Dynamic Web Pages : навч. посіб. / O'Reilly Media; 4th edition, 2020. – 544 ст.
4. Robin Nixon Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites : навч. посіб. / O'Reilly Media; 3rd edition, 2014. – 786 ст.

5. Технічні вимоги

Вхідні дані – інформація користувачів,інформація адвокатів; вихідні дані – графічний інтерфейс з результатами пошуку адвокатів, графічний інтерфейс для проведення онлайн консультацій.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи:

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник:

Оригінальність	**73,5**
Схожість	26,5

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Галушко Н.Д.

Керівник роботи _____ Ракитянська Г.Б.

Додаток В – Лістинг програми

Серверна частина додатку

DossierController.php

```
namespace App\Http\Controllers\API;

use App\Http\Resources\DossierResource;
use App\Models\Dossier;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class DossierController extends Controller
{
    public function getAll(Request $request)
    {
        $validated = $request->validate([
            'q' => 'nullable|string',
            'activities' => 'nullable|array',
            'years' => 'nullable|string',
            'specialization' => 'nullable|string',
            'degrees'=>'nullable|string',
            'fields'=>'nullable|string',
            'appeal'=>'nullable|string',
            'cases'=>'nullable|string',
            'cassation'=>'nullable|string',
            'patent'=>'nullable|string',
            'honor'=>'nullable|string',
            'publication'=>'nullable|string',
            'recommendation'=>'nullable|string',
            'specialties'=>'nullable|string',
            'ac_degrees' => 'nullable|string',
            'teaching' => 'nullable|string',
            'sort_by' => 'string',
            'regions_id' => 'nullable',
            'countries_id' => 'nullable',
            'locale' => 'nullable',
            'rating' => 'int|nullable',
            'cost_from' => 'int|nullable',
            'cost_to' => 'int|nullable',
            'languages' => 'int|nullable',
            'per_page' => 'int'
        ]);
        $preparedRules = explode('_', $validated['sort_by']);
        $sortRules['sort'] = $preparedRules[0];
        $sortRules['option'] = $preparedRules[1];

        $dossiers = DossierResource::collection(Dossier::where('status',
Dossier::STATUS_ACTIVE)
            ->get())->toArray($request);

        $dossiers = collect($dossiers)->filter(function ($value) { return
!is_null($value); });

        if ($sortRules['option'] == 'desc') {
            $dossiers = $dossiers->sortByDesc($sortRules['sort'])->values();
        } else {
```

```
            $dossiers = $dossiers->sortBy($sortRules['sort'])->values();
        }

        if ($validated['activities'] != []) {
            foreach ($validated['activities'] as $activity) {
                $dossiers = $dossiers->filter( function($dossier) use ($activity)
{
                    $result = isset($dossier['spec']['activities']) ?
array_search($activity, $dossier['spec']['activities']) : false;
                    return is_int($result);
                });
            }
        }

        if (isset($validated['specialization'])) {
            $specialization = $validated['specialization'];
            $dossiers = $dossiers->filter( function($dossier) use
($specialization) {
                $result = isset($dossier['spec']['specializations']) ?
array_search($specialization, $dossier['spec']['specializations']) : false;
                return is_int($result);
            });
        }
        if (isset($validated['degrees'])) {
            $degree = $validated['degrees'];
            $dossiers = $dossiers->filter( function($dossier) use ($degree) {
                $result =  array_search($degree, $dossier['education_degree']) ??
false;
                return is_int($result);
            });
        }
        if (isset($validated['fields'])) {
            $field = $validated['fields'];
            $dossiers = $dossiers->filter( function($dossier) use ($field) {
                $result =  array_search($field, $dossier['education_field']) ??
false;
                return is_int($result);
            });
        }
        if (isset($validated['specialties'])) {
            $specialty = $validated['specialties'];
            $dossiers = $dossiers->filter( function($dossier) use ($specialty) {
                $result =  array_search($specialty,
$dossier['education_specialty']) ?? false;
                return is_int($result);
            });
        }
        if (isset($validated['ac_degrees'])) {
            $ac_degree = $validated['ac_degrees'];
            $dossiers = $dossiers->filter( function($dossier) use ($ac_degree) {
                $result =  array_search($ac_degree, $dossier['ac_degree']) ??
false;
                return is_int($result);
            });
        }

        if (isset($validated['languages'])) {
            $language = $validated['languages'];
            $dossiers = $dossiers->filter( function($dossier) use ($language) {
                $result =  array_search($language, $dossier['languages']) ??
false;
```

```
                return is_int($result);
            });
        }

        if (isset($validated['teaching'])) {
            $teaching = $validated['teaching'];

            $dossiers = $dossiers->filter( function($dossier) use ($teaching) {
                $result =  array_search($teaching, $dossier['teaching']) ?? false;
                return is_int($result);
            });
        }

        if (isset($validated['appeal'])) {
            $appeal = $validated['appeal'];

            $dossiers = $dossiers->filter( function($dossier) use ($appeal) {
                $result =  array_search($appeal, $dossier['appeal_exp']) ?? false;
                return is_int($result);
            });
        }
        if (isset($validated['cases'])) {
            $cases = $validated['cases'];

            $dossiers = $dossiers->filter( function($dossier) use ($cases) {
                $result =  array_search($cases, $dossier['cases']) ?? false;
                return is_int($result);
            });
        }

        if (isset($validated['cassation'])) {
            $cassation = $validated['cassation'];

            $dossiers = $dossiers->filter( function($dossier) use ($cassation) {
                $result =  array_search($cassation, $dossier['cassation']) ??
false;
                return is_int($result);
            });
        }

        if (isset($validated['patent'])) {
            $patent = $validated['patent'];

            $dossiers = $dossiers->filter( function($dossier) use ($patent) {
                $result =  array_search($patent, $dossier['patent']) ?? false;
                return is_int($result);
            });
        }

        if (isset($validated['publication'])) {
            $publication = $validated['publication'];

            $dossiers = $dossiers->filter( function($dossier) use ($publication) {
                $result =  array_search($publication, $dossier['publication']) ??
false;
                return is_int($result);
            });
        }

        if (isset($validated['recommendation'])) {
            $recommendation = $validated['recommendation'];
```

```
            $dossiers = $dossiers->filter( function($dossier) use
($recommendation) {
                $result =  array_search($recommendation,
$dossier['recommendation']) ?? false;
                return is_int($result);
            });
        }

        if (isset($validated['honor'])) {
            $honor = $validated['honor'];

            $dossiers = $dossiers->filter( function($dossier) use ($honor) {
                $result =  array_search($honor, $dossier['honor']) ?? false;
                return is_int($result);
            });
        }


        if (isset($validated['years'])) {
            $preparedRules = explode('_', $validated['years']);
            $filterRules['from'] = $preparedRules[0];
            $filterRules['to'] = $preparedRules[1];
            $dossiers = $dossiers->filter( function($dossier) use ($filterRules) {
                return isset($dossier['experience']['years']) ?
$dossier['experience']['years'] > $filterRules['from'] &&
$dossier['experience']['years'] < $filterRules['to'] : false;
            });
        }
        if (isset($validated['cost_from']) || isset($validated['cost_to'] )) {
            $filterRules['from'] = $validated['cost_from'];
            $filterRules['to'] = $validated['cost_to'] ;
            $dossiers = $dossiers->filter( function($dossier) use ($filterRules) {

                return isset($dossier['cost']) ? min($dossier['cost']) >=
$filterRules['from'] && max($dossier['cost']) <= $filterRules['to'] : false;
            });
        }


        if (isset($validated['rating'])) {
            $rating = $validated['rating'];
            $dossiers = $dossiers->filter( function($dossier) use ($rating) {
                if ($rating <= 4) {
                    return $dossier['rating'] ? $dossier['rating'] > $rating :
false;
                }
                return $dossier['rating'] ? $dossier['rating'] == $rating : false;
            });
        }

        if ($validated['q'] != '') {
            $needle = $validated['q'];
            $dossiers = $dossiers->filter(function ($item) use ($needle) {
                return false !== stristr($item['general']['fullname'], $needle);
            });
        }

        if (isset($validated['regions_id'])) {
            $needle = $validated['regions_id'];
            $dossiers = $dossiers->filter( function($dossier) use ($needle) {
```

```
                $result = array_search($needle, $dossier['regions_id']);
                return is_int($result);
            });
        }

        if (isset($validated['countries_id'])) {
            $needle = $validated['countries_id'];
            $dossiers = $dossiers->filter( function($dossier) use ($needle) {
                $result = array_search($needle, $dossier['countries_id']);
                return is_int($result);
            });
        }

        if (isset($validated['locale'])) {
            $locale = $validated['locale'];

            $dossiers = $dossiers->filter( function($dossier) use($locale) {
               // var_dump(is_int(array_search($locale, $dossier['count_doss'])));
                    if(is_int(array_search($locale, $dossier['count_doss'])) ==
true){

                        return  $dossier['locale'] == $locale;
                    }else{

                        return  $dossier['locale'] != $locale;
                    }


            });

        } else {
            $dossiers = $dossiers->filter( function($dossier) {
                return $dossier['locale'] == app()->getLocale() ||
$dossier['user_dossiers_count'] === 1;
            });
        }

        $dossiers = $dossiers->unique('user_id');

        $dossiers = $this->paginate($dossiers, $perPage = $validated['per_page'],
$page = null,
            $options = ['path' => $request->url()]);

        return response()->json($dossiers);

    }
}
```

ChatsController.php

```
namespace App\Http\Controllers\API;

use App\Models\Chat;
use App\Models\Message;
use App\Notifications\MessageFromClient;
use App\Notifications\MessageFromLawyer;
use App\Notifications\ReplyWithoutMessage;
use Carbon\Carbon;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
```

```
use Illuminate\Support\Facades\Notification;

class ChatsController extends Controller
{
    /**
     * Fetch all messages
     *
     * @param $id
     * @return Message
     */


    public function deleteMessage($message_id){

        Message::find($message_id)->delete();

    }
    public function checkAccess($chat, $user_id){
        if($chat->application){
            if(!($user_id != $chat->user_id || $user_id != $chat->application->user_id)){
                abort(401);
            }
        }
        else if($user_id != $chat->user_id) {
            abort(401);
        }
    }

    public function fetchMessages($id)
    {
        $api_user = Auth::guard('api')->id();
        $chat = Chat::findOrFail([$id])->first();

        $this->checkAccess($chat, $api_user);

        return $chat->messages()->get();
    }

    /**
     * Persist message to database
     *
     * @param Request $request
     * @return void
     */
    public function sendMessage(Request $request)
    {
        $api_user = Auth::guard('api')->id();
        $chat = Chat::findOrFail([$request->input('chat_id')])->first();

        $this->checkAccess($chat, $api_user);

        if ($chat !== null) {
            $chat->messages()->create($request->all());
            if ($request->input('owner') == 'application') {
                $chat->save();
            }

            if($request->input('owner') == 'application'){
                $chat->user->notify(new MessageFromClient($chat));
            } else {
```

```
                Notification::route('mail', $chat->application->email)
                    ->notify(new MessageFromLawyer($chat));
            }
        }
    }

    public function updateSeenStatus($id)
    {
        $api_user = Auth::guard('api')->id();
        $chat = Chat::findOrFail([$id])->first();

        $this->checkAccess($chat, $api_user);

        $chat->messages()->whereNull('seen_at')->update(['seen_at' =>
\Illuminate\Support\Carbon::now()]);
    }

    public function startChat(Request $request)
    {
        $request->validate([
            'reply-message' => 'string|required'
        ]);
        $chat = new Chat([
            'user_id' => $request->input('user'),
            'application_id' => $request->input('application'),
        ]);
        $chat->save();
        $chat->messages()->create([
            'message' => $request->input('reply-message'),
            'owner' => 'user',
            'seen_at' => Carbon::now()
        ]);
        if($chat->application->notification == 1){
            Notification::route('mail', $chat->application->email)
                ->notify(new MessageFromLawyer($chat));
        }


        return redirect()->route('cabinet.request', app()->getLocale())-
>with('notification', [
            'type' => 'success',
            'title' => __('modals.result_success'),
            'content' =>  __('modals.application_response_message')
        ]);
    }


    public function replyWithoutMessage(Request $request){

        $chat = new Chat([
            'user_id' => $request->input('user_id'),
            'application_id' => $request->input('application_id'),
        ]);
        $chat->save();
        $chat->messages()->create([
            'message' => __('mail.responded'),
            'owner' => 'user',
            'seen_at' => Carbon::now()
        ]);
        if($chat->application->notification == 1){
            Notification::route('mail', $chat->application->email)
```

```
                ->notify(new ReplyWithoutMessage ($chat->application->id));

        }


    }
}
```

TrackerController.php

```
namespace App\Http\Controllers\API;

use App\Models\Application;
use App\Events\AccessRequested;
use App\Events\RemindCode;
use App\Notifications\RequestCode;
use App\Notifications\TrackerAccess;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Notification;
use Illuminate\Validation\Rule;

class TrackerController extends Controller
{
    public function getApplication(Request $request)
    {
        app()->setLocale($request->get('locale'));
        $request->validate([
            'code' => [
                'required',
                'string',
                'size:12',
                Rule::exists('applications')->where(function ($query) use
($request) {
                    $query->where('status', Application::STATUS_ACTIVE);
                }),
            ],
        ]);
        $application = Application::where('code', '=', $request->code)->first();
        return response()->json($application->email, 200);
    }

    public function sendAccessMail(Request $request)
    {
        app()->setLocale($request->get('locale'));
        $request->validate([
            'code' => [
                'required',
                'string',
                'size:12',
                Rule::exists('applications')->where(function ($query) use
($request) {
                    $query->where('code', $request->code)->where('status',
Application::STATUS_ACTIVE);
                }),
            ],
        ]);
        $application = Application::where('code', '=', $request->code)->first();
        Notification::route('mail', $application->email)
            ->notify(new TrackerAccess($application));
        return response(null, 200);
```

```
    }

    public function remindCodeMail(Request $request)
    {
        app()->setLocale($request->get('locale'));
        $request->validate([
            'email' => 'email|required|string|exists:applications,email'
        ]);
        $applications = Application::where('email', '=', $request->email)-
>where('status', Application::STATUS_ACTIVE)->get();
        event(new RemindCode($applications));
        return response(null, 200);
    }
}
```

AccessController.php

```
namespace App\Http\Controllers;

use App\Models\Access;
use App\Models\Country;
use App\Models\Dossier;
use App\Models\Page;
use App\Models\Region;
use Illuminate\Http\Request;
use Illuminate\Support\Carbon;

class AccessController extends Controller
{
    public function showForm($locale, $location = "cabinet.settings.access",
Request $request, $js = ['js/access.js'])
    {

        if(auth()->user()->dossiers()->count() == 0 ){
            return redirect()->route('cabinet.dossier.index', app()->getLocale())-
>with('notification', [
                'type' => 'success',
                'title' => __('modals.access_no_dossier'),
                'content' => __('modals.access_no_dossier_content')
            ]);
        }
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        $countries = Country::where('active', 1)->get();
        $access = new Access();
        $accesses = auth()->user()->access->all();

        $accesses = auth()->user()->access;

        $accesses = $accesses->map(function ($item) {
            return [
                'id' => $item->id,
                'region_id' => $item->region_id,
                'region' => __('regions.'.$item->region->name),
                'active_until' => date_format($item->active_until, 'Y-m-d' ),
                'created_at' => date_format($item->created_at, 'Y-m-d' ),
                'status' => $item->active_until > Carbon::now() ? 'active' : 'not
active'


            ];
```

```
        });


        return view('site.index', compact(['location', 'title', 'countries',
'access', 'js','accesses']));
    }
    public function showEditForm($locale, Access $access, $location =
"cabinet.settings.access", Request $request)
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        if($access->user_id != auth()->id()){
            abort(401);
        }
        $countries = Country::where('active', 1)->get();
        return view('site.index', compact(['location', 'title', 'countries',
'access']));
    }

    public function create(Request $request){
        if(auth()->user()->access()->where('active_until', '>=', Carbon::now())-
>count() >= 2){
            return response()->json([], 403);
        }
        $request->validate([
            'region_id' => 'required',
        ]);
        //todo calculate price with discounts and navigate to merchant
        if(is_array($request->region_id)){
            foreach ($request->region_id as $region) {
                $access = Access::firstOrNew([
                    'user_id' => auth()->id(),
                    'region_id' => $region['region_id'],
                ]);

                $until = $access->active_until && $access->active_until-
>gt(Carbon::now()) ? $access->active_until : Carbon::now();
                $access->active_until = $until->addDays(30);
                $access->save();
            }
        }else{

            $access = Access::firstOrNew([
                'user_id' => auth()->id(),
                'region_id' => 1,
            ]);


       $until = $access->active_until && $access->active_until->gt(Carbon::now())
? $access->active_until : Carbon::now();

            $access->active_until = $until->addDays(30);
            $access->save();
        }

        return route('cabinet.access', app()->getLocale());
    }

    public function regions(Request $request){
        if ($id = $request->post('country_id')){
            $regions = Region::select(['name', 'id', 'cost'])->where('country_id',
```

```
$id)->get();

            return response()->json($regions->map(function($item){
                if($item['active_until']){
                    $active = Carbon::createFromTimeString($item['active_until']);
                    if($active->diffInDays(Carbon::now()) > 7){
                        return null;
                    } else $item['active'] = $active->format('d.m.Y');
                } else {
                    $item['active'] = null;
                }
                $item['name'] = __('regions.'.$item["name"]);
                return $item;
            }));
        }
    }

//    public function update($locale, Access $access, Request $request){
//        if($access->user_id != auth()->id()){
//            abort(401);
//        }
//        return $this->store($request, $access);
//    }

    public function history($locale, $location = "cabinet.settings.access-
history", Request $request)
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        $access = auth()->user()->access->all();
        return view('site.index', compact(['location', 'title', 'access']));
    }

    public function toggleActive(Request $request){
        $access = Access::findOrFail($request->post('id'));
        if($access->user_id !== auth()->id()){
            abort(401);
        }
        $access->active = !$access->active;
        $access->save();

        return back();
    }

    public function toggleActiveNew(Request $request){
        $access = Access::findOrFail($request->post('id'));
        if($access->user_id !== auth()->id()){
            abort(401);
        }
        if ($access->active_until->gt(Carbon::now())){
            $access->active_until = Carbon::now()->subMinutes(1); // expired date
            $access->save();

            return back();
        }

        return redirect()->route('cabinet.access', app()->getLocale());
    }

//    public function delete(Request $request){
//        $access = Access::findOrFail($request->post('id'));
```

```
//         if($access->user_id !== auth()->id()){
//             abort(401);
//         }
//
//         $access->delete();
//
//         return back();
//     }
}
```

ApplicationController.php

```
namespace App\Http\Controllers;

use App\Models\Chat;
use App\Models\Country;
use App\Models\Currency;
use App\Events\ApplicationView;
use App\Models\Page;
use App\Models\Region;
use App\Models\City;
use App\Models\Application;
use App\Models\Specialization;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Carbon;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Cookie;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;

class ApplicationController extends Controller
{
    public function __construct()
    {
        $this->middleware('throttle:5')->only('verify');
    }

    public function rules(){
        return [
            'country_id' => 'required|integer',
            'region_id' => 'required|integer',
            'city_id' => 'nullable|integer',
            'specialization_id' => 'nullable|integer',
            'currency_id' => 'required',
            'content' => 'required|max:5000',
            'cost' => 'nullable|integer|min:0|max:100000',
            'deadline_type' => 'nullable|integer',
            'name' => 'required|min:3|max:50',
            'email' => 'required|email',
            'phone' => 'max:100',
            'attachment' => 'mimetypes:image/*,application/vnd.openxmlformats-
officedocument.wordprocessingml.document,application/msword,application/pdf|max:15
000',
            'agreement' => 'accepted',
```

```
            //'photo' => 'image|mimes:jpeg,png,jpg,gif'
        ];
    }

    public function index($locale, $location = "cabinet.request.index", Request
$request)
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        $query = Application::getVerifiedNotDeleted();
       // $applications = Application::filter($query, $request)->paginate(10);
        $applications_ = Application::get();
        $regions = Region::whereIn('id',auth()->user()->getActiveRegions())-
>get();
        $min_cost = Application::min('cost');
        $max_cost = Application::max('cost');
        foreach ($applications_ as $application){

            if( strtotime($application->getDeadline().'+ 1 days') <
strtotime(Carbon::now())){

                $application->delete();
            }
        }
        $applications = Application::filter($query, $request)->paginate(10);


        return view('site.index', compact(['location', 'title',
'applications','regions','min_cost','max_cost']));
    }

    public function show($locale, Application $application, $location =
"cabinet.request.single", Request $request)
    {
        if(!in_array($application->region_id, auth()->user()->getActiveRegions()-
>toArray())){
            return back()->with('notification', [
                'type' => 'success',
                'title' => __('modals.result_error'),
                'content' => __('modals.application_noaccess')
            ]);
        }
        $chat = Chat::query()
            ->where('application_id', '=', $application->id)
            ->where('user_id', '=', auth()->id())
            ->first();
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        event(new ApplicationView($application));
        return view('site.index', compact(['location', 'title', 'application',
'chat']));
    }

    public function requestForm($locale, $location = "site.request.index", Request
$request, $js = ['js/request.js'])
    {
        $allowEmailChange = true;
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        $countries = Country::where('active', 1)->get();
        $specializations = Specialization::all();
```

```
        $application = new Application();
        $currencies = Currency::all();
        return view('site.index', compact(['location', 'title', 'countries',
'specializations', 'currencies', 'js', 'allowEmailChange']))-
>withApplication($application);
    }

    public function create(Request $request){
        $application = new Application();

//        if ($request->hasFile('photo')){
//            $attachment = str_replace('public','storage',$request-
>file('photo')->store('request_photo'));
//            $application->photo = $attachment;
//        }

        if (isset($request->photo)){
            //dd($request->photo);
           // $attachment = str_replace('public','storage',$request-
>file('photo')->store('request_photo'));
            $application->photo = \GuzzleHttp\json_encode($request->photo);
        }


        return $this->store($request, $application);
    }

    public function update(Request $request){

        $application = Application::findOrFail(Cookie::get('application_id'));


        return $this->change($request, $application);
    }

    public function changeEmail($locale, Request $request){
        return $this->showEditForm($locale, $allowEmailChange = true,
$needVerification = true, $request);
    }

    public function showEditForm($locale, $allowEmailChange = false,
$needVerification = false, Request $request, $location = "site.request.index", $js
= ['js/request.js']){
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        $application = Application::find(Cookie::get('application_id'));
        if(is_null($application)){
            return redirect()->route('tracker.index', app()->getLocale());
        }
        $countries = Country::where('active', 1)->get();
        $specializations = Specialization::all();
        $currencies = Currency::all();
        $old_file = $application->attachment;
        return view('site.index', compact(['location', 'title', 'countries',
'specializations', 'currencies', 'js', 'old_file', 'allowEmailChange',
'needVerification']))->withApplication($application)->with('update', true);
    }

    public function store($request, $application){
        $validator = Validator::make($request->except('_token'), $this->rules());
        if ($validator->fails()) {
```

```
            return back()
                ->withErrors($validator)
                ->withInput();
        }


        if (Application::where('email', $request->email)->where('status',
Application::STATUS_ACTIVE)->count() > 3){
            return back()->withInput()->with('notification', [
                'type' => 'error',
                'title' => __('modals.result_error'),
                'content' => __('modals.application_more_than_3')
            ]);
        }
        $attachment = null;
        if ($request->hasFile('attachment')){
            $attachment = $request->file('attachment')-
>store('request_attachments');
        }
        if ($request->photo){
            //$application->photo =  str_replace('public','storage',$request-
>file('photo')->store('request_photo'));
            $application->photo = json_encode($request->photo);
        }
        $application->notification = $request->notification;

        foreach($request->except(['_token', 'agreement', 'country_id', 'cost',
'deadline_type', 'formatted-phone', 'attachment','photo','notification']) as $k =>
$v){
            $application->$k = $v;
        }

        $application->attachment = $attachment;
        if($request->cost){
            $application->cost = $request->cost;
        }
        do{
            $token = Str::random(60);
        }
        while (Application::where('token', $token)->first());

        $application->token = $token;
        $application->deadline = $application->calculateDeadline($request-
>deadline_type);
        $application->save();

        $user = User::firstOrCreate(['email' => $application->email], [
            'api_token' => Str::random(60),
            'password' => Hash::make(Str::random()),
            'email' => $request->email,
            'created_at' => now(),
            'updated_at' => now(),
        ]);

        if (!$user->hasRole('Клієнт')){
            $user->assignRole('Клієнт');
        }

        $application->user_id = $user->id;

        $application->save();
```

```
        $application->sendVerificationLink();

        Cookie::queue('application_id', $application->id, 60);
//        session()->put('application_id', $application->id);

        return redirect()->route('request.verification', app()->getLocale())-
>with('email', $application->email)->withInput();
    }

    public function change($request, $application){

        $validator = Validator::make($request->except('_token'), [
            'country_id' => 'required|integer',
            'region_id' => 'required|integer',
            'city_id' => 'nullable|integer',
            'specialization_id' => 'nullable|integer',
            'currency_id' => 'required',
            'content' => 'required|max:5000',
            'cost' => 'nullable|integer|min:0|max:100000',
            'deadline_type' => 'nullable|integer',
            'name' => 'required|min:3|max:50',
            'email' => 'required|email',
            'phone' => 'max:100',
            'attachment' => 'mimetypes:image/*,application/vnd.openxmlformats-
officedocument.wordprocessingml.document,application/msword,application/pdf|max:15
000',
        ]);

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->with('update', true)
                ->withInput();
        }

        if ($request->hasFile('attachment')){
            $application->attachment = $request->file('attachment')-
>store('request_attachments');
        } elseif($request->old_file) {
            $application_attachment = $request->old_file;
        }

        foreach($request->except(['_token', 'agreement', 'country_id', 'cost',
'deadline_type', 'notification', 'formatted-phone', 'attachment', 'old_file',
'need_verification']) as $k => $v){
            $application->$k = $v;
        }
        if($request->cost){
            $application->cost = $request->cost;
        }

        $application->deadline = $application->calculateDeadline($request-
>deadline_type);
        $application->save();

        $user = User::firstOrCreate(['email' => $application->email], [
            'api_token' => Str::random(60),
            'password' => Hash::make(Str::random()),
            'email' => $request->email,
            'created_at' => now(),
```

```
            'updated_at' => now(),
        ]);
        if (!$user->hasRole('Клієнт')){
            $user->assignRole('Клієнт');
        }

        $application->user_id = $user->id;
        $application->save();
        if($request->need_verification){
            $application->sendVerificationLink();
        }

        Cookie::queue('application_id', $application->id, 60);

        if($request->need_verification){

            return redirect()->route('request.verification', app()->getLocale())->withEmail($application->email);
        }
        return redirect()->route('tracker.application', app()->getLocale())->with('notification', [
            'title' => __('modals.result_success'),
            'type' => 'success',
            'content' => __('modals.request_changed')
        ]);
    }

    public function verification($locale, Request $request, $location = "site.request.verification"){
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        return view('site.index', compact(['location', 'title']));
    }

    public function verify(Request $request){
        if (!$request->hasValidSignature()) {
            abort(401);
        }

        $user_request = Application::where('token', $request->query('token'))->firstOrFail();

        if (!is_null($user_request->code)){ // already verified
            return redirect()->route('index', app()->getLocale())->with('notification', [
                'type' => 'error',
                'title' => __('modals.result_error'),
                'content' => __('modals.application_confirmed')
            ]);
        }
        $user_request->generateCode();
        $user_request->updated_at = $user_request->created_at;
        $user_request->save();

        return redirect()->route('request.verified', app()->getLocale())->with('code', $user_request->code);
    }

    public function verified($locale, Request $request, $location = "site.request.verified"){
        $title = Page::getTitle($location);
```

```
        $description = Page::getDescription($location);
        return view('site.index', compact(['location', 'title']));
    }

    public function sendCode(Request $request){
        /** @var Application $user_request */
        $user_request = Application::where(['code' => $request->input('code')])->firstOrFail();
        $user_request->sendCode();

        return redirect()->route('tracker.application', app()->getLocale());
    }

    public function stop(){
        $application = Application::findOrFail(Cookie::get('application_id'));
        $application->status = Application::STATUS_STOPPED;
        $application->save();
        Cookie::queue(Cookie::forget('application_id'));
//        session()->forget('application_id');
        return redirect()->route('index', app()->getLocale())->with('notification', [
            'type' => 'success',
            'title' => __('modals.result_success'),
            'content' => __('site.tracker.stopped')
        ]);
    }

    public function exit(){
        Cookie::queue(Cookie::forget('application_id'));
        return redirect()->route('tracker.index', app()->getLocale());
    }

    public function delete($locale, Request $request){
        if(is_array($id = $request->input('id'))){
            auth()->user()->deleteApplications($id);
        } else {
            auth()->user()->deleteApplication($id);
        }
        return back();
    }

    public function regions(Request $request){
        if ($id = $request->post('country_id')){
            $regions = Region::select(['name', 'id'])->where('country_id', $id)->get();
            return response()->json($regions->map(function($item){
                $item['name'] = __('regions.'.$item["name"]);
                return $item;
            })->toJson());
        }
    }

    public function cities(Request $request){
        if ($id = $request->post('region_id')){
            $cities = City::select(['id', 'name'])->where('region_id', $id)->get();
            return response()->json($cities->map(function($item){
                $item['name'] = __('cities.'.$item["name"]);
                return $item;
            })->toJson());
        }
```

```
    }
}
```

CabinetController.php

```
namespace App\Http\Controllers;

use App\Models\Application;
use App\Models\Page;
use App\Models\Chat;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Pagination\LengthAwarePaginator;
use Illuminate\Pagination\Paginator;
use Illuminate\Support\Collection;

class CabinetController extends Controller
{
    public function index($locale, $location = "cabinet.home.index", Request
$request)
    {
       // $title = Page::getTitle($location);
        //return view('site.index', compact(['location', 'title']));redirect()-
>route('login');
        return redirect()->route('cabinet.dossier.index', app()->getLocale());
    }
    public function mail($locale, $location = "cabinet.mail.index", Request
$request, $js = ['js/cabinet-chat.js'])
    {
        $title = Page::getTitle($location);
        $chats = null;
        $responseListData = [];
        if (auth()->user()->chats()->first() != null) {
            $chats = Chat::where('user_id', '=', auth()->id())->get();
            foreach ($chats as $chat) {
                $application = Application::find($chat->application_id);
                $last_message = $chat->messages()->get()->last();
                if (!is_null($last_message)){
                    $responseListData[] = [
                        'chat' => $chat,
                        'lastMessage' => $last_message->message,
                        'application' => $application,
                        'application_link' => route('cabinet.request-single',
['application' => $application, 'locale' => app()->getLocale()]),
                        'date' => $last_message->dateCreated(),
                        'chat_seen' => $last_message->seen_at,
                        'time' => $last_message->timeCreated(),
                        'date_time' => $last_message->created_at,
                        'new_messages' => $chat->messages()->where('seen_at', '=',
null)->count(),
                        'api_token' => auth()->user()->api_token
                    ];
                }

            }
        }
        $responseListData = collect($responseListData);
        $responseListData = $responseListData->sortByDesc('date_time')->values();
        $responseListData = $this->paginate($responseListData, $perPage = 5, $page
= null, $options = ['path' => $request->url()]);
        $ratings = auth()->user()->ratings()
```

```
            ->whereNull('blocked_at')
            ->where('client_email', '<>', 'example@mail.com') // default rating
            ->latest()
            ->paginate(10);
        return view('site.index', compact(['location', 'title',
'responseListData', 'js', 'ratings']));
    }
}
```

ChangePasswordController.php

```
namespace App\Http\Controllers;

use App\Models\Page;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Validation\Rule;

class ChangePasswordController extends Controller
{
    public function messages(){
        return [
            'password.current_password' => __('passwords.current_password'),
            'newpassword.confirmed' => __('validation.password_confirmation'),
        ];
    }

    public function rules(){
        return [
            'email' => ['required', Rule::unique('users')->ignore(auth()-
>id()),'max:255','email'],
            'password' => 'nullable|min:8',
            'newpassword' => 'nullable|confirmed|min:8',
        ];
    }

    public function index($locale, $location = "cabinet.settings.index", Request
$request)
    {
        $title = Page::getTitle($location);
        return view('site.index', compact(['location', 'title']));
    }

    public function store(Request $request){
        $validator = Validator::make($request->all(), $this->rules(), $this-
>messages());

        if ($validator->fails()) {
            return back()
                ->withErrors($validator)
                ->withInput();
        }
        /** @var App\Models\User $user */
        $user = auth()->user();
        if(!empty($request->newpassword) && !is_null($request->newpassword)){
            $user->password = Hash::make($request->newpassword);
        }
        $user->email = $request->email;
        $user->save();
```

```
        return redirect()->route('cabinet.index', app()->getLocale())-
>with('notification', [
            'type' => 'success',
            'title' => __('modals.result_success'),
            'content' => __('modals.password_saved')
        ]);
    }
}
```

DossierController.php

```
namespace App\Http\Controllers;

use App\Models\City;
use App\Models\Country;
use App\Models\Currency;
use App\Models\Dossier;
use App\Models\Experience;
use App\Models\FormHelper;
use App\Models\General;
use App\Models\Languages;
use App\Models\Page;
use App\Models\Region;
use App\Models\Spec;
use App\Models\Specialization;
use http\Client\Response;
use Illuminate\Support\Str;
use Illuminate\Validation\Rule;
use SimpleSoftwareIO\QrCode;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;
use function GuzzleHttp\Promise\all;

class DossierController extends Controller
{

    public function index($locale, $location = "site.cabinet.dossier-index",
Request $request, $js = ['js/dossier.js'])
    {


        $title = Page::getTitle($location);
        $dossier = auth()->user()->dossiers()->get();
        if(auth()->user()->getDossiersCount() == 0 ) {

            return redirect()->route('cabinet.dossier.create' , app()->getLocale()
);
        }
        return view('site.index', compact(['location', 'title', 'dossier',
'js']));
    }

    public function create($locale, $location = "site.cabinet.dossier-form",
Request $request, $js = ['js/dossier.js'])
    {
        $title = Page::getTitle($location);
        $countries = Country::where('active', 1)->get();
        $specializations = Specialization::all();
        $dossier = new Dossier();
```

```
        $formHelper = new FormHelper();
        $currencies = Currency::getAllAndPrepare();
        $languages = Languages::where('locale','=',app()->getLocale())->get();
        $existing_photo = [];
        $existing_region = null;
        $existing_city = null;
        $existing_phone = null;
        $existing_license = null;
         if ($any_dossier = auth()->user()->dossiers()->first()) {
            $existing_photo[] = $any_dossier->general->photo;
            $existing_region = $any_dossier->region_id;
             $existing_city = $any_dossier->city_id;
             $existing_phone = $any_dossier->general->office['phones'];
             $existing_license = $any_dossier->license;
        }
        $lang = is_null(app('request')->input('lang')) ? app()->getLocale() :
app('request')->input('lang');
        if (Dossier::isExistWithLang($lang)) {
            return redirect()->route('cabinet.dossier.edit', ['locale' => app()-
>getLocale(), 'id' => Dossier::getUserDossierByLang($lang)->id, 'lang' =>
Dossier::getUserDossierByLang($lang)->locale])->with('notification', [
                'type' => 'error',
                'title' => __('modals.result_error'),
                'content' => __('modals.lang_exist')
            ]);
        }
        return view('site.index', compact(['location', 'title', 'countries',
'languages','specializations', 'dossier', 'formHelper', 'currencies', 'lang',
'js', 'existing_photo','existing_region','existing_phone','existing_license']));
    }

    public function store(Request $request)
    {

        app()->setLocale(substr($request->post('lang'), -2));
        $request->validate([
            'lang' => 'required|string',
            'general.fullname' => 'required|string',
            'general.office.country' => 'required|int',
            'general.office.region' => 'required|int',
            'general.office.city' => [Rule::requiredIf(function () use ($request)
{
                return Region::citiesOfRegionCount($request-
>input('general.office.region')) > 0;
            })],
            'general.office.address.*' => 'required|string',
            'general.office.phones.*.number' => 'required|string',
            'license.*.country' => 'required|string',
            'license.*.institution' => 'required|string',
            'license.*.year' => 'required|string',
            'license.*.number' => 'required|string',
            'file.license.*.' => 'required',
            'license.*.photo' => 'required',
//            'file.photo' => [
//                'mimes:jpeg,jpg,png,gif',
//                'max:10000',
//                [Rule::requiredIf(function () use ($request) { // if theres no
photo from existing dossier
//                    return !$request->existing_photo;
//                })],
//            ],
```

```
        ]);
        $data = $request->except('_token');
        if (!isset($data['redirect'])) {

            $request->validate([
                'agreement' => 'accepted',
            ]);

        }
        $dossier = new Dossier();
        $dossier->locale = substr($data['lang'], -2);
        if (is_int(array_search($dossier->locale, Dossier::getExistLanguages())))
{
            if (is_int(array_search($dossier->locale,
Dossier::getExistLanguages()))) {
                return redirect()->route('cabinet.dossier.index', app()-
>getLocale())->with('notification', [
                    'type' => 'error',
                    'title' => __('modals.result_error'),
                    'content' => __('modals.lang_exist')
                ]);
            }
        }

        request()->has('file.photo') ? $data['general']['photo'] =
Str::replaceFirst('public/', '', request()->file('file.photo')->store('avatars'))
: '';
        $general = new General([
            'fullname' => $data['general']['fullname'],
            'photo' => $data['general']['photo'] ?? Str::replaceFirst('public/',
'', $request->existing_photo) ?? '',
            'office' => $data['general']['office'],
        ]);
        $general->save();


        $dossier->license = $data['license'];
        $dossier->overview = Dossier::checkForNullable($data, 'overview');
        $dossier->contacts = Dossier::checkForNullable($data, 'contacts');
        $dossier->education = Dossier::checkForNullable($data, 'education');
        $dossier->experience_other = Dossier::checkForNullable($data,
'experience_other');
        $dossier->training = Dossier::checkForNullable($data, 'training');
        $dossier->publications = Dossier::checkForNullable($data, 'publications');
        $dossier->recommendations = Dossier::checkForNullable($data,
'recommendations');
        $dossier->honors = Dossier::checkForNullable($data, 'honors');
        $dossier->services = Dossier::checkForNullable($data, 'services');
        $dossier->user_id = auth()->user()->id;
        if (isset($data['general']['office']['city']) &&
$data['general']['office']['city'] !== "") {
            $dossier->city_id = $data['general']['office']['city'];
            $dossier->region_id = $data['general']['office']['region'];
        } else {
            $dossier->region_id = $data['general']['office']['region'];
        }
        $notification = [
            'type' => 'success',
            'title' => __('modals.result_success'),
            'content' => __('modals.dossier_saved')
```

```
        ];
        $new = false;
        if (isset($data['status'])) {
            $new = true;
            if ($data['status'] != Dossier::STATUS_NEW) {
                abort(400);
            }
            $notification = [
                'type' => 'success',
                'title' => __('modals.result_success'),
                'content' => __('modals.dossier_created')
            ];
            $dossier->status = $data['status'];
        }
        $dossier->save();
        $dossier->number = '0000' . (1000 + $dossier->id);
        $dossier->save();
        $data['experience'] = Dossier::checkForNullable($data, 'experience');
        if (isset($data['experience'])) {
            if (isset($data['experience']['specs'])) {
                $specData = $data['experience']['specs'];
                unset($data['experience']['specs']);
            }
            $experience = new Experience($data['experience']);
            $experience->save();
            $dossier->experience()->save($experience);
            if (isset($specData)) {
                $spec = new Spec($specData);
                $spec->save();
                $dossier->spec()->save($spec);
            }
        }

        $url = config('app.url') . route('dossier.single', ['locale' => app()->getLocale(), 'dossier' => $dossier->id], false);
        $path = "qr_codes/$dossier->id.png";
        \QrCode::size(500)->format('png')->generate($url, storage_path("app/public/$path"));
        $dossier->qr = $path;
        $dossier->save();
        $dossier->general()->save($general);
        if (isset($data['redirect'])) {
            return redirect()->route('cabinet.dossier.edit', ['locale' => app()->getLocale(), 'dossier' => $dossier->id])
                ->with('url_review', route('dossier.single', ['locale' => app()->getLocale(), 'dossier' => $dossier->id]));
        }
        if(auth()->user()->getDossiersCount() == 1){
            return redirect()->route($new ? 'cabinet.access' : 'cabinet.dossier.index', app()->getLocale())->with('notification', $notification);
        }else{
            return redirect()->route( 'cabinet.dossier.index', app()->getLocale())->with('notification', $notification);

        }

    }

    public function edit($locale, $id, $location = "site.cabinet.dossier-form", Request $request, $js = ['js/dossier.js'])
    {
```

```
        $title = Page::getTitle($location);
        if (auth()->user()->dossiers()->get()->contains('id', $id)) {
            $dossier = Dossier::findOrFail($id);
        } else {
            abort(401);
        }
        $old_lang = $dossier->overview['languages'] ?? null;
       // dd($dossier->license);
        $countries = Country::where('active', 1)->get();
        $specializations = Specialization::all();
        $formHelper = new FormHelper();
        $currencies = Currency::getAllAndPrepare();
        $languages = Languages::where('locale','=',app()->getLocale())->get();
        $lang = is_null(app('request')->input('lang')) ? app()->getLocale() :
app('request')->input('lang');
        if (!Dossier::isExistWithLang($lang)) {
            return redirect()->route('cabinet.dossier.create', ['locale' => app()-
>getLocale(), 'lang' => $lang])->with('notification', [
                'type' => 'error',
                'title' => __('modals.result_error'),
                'content' => __('modals.lang_not_exist')
            ]);
        }

        return view('site.index', compact(['location', 'title', 'countries',
'languages', 'old_lang', 'specializations', 'dossier', 'formHelper', 'currencies',
'lang', 'js']));
    }

    public function update(Request $request, $locale, $id)
    {
        app()->setLocale(substr($request->post('lang'), -2));
        $request->validate([
            'lang' => 'required|string',
            'general.fullname' => 'required|string',
            'general.office.country' => 'required|int',
            'general.office.region' => 'required|int',
            'general.office.city' => [Rule::requiredIf(function () use ($request)
{
                return Region::citiesOfRegionCount($request-
>input('general.office.region')) > 0;
            })],
            'general.office.address.*' => 'required|string',
            'general.office.phones.*.number' => 'required|string',
            'license.*.country' => 'required|string',
            'license.*.institution' => 'required|string',
            'license.*.year' => 'required|string',
            'license.*.number' => 'required|string',
            'license.*.photo' => 'required',
            'file.photo' => 'mimes:jpeg,jpg,png,gif|nullable|max:10000',

        ]);


        $data = $request->except('_token');
        if (!isset($data['redirect'])) {

            $request->validate([
                'agreement' => 'accepted',
            ]);
```

```
        }
        $dossier = Dossier::findOrFail($id);


        $new = false;
        if (isset($data['status'])) {
            $new = true;
        }


        $dossier->locale = substr($data['lang'], -2);

        $general = General::find([$dossier->general->id])->first();
        request()->has('file.photo') ? $data['general']['photo'] =
Str::replaceFirst('public/', '', request()->file('file.photo')->store('avatars'))
: '';
        $general->update($data['general']);
        $general->save();

        $dossier->license = $data['license'];

        $dossier->overview = Dossier::checkForNullable($data, 'overview');
        $dossier->contacts = Dossier::checkForNullable($data, 'contacts');
        $dossier->education = Dossier::checkForNullable($data, 'education');
        $dossier->experience_other = Dossier::checkForNullable($data,
'experience_other');
        $dossier->training = Dossier::checkForNullable($data, 'training');
        $dossier->publications = Dossier::checkForNullable($data, 'publications');
        $dossier->recommendations = Dossier::checkForNullable($data,
'recommendations');
        $dossier->honors = Dossier::checkForNullable($data, 'honors');
        $dossier->services = Dossier::checkForNullable($data, 'services');
        $dossier->admin_seen_at = null;
        if (isset($data['general']['office']['city']) &&
$data['general']['office']['city'] !== "") {
            $dossier->city_id = $data['general']['office']['city'];
            $dossier->region_id = $data['general']['office']['region'];
        } else {
            $dossier->region_id = $data['general']['office']['region'];
        }
        $dossier->status = Dossier::STATUS_SAVED;
        $notification = [
            'type' => 'success',
            'title' => __('modals.result_success'),
            'content' => __('modals.dossier_updated')
        ];
        if (isset($data['status'])) {
            if ($data['status'] != Dossier::STATUS_NEW) {
                abort(400);
            }
            $notification = [
                'type' => 'success',
                'title' => __('modals.result_success'),
                'content' => __('modals.dossier_status_updated')
            ];
            $dossier->status = $data['status'];
        }

        $data['experience'] = Dossier::checkForNullable($data, 'experience');
        if ((!is_null($dossier->experince) || !is_null($data['experience'])) &&
isset($data['experience']['specs'])) {
```

```
            $specData = $data['experience']['specs'];
            unset($data['experience']['specs']);
        }
        if (!is_null($dossier->experience)) {
            Experience::destroy($dossier->experience->id);
            $experience = new Experience($data['experience']);
            $experience->save();
            $dossier->experience()->save($experience);
        } elseif(!is_null($data['experience'])) {
            $experience = new Experience($data['experience']);
            $experience->save();
            $dossier->experience()->save($experience);
        }

        if (!is_null($dossier->spec)) {
            Spec::destroy($dossier->spec->id);
            $spec = new Spec($specData);
            $spec->save();
            $dossier->spec()->save($spec);
        } elseif(isset($specData)) {
            $spec = new Spec($specData);
            $spec->save();
            $dossier->spec()->save($spec);
        }
        $dossier->save();
        $dossier->general()->save($general);

        if (isset($data['redirect'])) {
            return redirect()->route( 'cabinet.dossier.edit', ['locale' => app()-
>getLocale(), 'dossier' => $dossier->id])->with('url_review',
route('dossier.single', ['locale' => app()->getLocale(), 'dossier' => $dossier-
>id]));
        }
        return redirect()->route($new ? 'cabinet.access' :
'cabinet.dossier.index', app()->getLocale())->with('notification', $notification);
    }

    public function destroy(Request $request, $locale, $id)
    {
        if (auth()->user()->dossiers()->get()->contains('id', $id)) {
            Dossier::destroy($id);
            return back()->with('notification', [
                'type' => 'success',
                'title' => __('modals.result_success'),
                'content' => __('modals.dossier_deleted')
            ]);
        }
        abort(401);
    }

    public function activate(Request $request, $locale, $id)
    {
        if (auth()->user()->dossiers()->get()->contains('id', $id)) {
            $dossier = Dossier::find([$id])->first();
            if ($dossier->status != Dossier::STATUS_NOT_ACTIVE) {
                abort(401);
            }
            $dossier->status = Dossier::STATUS_ACTIVE;
            $dossier->save();
            return back()->with('notification', [
                'type' => 'success',
```

```
                'title' => __('modals.result_success'),
                'content' => __('modals.dossier_status_updated')
            ]);
        }
        abort(401);
    }

    public function stop(Request $request, $locale, $id)
    {
        if (auth()->user()->dossiers()->get()->contains('id', $id)) {
            $dossier = Dossier::find([$id])->first();
            if ($dossier->status != Dossier::STATUS_ACTIVE) {
                abort(401);
            }
            $dossier->status = Dossier::STATUS_NOT_ACTIVE;
            $dossier->save();
            return back()->with('notification', [
                'type' => 'success',
                'title' => __('modals.result_success'),
                'content' => __('modals.dossier_status_updated')
            ]);
        }
        abort(401);
    }

    public function addToBookmark(Request $request, $id)
    {
        $dossier = Dossier::find($id);
        $dossier->bookmarks_counter = $dossier->bookmarks_counter + 1;
        $dossier->update();


        if ($request->session()->exists('bookmark')) {
            if (!array_search($id, session('bookmark'))) {
                $request->session()->push('bookmark', $id);
                $request->session()->save();
                return response()->json([
                    'success' => true,
                    'count' => count(session('bookmark')),
                    'amount' => $dossier->bookmarks_counter
                ]);
            }
        }
        $request->session()->push('bookmark', $id);
        $request->session()->save();

        return response()->json([
            'success' => true,
            'count' => count(session('bookmark')),
            'amount' => $dossier->bookmarks_counter
        ]);




    }

    public function deleteFromBookmark(Request $request, $id)
    {
```

```
        $dossier = Dossier::find($id);
        $dossier->bookmarks_counter = $dossier->bookmarks_counter - 1;
        $dossier->update();

        if ($request->session()->exists('bookmark')) {
            $sessionDossierKey = array_search($id, session('bookmark'));
            $request->session()->forget('bookmark.' . $sessionDossierKey);
            $request->session()->save();
            return response()->json([
                'success' => true,
                'count' => count(session('bookmark')),
                'amount' => $dossier->bookmarks_counter
            ], 200);
        }
    }
}
```

FeedbackController.php

```
namespace App\Http\Controllers;

use App\Models\Feedback;
use Illuminate\Http\Request;

class FeedbackController extends Controller
{
    public function store(Request $request)
    {
        $request->validate([
            'email' => 'required|email',
            'name' => 'max:100',
            'message' => 'required|max:5000',
            'place_of_living' => 'max:1000',
            'contact_number' => 'max:1000',
        ]);
        $feedback = new Feedback();
        $feedback->name = $request->post('name');
        $feedback->email = $request->post('email');
        $feedback->message = $request->post('message');
        $feedback->place_of_living = $request->post('place_of_living');
        $feedback->contact_number = $request->post('contact_number');
        $feedback->save();

        return back()->with('notification', [
            'type' => 'success',
            'title' => __('modals.feedback_title'),
            'content' => ''
        ]);
    }
}
```

RatingController.php

```
namespace App\Http\Controllers;

use App\Models\Application;
use App\Events\RatingView;
use App\Models\Page;
use Illuminate\Http\Request;
use App\Models\Rating;
```

```
use Illuminate\Support\Facades\Cookie;

class RatingController extends Controller
{
    public function index($locale, $location = "cabinet.rating.index", Request
$request)
    {
        $title = Page::getTitle($location);
        $ratings = auth()->user()->ratings()
            ->whereNull('blocked_at')
            ->where('client_email', '<>', 'example@mail.com') // default rating
            ->latest()
            ->paginate(10);
        return view('site.index', compact(['location', 'title', 'ratings']));
    }

    public function show($locale, Rating $rating, $location =
"cabinet.rating.single", Request $request)
    {
        $title = Page::getTitle($location);
        if($rating->user_id !== auth()->id()){
            abort(401);
        }
        event(new RatingView($rating));
        return view('site.index', compact(['location', 'title', 'rating']));
    }



    public function store(Request $request){
        $this->validate($request, [
            'user' => 'required',
            'rating' => 'required|min:1|max:5',
            'text' => 'required'
        ]);

        $application = Application::findOrFail(Cookie::get('application_id'));

        if (Rating::where('user_id', $request->input('user'))-
>where('client_email', $application->email)->first()) {
            return response()->json([
                'success' => false
            ]);
        }

        $rating_old = Rating::where('user_id', '=', $request->input('user'))-
>sum('stars');
       // $sum_rating = round((($request->input('rating') +
$rating_old)/($rating_old + 5)) * 1.5)+3.5, 2);
       // Rating::where('user_id', '=', $request->input('user'))-
>update(['sum_rating' => $sum_rating]);

        $rating = new Rating([
            'user_id' => $request->input('user'),
            'stars' => $request->input('rating'),
            'comment' => $request->input('text'),
            'client_email' => $application->email,
           // 'sum_rating' => $sum_rating
        ]);
        return response()->json([
            'success' => $rating->save()
```

```
        ]);
    }
}
```

TrackerController.php

```
namespace App\Http\Controllers;

use App\Models\Application;
use App\Models\Page;
use App\Models\Country;
use App\Models\Dossier;
use App\Models\Message;
use App\Models\Rating;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Pagination\LengthAwarePaginator;
use Illuminate\Pagination\Paginator;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Cookie;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;

class TrackerController extends Controller
{
    public function index($locale, $location = "site.tracker.index", Request
$request, $js = ['js/tracker.js'])
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        return view('site.index', compact(['location', 'title', 'js']));
    }

    public function remindCode($locale, $location = "site.tracker.remind-code",
Request $request, $js = ['js/tracker.js'])
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
        return view('site.index', compact(['location', 'title', 'js']));
    }

    public function application($locale, $location = "site.tracker.response-list",
Request $request, $js = ['js/tracker.js'])
    {
        $title = Page::getTitle($location);
        $description = Page::getDescription($location);
//        $application = $request->session()->has('application_id') ?
Application::find($request->session()->get('application_id')) :
Application::where('code', '=', $request->code)->first();

       if(isset($request->code) && !is_null($request->code)){
           $application =  Application::where('code', '=', $request->code)-
>first();
       }elseif (Cookie::get('application_id')){
           $application = Application::find(Cookie::get('application_id'));
       }

        $chats = null;
       $user_id = 0;
        $responseListData = [];
```

```
        if ($application->chats()->first() != null) {
            $chats = $application->chats()->get();
            foreach ($chats as $chat) {
                $dossier = User::find([$chat->user_id])->first()->dossiers()->first();
                $dossier_general = $dossier->general;
                if ( Rating::where('user_id', $chat->user_id)->where('client_email', $application->email)->first() != null){
                    $rating = Rating::where('user_id', $chat->user_id)->where('client_email', $application->email)->first()->sum_rating;
                 }elseif(!is_null(Rating::where('user_id', $chat->user_id)->where('client_email', 'example@mail.com')->first())){
                    $rating =  Rating::where('user_id', $chat->user_id)->where('client_email', 'example@mail.com')->first()->sum_rating;
                }else{
                    $rating = 5;
                }
                Rating::where('user_id', $chat->user_id)->where('client_email', $application->email)->first();
                $responseListData[] = [
                    'chat' => $chat,
                    'api_token' => $chat->application->user->api_token,
                    'dossier' => $dossier_general,
                    'ratingExists' => Rating::where('user_id', $chat->user_id)->where('client_email', $application->email)->count(),
                    'rating' => $rating,
                    'ratingCount' => $dossier->ratingCount,
                    'city' => $dossier->city ? $dossier->city->name : null,
                    'region' => $dossier->region->name,
                    'country' => $dossier->region->country->name,
                    'dossier_url' => route('dossier.single', ['locale' => app()->getLocale(), 'dossier' => $dossier]),
                    'lastMessage' => $chat->messages()->get()->last()->message ?? '',
                    'application' => Application::find($chat->application_id),
                ];

                $user_id = $chat->user_id;
            }
        }

        $responseListData = collect($responseListData);
        $responseListData = $this->paginate($responseListData, $perPage = 5, $page = null, $options = ['path' => $request->url()]);
        return view('site.index', compact(['location', 'title', 'application', 'chats', 'responseListData', 'js','user_id']));
    }
}
```

Клієнтська частина

DossiersComponent.vue

```
<template>
    <div v-cloak>
        <div class="row justify-content-center mb-5">
            <div v-show="!this.category_region_id" class="col-12 col-md-6 col-xl-3 mb-3">
                <select id="country-select-cabinet-general" v-model="countries_id" :data-locale="this.local"  class="dossier-menu-select pl-1 custom-select">
```

```
                    <option value="" disabled>{{
$t('forms.placeholder_choose_country') }}</option>
<!--                    <option value="1" selected>{{ $t('countries.ukraine')
}}</option>-->
                    <option :value="country.id" v-for="country in countries">{{
$t('countries.' + country.name) }}</option>
                </select>
            </div>
            <div v-show="!this.category_region_id" class="col-12 col-md-6 col-xl-3
mb-3">
                <select id="region-select-cabinet-general" :data-
locale="this.local" disabled class="dossier-menu-select pl-1">
                    <option value=""  selected>{{
$t('forms.placeholder_choose_region') }}</option>
                </select>
            </div>
            <div class="col-12 col-md-6 col-xl-3 mb-3">
                <div class="dropdown" id="myDropdown">
                    <button class="w-100 dropdown-placeholder text-left"
type="button" id="dropdownMenuButton" data-flip="false"
                            data-toggle="dropdown">
                       {{ $t('site.dossier.adve_search') }}
                    </button>
                    <div id="dropdown-menu" class="dropdown-menu dropdown-menu-
search" aria-labelledby="dropdownMenuButton">
                        <div class="dropdown-menu-search-container">
                            <div class="mt-3">
                                {{ $t('site.dossier.find_law') }}
                            </div>
                            <input class="dropdown-menu-input pl-1"
:placeholder="$t('site.dossier.fullname_placeholder')" type="text" v-
model.trim="q">

                            <div class="mt-3">
                                {{ $t('cabinet.form.experience_activities') }}
                            </div>
                            <label class="checkbox-container checkbox-container-
dossier display-aling-center" >
                                {{ $t('cabinet.form.consultations') }}
                                <input class="checkbox" type="checkbox" v-
model="activities" value="consultations">
                                <span class="checkmark checkmark-dossier"></span>
                            </label>
                            <label class="checkbox-container checkbox-container-
dossier display-aling-center">
                                {{ $t('cabinet.form.accompaniment') }}
                                <input class="checkbox" type="checkbox" v-
model="activities" value="accompaniment">
                                <span class="checkmark checkmark-dossier"></span>
                            </label>
                            <label class="checkbox-container checkbox-container-
dossier display-aling-center">
                                {{ $t('cabinet.form.preparation_doc') }}
                                <input class="checkbox" type="checkbox" v-
model="activities"  value="preparation_doc">
                                <span class="checkmark checkmark-dossier"></span>
                            </label>
                            <label class="checkbox-container checkbox-container-
dossier display-aling-center">
                                {{ $t('cabinet.form.representation') }}
                                <input class="checkbox" type="checkbox" v-
```

```
model="activities" value="representation">
                                <span class="checkmark checkmark-dossier"></span>
                            </label>
                            <div v-show="!this.category_specialization_name"
class="mt-3">
                                {{ $t('cabinet.form.experience_specializations')
}}
                            </div>

                            <select v-bind:class="{ 'select2-hidden':
this.category_specialization_name }" id="select-specialization" class="w-100
dropdown-menu-select mt-2 pl-1 custom-select" >
                                <option value selected >{{
$t('forms.placeholder_choose') }}</option>
                                <option :value="specialization.name" v-
for="specialization in specializations">{{ $t('specializations.' +
specialization.name) }}</option>
                            </select>
                            <!--                                <div>-->
<!--                                    Місто-->
<!--                                </div>-->
<!--                                <select class="w-100 dropdown-menu-select mt-2 pl-
1" name="" id="">-->
<!--
                            </select>-->
                            <div v-show="!this.degree_name" class="mt-3">
                                {{ $t('cabinet.form.education_degree') }}
                            </div>
                            <select v-bind:class="{ 'select2-hidden':
this.degree_name }" id="select-degree" class="w-100 dropdown-menu-select mt-2 pl-1
custom-select" >
                                <option value selected >{{
$t('forms.placeholder_choose') }}</option>
                                <option :value="degree" v-for="degree in
degrees">{{degree }}</option>
                            </select>
<!--                                <div v-show="!this.language_name" class="mt-3">-->
<!--                                    {{ $t('cabinet.form.lawyer_language') }}-->
<!--                                </div>-->
<!--                                <select v-bind:class="{ 'select2-hidden':
this.language_name }" id="select-language" class="w-100 dropdown-menu-select mt-2
pl-1 custom-select" >-->
<!--                                    <option value selected >{{
$t('forms.placeholder_choose') }}</option>-->
<!--                                    <option :value="language.id" v-for="language
in languages">{{language.name }}</option>-->
<!--                                </select>-->
                            <div v-show="!this.field_name" class="mt-3">
                                {{ $t('cabinet.form.education_field') }}
                            </div>
                            <select v-bind:class="{ 'select2-hidden':
this.field_name }" id="select-field" class="w-100 dropdown-menu-select mt-2 pl-1
custom-select" >
                                <option value selected >{{
$t('forms.placeholder_choose') }}</option>
                                <option :value="field" v-for="field in fields">{{
field }}</option>
                            </select>
                            <div v-show="!this.specialty_name" class="mt-3">
                                {{ $t('cabinet.form.education_specialty') }}
                            </div>
```

```
                            <select v-bind:class="{ 'select2-hidden':
this.specialty_name }" id="select-specialty" class="w-100 dropdown-menu-select mt-
2 pl-1 custom-select" >
                                <option value selected >{{
$t('forms.placeholder_choose') }}</option>
                                <option :value="specialty" v-for="specialty in
specialties">{{ specialty }}</option>
                            </select>
                            <div v-show="!this.ac_degree_name" class="mt-3">
                                {{ $t('cabinet.form.academic_degree') }}
                            </div>
                            <select v-bind:class="{ 'select2-hidden':
this.ac_degree_name }" id="select-acdegree" class="w-100 dropdown-menu-select mt-2
pl-1 custom-select" >
                                <option value selected >{{
$t('forms.placeholder_choose') }}</option>
                                <option :value="ac_degree" v-for="ac_degree in
ac_degrees">{{ ac_degree }}</option>
                            </select>
                            <div  class="mt-3">
                                {{ $t('cabinet.form.teaching_experience') }}
                            </div>
                            <select id="select-teaching" class="w-100 dropdown-
menu-select mt-2 pl-1 custom-select" >
                                <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                <option value="yes">{{ $t('cabinet.form.yes')
}}</option>
                                <option value="no">{{ $t('cabinet.form.no')
}}</option>
                            </select>
                            <div  class="mt-3">
                                {{ $t('cabinet.form.appeal__exp') }}
                            </div>
                            <select id="select-app" class="w-100 dropdown-menu-
select mt-2 pl-1 custom-select" >
                                <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                <option value="yes">{{ $t('cabinet.form.yes')
}}</option>
                                <option value="no">{{ $t('cabinet.form.no')
}}</option>
                            </select>
                            <div  class="mt-3">
                                {{ $t('cabinet.form.appeal__cases') }}
                            </div>
                            <select id="select-case" class="w-100 dropdown-menu-
select mt-2 pl-1 custom-select" >
                                <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                <option value="yes">{{ $t('cabinet.form.yes')
}}</option>
                                <option value="no">{{ $t('cabinet.form.no')
}}</option>
                            </select>
                            <div  class="mt-3">
                                {{ $t('cabinet.form.cassation') }}
                            </div>
                            <select id="select-cassation" class="w-100 dropdown-
menu-select mt-2 pl-1 custom-select" >
                                <option value="" selected>{{
```

```
$t('forms.placeholder_choose') }}</option>
                                    <option value="yes">{{ $t('cabinet.form.yes')
}}</option>
                                    <option value="no">{{ $t('cabinet.form.no')
}}</option>
                                </select>
                                <div  class="mt-3">
                                    {{ $t('cabinet.form.patent') }}
                                </div>
                                <select id="select-patent" class="w-100 dropdown-menu-
select mt-2 pl-1 custom-select" >
                                    <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="yes">{{
$t('cabinet.form.available_pat') }}</option>
                                    <option value="no">{{
$t('cabinet.form.noavailable_pat') }}</option>
                                </select>
                                <div  class="mt-3">
                                    {{ $t('cabinet.form.publications') }}
                                </div>
                                <select id="select-publication" class="w-100 dropdown-
menu-select mt-2 pl-1 custom-select" >
                                    <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="yes">{{
$t('cabinet.form.available_pl') }}</option>
                                    <option value="no">{{
$t('cabinet.form.noavailable_pl') }}</option>
                                </select>

                                <div  class="mt-3">
                                    {{ $t('cabinet.form.recommendations') }}
                                </div>
                                <select id="select-recommendation" class="w-100
dropdown-menu-select mt-2 pl-1 custom-select" >
                                    <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="yes">{{
$t('cabinet.form.available_pl') }}</option>
                                    <option value="no">{{
$t('cabinet.form.noavailable_pl') }}</option>
                                </select>

                                <div  class="mt-3">
                                    {{ $t('cabinet.form.honors') }}
                                </div>
                                <select id="select-honor" class="w-100 dropdown-menu-
select mt-2 pl-1 custom-select" >
                                    <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="yes">{{
$t('cabinet.form.available_pl') }}</option>
                                    <option value="no">{{
$t('cabinet.form.noavailable_pl') }}</option>
                                </select>

                                <div class="mt-3">
                                    {{ $t('site.dossier.rating') }}
                                </div>
                                <select data-minimum-results-for-search="Infinity"
```

```
id="select-rating"  class="w-100 dropdown-menu-select  custom-select mt-2 pl-1">
                                    <option value selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="1">{{ $t('site.dossier.from') }}
1</option>
                                    <option value="2">{{ $t('site.dossier.from') }}
2</option>
                                    <option value="3">{{ $t('site.dossier.from') }}
3</option>
                                    <option value="4">{{ $t('site.dossier.from') }}
4</option>
                                    <option value="5">5</option>
                                </select>



                                <div class="mt-3">
                                    {{ $t('cabinet.form.dossier_lang') }}
                                </div>
                                <select data-minimum-results-for-search="Infinity"
id="select-lang" class="w-100 dropdown-menu-select mt-2 pl-1 custom-select"
name="">
                                    <option value>{{ $t('forms.placeholder_choose')
}}</option>
                                    <option v-if="this.local == 'uk'" selected
value="uk">Українська</option>
                                    <option v-else  value="uk">Українська</option>
                                    <option v-if="this.local == 'en'" selected
value="en">English</option>
                                    <option v-else value="en">English</option>
                                </select>
                                <div class="mt-3">
                                    {{ $t('cabinet.form.experience_year') }}
                                </div>
                                <select data-minimum-results-for-search="Infinity"
id="select-years" class="w-100 dropdown-menu-select mt-2 pl-1 mb-3  custom-select"
v-model="years">
                                    <option value="" selected>{{
$t('forms.placeholder_choose') }}</option>
                                    <option value="1_5">{{
$t('site.dossier.sort_year_1_5') }}</option>
                                    <option value="5_10">{{
$t('site.dossier.sort_year_5_10') }}</option>
                                    <option value="10_20">{{
$t('site.dossier.sort_year_10_20') }}</option>
                                    <option value="20_30">{{
$t('site.dossier.sort_year_20_30') }}</option>
                                    <option value="30_40">{{
$t('site.dossier.sort_year_30_40') }}</option>
                                    <option value="40_50">{{
$t('site.dossier.sort_year_40_50') }}</option>
                                </select>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="col-12 col-md-6 col-xl-3 mb-3">
                    <select data-minimum-results-for-search="Infinity" id="select-
sort_by" class="dossier-menu-select pl-1 custom-select">
                        <option value="id_desc" disabled selected>{{
$t('cabinet.request.sort_placeholder') }}</option>
```

```
                    <option value="rating_asc">{{
$t('site.dossier.filter_rating_asc') }}</option>
                    <option value="rating_desc">{{
$t('site.dossier.filter_rating_desc') }}</option>
                    <option value="general.fullname_asc">{{
$t('site.dossier.filter_fullname_asc') }}</option>
                    <option value="general.fullname_desc">{{
$t('site.dossier.filter_fullname_desc') }}</option>
                    <option value="years_desc">{{
$t('site.dossier.filter_years_desc') }}</option>
                    <option value="years_asc">{{
$t('site.dossier.filter_years_asc') }}</option>
                </select>
            </div>
        </div>
        <div v-if="!loaded" style="font-size: 24px;" class="col-12 text-center
about-page-heading mb-3">
            {{ $t('site.dossier.loading') }}
        </div>
        <div v-if="loaded && Object.keys(dossiers).length" class="row justify-
content-center mb-3">
            <div class="col-12 col-md-6 col-lg-4 col-xl-3 mb-4 text-center" v-
for="(dossier, index) in dossiers" :key="dossier.id">
                <div :id="'item-dossier-' + index" class="dossier-item mx-auto w-
100"   :style="{ backgroundColor: (dossier.in_bookmarks === true ||
bookmarks.includes(dossier.id.toString()) || (!dossier.in_bookmarks  &&
bookmarks.includes(dossier.id.toString())) )  ? '#eb9c231f' : '' }">
                    <div class="dossier-seen" v-
if="Array.from(seen).includes(String(dossier.id))" >
                        {{ $t('site.dossier.seen') }}
                    </div>
                    <div class="dossier-single-img-wrapper">
                        <img class="dossier-item-img" :src="'/../storage/' +
dossier.general.photo" alt="">
                    </div>
                    <div class="dossier-item-name">{{ dossier.general.fullname
}}</div>
                    <div class="dossier-item-stats">
                        <button data-toggle="modal" :data-target="`.dossier-
`+dossier.user_id" class="dossier-item-stats-a">
                            {{ dossier.rating }}
                        </button>
                        <div class="dossier-item-stats-b">
                            {{ dossier.seen_counter }}
                        </div>

                    </div>
                    <div class="text-center dossier-item-city">
                        {{ $t('countries.'+dossier.country) }}, {{
$t('regions.'+dossier.region) }}{{ dossier.city !== null ? ",
"+$t('cities.'+dossier.city) : ''}}
                    </div>
                    <div>
                        <div class="dossier-item-address">
                            {{ dossier.general.office.address[0] }}
                        </div>
                    </div>
                    <a :href="`/${local}/dossier/view/${dossier.id}`"
class="dossier-item-link">
                        {{ $t('cabinet.form_index.view') }}
                    </a>
```

```
                    <div v-if="user === false">
                    <button v-if="(!dossier.in_bookmarks  &&
!bookmarks.includes(dossier.id.toString()) )" @click="addToBookmark(index,
dossier.id)" :id="'old-add-button' + index"
                                class="dossier-item-bookmark">
                        {{ $t('site.dossier.to_bookmark') }}
                    </button>
                    <button v-else @click="deleteFromBookmark(index, dossier.id)"
:id="'delete-button-'+ index" class="dossier-item-bookmark-painted">
                        {{ $t('site.dossier.delete_from_bookmark') }}
                    </button>
                        <button :id="'add-button-'+ index"
@click="addToBookmark(index, dossier.id)" style="display: none"
                                class="dossier-item-bookmark">
                            {{ $t('site.dossier.to_bookmark') }}
                        </button>

                    </div>
                </div>



                <div :class="`modal fade bd-example-modal-lg dossier-
`+dossier.user_id" :id=" dossier.user_id" tabindex="-1" role="dialog" aria-
labelledby="myLargeModalLabel" aria-hidden="true">
                    <div class="modal-dialog modal-lg">
                        <div class="modal-content">
                            <div class="modal-header">
                                <h5 class="modal-title" :id=" dossier.user_id"
style="color: black">Відгуки</h5>
                                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                                    <span aria-hidden="true">&times;</span>
                                </button>
                            </div>
                            <div class="modal-body">



                                <div  v-for="rat in ratings" :data-id="dossier.id"
v-if="rat.user_id == dossier.user_id" data-locale="uk" class="col-12 col-xl-9
order-1 order-xl-2 pbt-10 request-main-content">
                                    <div   class="table-user-info mb-2"
style="text-align: left">

<!--                                        <span style="color: black" >{{
rat.stars}}</span>-->
                                    <span v-for="n in rat.stars" style="font-
size:18px;" class="icon">★</span>
<!--                                        <img class="d-inline-block d-sm-none"
style="vertical-align: sub; width: 24px;"
src="https://googllaw.local/img/cabinet/reply.png" alt="">-->
                                    </div>

                                    <div v-if="rat.client_email !==
'example@mail.com'" class="text-modal-rating" >

                                        {{rat.comment}}
                                    </div>
                                </div>
```

```
                            </div>
                        </div>
                    </div>
                </div>


            </div>
            <div class="col-12 text-center" v-
if="Object.keys(this.dossiers).length >= dossiersToShow">
                <button class="dossier-more" @click="dossiersToShow += 16; loaded
= !loaded">
                    {{ $t('site.dossier.show_more') }}
                </button>
            </div>



        </div>
        <div v-else-if="loaded" style="font-size: 24px;" class="col-12 text-center
about-page-heading mb-3">
            {{ $t('site.dossier.empty_results') }}
        </div>
    </div>

</template>

<script>
    export default {
        props: ['data'],
        data() {
            return {
                dossiers: [],
                bookmarks: [],
                ratings: this.data.ratings,
                loaded: false,
                countries: [],
                regions: [],
                specializations: [],
                dossiersToShow: 16,
                rating: '',
                timeoutID: null,
                q: '',
                user: this.data.user,
                cost_from: '',
                cost_to: '',
                cost: this.data.cost,
                activities: [],
                years: '',
                specialization: '',
                sort_by: 'id_desc',
                regions_id: '',
                category_region_id: '',
                category_specialization_name: '',
                degree_name: '',
                field_name: '',
                ac_degree_name: '',
```

```
                specialty_name: '',
                language_name: '',
                country: '',
                city: '',
                countries_id: '',
                local: '',
                seen: this.data.seen,
                degrees: [],
                fields: [],
                specialties: [],
                languages: [],
                ac_degrees: []

            }
        },
        methods: {
            deleteFromBookmark(index, id) {
                axios.post(`/dossier/${id}/bookmark/delete`)
                    .then(response => {
                        let header = document.getElementById('header-bookmark-
toggle');
                        if(response.data.count > 0){
                            header.classList.add('active');
                            if(document.getElementById("bookmark-count") ===
null){
                                let newSpan = document.createElement("span");
                                newSpan.className = 'bookmark-count';
                                newSpan.id = 'bookmark-count';
                                newSpan.innerHTML = response.data.count
                                document.getElementById("mark-
img").after(newSpan);

                            }else{
                                document.getElementById("bookmark-
count").innerHTML    = response.data.count;
                            }

                            //document.getElementById("amount-" + id).innerHTML =
null;

                        }
                        else {
                            header.classList.remove('active');
                            document.getElementById("bookmark-count").remove();
                        }


                        this.$set(this.dossiers[index], 'in_bookmarks', false);
                       //  document.getElementById('old-add-button-' +
index).style.display = 'none';

                        if(!this.dossiers[index].in_bookmarks  &&
this.bookmarks.includes(id.toString()) ) {
                            document.getElementById('item-dossier-' +
index).style.backgroundColor = 'white';
                            document.getElementById('delete-button-' +
index).style.display = 'none';
                            document.getElementById('add-button-' +
index).style.display = 'inline-block';
                        }
```

```
                        //document.getElementById('delete-button-' +
index).style.display = 'none';
                    })
            },
            addToBookmark(index, id) {
                let dossier = this.dossiers[index];
                axios.post(`/dossier/${id}/bookmark/add`)
                    .then(response => {
                        let header = document.getElementById('header-bookmark-
toggle');
                        if(response.data.count > 0){
                            header.classList.add('active');

                            if(document.getElementById("bookmark-count") ===
null){
                                let newSpan = document.createElement("span");
                                newSpan.className = 'bookmark-count';
                                newSpan.id = 'bookmark-count';
                                newSpan.innerHTML = response.data.count
                                document.getElementById("mark-
img").after(newSpan);

                            }else{
                                document.getElementById("bookmark-
count").innerHTML    = response.data.count;
                            }
                            //document.getElementById("amount-" + id).innerHTML
= '(' + response.data.amount + ')';
                        }
                        else {
                            header.classList.remove('active');
                        }


                        document.getElementById('add-button-' +
index).style.display = 'none';
                        this.$set(this.dossiers[index], 'in_bookmarks', true);
                        console.log(document.getElementById('delete-button-' +
index));
                        if (document.getElementById('delete-button-' + index) !==
null)
                        {
                            console.log(document.getElementById('delete-button-' +
index).style.display);
                            document.getElementById('delete-button-' +
index).style.display = 'inline-block';}

                    })


            },
            getDossiers(showmore = false) {


                if (!showmore){
                    this.loaded = false;
                    this.dossiers = {};
                }
                axios.post(`/api/dossier`, {
                    q: this.q,
                    activities: this.activities,
```

```
                    cost_from: this.cost_from,
                    cost_to: this.cost_to,
                    years: $('#select-years').children("option:selected").val(),
                    degrees: this.degree_name ? this.degree_name : $('#select-
degree').children("option:selected").val(),
                    fields: this.field_name ? this.field_name : $('#select-
field').children("option:selected").val(),
                    specialties: this.specialty_name ? this.specialty_name :
$('#select-specialty').children("option:selected").val(),
                    languages: this.language_name ? this.language_name :
$('#select-language').children("option:selected").val(),
                    ac_degrees: this.ac_degree_name ? this.ac_degree_name :
$('#select-acdegree').children("option:selected").val(),
                    specialization: this.category_specialization_name ?
this.category_specialization_name : $('#select-
specialization').children("option:selected").val(),
                    rating: $('#select-rating').children("option:selected").val(),
                    teaching: $('#select-
teaching').children("option:selected").val(),
                    appeal:  $('#select-app').children("option:selected").val(),
                    cases:  $('#select-case').children("option:selected").val(),
                    patent:  $('#select-
patent').children("option:selected").val(),
                    publication:  $('#select-
publication').children("option:selected").val(),
                    recommendation:  $('#select-
recommendation').children("option:selected").val(),
                    cassation:  $('#select-
cassation').children("option:selected").val(),
                    honor:  $('#select-honor').children("option:selected").val(),
                    sort_by: $('#select-
sort_by').children("option:selected").val(),
                    locale: $('#select-lang').children("option:selected").val(),
                    regions_id: this.category_region_id ?
this.category_region_id.toString() : $('#region-select-cabinet-
general').children("option:selected").val(),
                    countries_id: $('#country-select-cabinet-
general').children("option:selected").val(),
                    per_page: this.dossiersToShow
                })
                    .then((response) => {
                        this.dossiers = response.data.data;
                        this.loaded = true;
                    })
            }
        },
        watch: {
            dossiersToShow() {
                this.getDossiers( true);
            },
            rating() {
                this.getDossiers();
            },
            activities() {
                this.getDossiers();
            },
            cost_from() {
                this.getDossiers();
            },
            cost_to() {
                this.getDossiers();
```

```
            },
            years() {
                this.getDossiers();
            },
            specialization() {
                this.getDossiers();
            },
            q() {
                clearTimeout(this.timeoutID);
                this.timeoutID = setTimeout(() => {
                    this.getDossiers();
                }, 300)
            },
            sort_by() {
                this.getDossiers();
            },
            countries_id() {
                this.getDossiers();
            }
        },
        created() {
            this.local= window.location.pathname.replace(/^\/([^\/]+).*/i,'$1');
            this.countries = this.data.countries;
            this.specializations = this.data.specializations;
            if(this.data.region_id){
                this.category_region_id = this.data.region_id;
            }
            if(this.data.specialization_name){
                this.specialization = this.data.specialization_name;
            }
            this.degrees = this.data.degrees;
            this.fields = this.data.fields;
            this.cost_from = this.data.cost_from;
            this.specialties = this.data.specialties;
            this.languages = this.data.languages;
            this.ac_degrees= this.data.ac_degrees;
            this.bookmarks =  this.data.bookmarks !== null ?
Object.values(this.data.bookmarks) : [];

        },
        mounted() {
            $('#country-select-cabinet-general').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-rating').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-sort_by').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-lang').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-specialization').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-degree').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-acdegree').on('select2:select', e => {
                this.getDossiers();
```

```
            });
            $('#select-field').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-teaching').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-app').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-case').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-honor').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-cassation').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-language').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-patent').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-publication').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-recommendation').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-specialty').on('select2:select', e => {
                this.getDossiers();
            });
            $('#select-years').on('select2:select', e => {
                this.getDossiers();
            });
            $('#region-select-cabinet-general').on('select2:select', e => {
                this.getDossiers();
            });
            this.getDossiers();
        }
    }
</script>

<style scoped>

</style>
```

MessageListComponent.vue

```
<template>
    <div class="tracker-chat__container mb-4"
         style="box-shadow: none; background: #ffffff;" v-cloak>
        <div class="row ml-3">
            <a :href="localChat"><img style="transform: rotate(180deg);height:
37px;" src="/img/tracker/next.png" alt=""></a>
        </div>
        <div class="cabinet-chat__header">
            <div class="row justify-content-between">
                <div class="col-6">
```

```
                    <span>{{ item.chat.admin ? $t('cabinet.mail.admin') :
item.application.name }}</span>
                </div>
                <div v-if="item.application && item.application.status === 10"
class="d-none d-xl-flex col-6 row text-center cabinet-chat__header-nav">
                    <div class="offset-1 col-6 p-0">
                        <a class="tracker-chat__link" target="_blank"
:href="`request/${this.item.application.id}`">{{ $t('cabinet.mail.show_request')
}}</a>
                    </div>
                    <div class="col-3">
                        <div class="cursor-pointer" @click="accessReport()"
onclick="$('#mail-report-modal').modal('show');" href="javascript:;">{{
$t('site.tracker.response.complaint') }}</div>
                    </div>
                </div>
                <div v-if="item.application && item.application.status === 10"
class="d-block d-xl-none offset-4 offset-md-3 col-2 text-right">
                    <a class="pl-1 dossier-item-cabinet-index-more-actions"
type="button"
                       id="dropdownMenuButton" data-toggle="dropdown" aria-
haspopup="true"
                       aria-expanded="false">
                    </a>
                    <div class="dropdown-menu dropdown-menu-right" aria-
labelledby="dropdownMenuButton">
                        <div class="dropdown-item">
                            <div class="cursor-pointer" @click="accessReport()"
onclick="$('#mail-report-modal').modal('show');" href="javascript:;">{{
$t('site.tracker.response.complaint') }}</div>
                        </div>
                        <a class="tracker-chat__link dropdown-item"
target="_blank" :href="`request/${this.item.application.id}`">{{
$t('cabinet.mail.show_request') }}</a>
                    </div>
                </div>
            </div>
        </div>
        <div class="cabinet-chat__message-list w-100" v-chat-scroll>
            <div style="clear:both;
}" v-for="message in messages" >
                <img v-if="message.owner == 'application' &&
item.application.photo != null" :src="`/storage/${item.application.photo}`" alt=""
class="lawyer-avatar mr-2" style="
    /* margin-right: 24px; */
    display: inline-block;
    margin-top: auto;
    margin-t: 0px;
    margin-left: 10px;
">
                <div v-if="message.owner == 'application' || message.owner ==
'admin'"
                     class="tracker-chat__message-container mt-3">
                    {{ message.message }}
                </div>
                <div v-else style="clear: both;" :id="message.id" class="tracker-
chat__message-mine-container mt-3">
                    {{ message.message }}
                    <span @click="deleteMessage(message.id)" style="float:
right;cursor: pointer;margin-left: 11px;" title="Видалити для всіх">
                        <svg xmlns="http://www.w3.org/2000/svg" width="16"
```

```
height="16" fill="currentColor" class="bi bi-trash" viewBox="0 0 16 16">
  <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm2.5 0a.5.5
0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0 0 1
0V6z"/>
  <path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0
1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1
0 0 1 1 1v1zM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882
4H4.118zM2.5 3V2h11v1h-11z"/>
</svg>
                    </span>
                </div>

            </div>
        </div>
        <div v-if="!item.chat.admin" class="tracker-chat__input-container">
            <div class="h-100">
                <div v-if="item.application.status == 0 || item.application.status
== 10" class="row justify-content-center">
                    <div class="col-12 col-md-9 col-lg-8 col-xl-9">
                                <textarea style="resize: none; background-color:
#f5f5f5; border: none;"
                                          class="tracker-chat__textarea pl-2 pt-2
w-100" :placeholder="$t('site.tracker.response.send_placeholder')"
                                          v-model.trim="newMessage" ></textarea>
                    </div>
                    <div class="col-8 col-md-3 col-lg-4 col-xl-3 my-auto">
                        <a class="p-2 tracker-chat__send-message-button mx-auto
ml-md-auto mr-md-0" @click="sendMessage()">
                            {{ $t('site.tracker.response.send') }}
                        </a>
                    </div>
                </div>
                <div v-else class="row">{{ $t('cabinet.mail.request_notactive')
}}</div>
            </div>
        </div>
    </div>
</template>

<script>
export default {
    props: ['data'],
    data() {
        return {
            messages: [],
            newMessage: '',
            objDiv: [],
            item: [],
            localChat: '',
            api_token: '',
        }
    },
    created() {
        this.api_config = {
            headers: {
                Authorization: `Bearer ` + this.data.api_token,
                Accept: 'application/json',
            }
        };
        this.item = this.data;
        let a = 53;
```

```
        this.fetchMessages();
    },
    mounted() {
        this.objDiv = document.getElementById("listContainer");
        this.localChat = '/' +
window.location.pathname.replace(/^\/([^\/]+).*/i,'$1') + '/cabinet/mail';
    },
    methods: {
        accessRate() {
            $('#rate-lawyer-id').val(this.data.chat.user_id);
        },
        accessComplaint() {
            $('#modal-lawyer-id').val(this.data.chat.user_id)
        },
        fetchMessages() {
            axios.get(`/api/messages/${this.data.chat.id}`, this.api_config)
                .then(response => {
                    this.messages = response.data;
                })
        },
        sendMessage() {
            if(this.newMessage === ''){
                return;
            }
            this.messages.push({
                owner: 'user',
                message: this.newMessage
            })
            axios.post(`/api/message/store`, {
                chat_id: this.item.chat.id,
                message: this.newMessage,
                owner: 'user'
            }, this.api_config)
                .catch(error => {
                    this.messages.pop()
                    alert(this.$t('validation.chat.session_expire'))
                })
                .then(() => {
                this.fetchMessages();
            });
            this.newMessage = ''
        },
        deleteMessage(id){
            axios.post(`/api/messages/delete/${id}`, {

            }, this.api_config);
            document.getElementById(id).style.display = 'none' ;



        },
        accessReport() {
            console.log(this.item.chat.id);
            $('#modal-chat-id').val(this.item.chat.id);
            $('#modal-application-id').val(this.item.application.id)
        }
    }
}
</script>

<style scoped>
```

```
</style>
```

ItemContainerComponent.vue

```
<template>
    <div>
        <div style="margin-bottom: 10px" v-for="item in items">
            <div v-show="showChat" class="row table-row ml-3" v-bind:class="{
active: item.chat_seen !== null }">
                <div class="col-12 col-sm-10 order-1 order-xl-2 pbt-10 mail-main-
content"
                     @click="showChat = !showChat; updateSeenStatus(item);
setChatToShow(item.chat.id); updateMessagesCount(item.new_messages)">
                    <div class="table-user-info mb-2">
                        {{ item.chat.admin ? $t('cabinet.mail.admin') :
item.application.name }}
                    </div>
                    <div class="table-text" >
                        {{ item.lastMessage }}
                    </div>
                </div>
                <div class="col-12 col-sm-2 order-3 order-xl-3 pbt-10 my-auto">
                    <div class="table-mail-info">
                        <div class="cabinet-request-time">
                            {{ item.time }}
                        </div>
                        <div>
                            {{ item.date }}
                        </div>
                    </div>
                </div>
            </div>
            <cabinet-chat-message-list-container :data="item"
                                                 v-if="!showChat && chatToShow ==
item.chat.id"></cabinet-chat-message-list-container>
        </div>
    </div>
</template>

<script>
export default {
    props: ['data'],
    data() {
        return {
            showChat: true,
            objDiv: [],
            chatToShow: '',
            chatId: '',
            items: [],
            api_token: '',
        }
    },
    created() {
        const urlParams = new URLSearchParams(window.location.search);
        const chatParam = urlParams.get('chat');
        if (chatParam !== null) {
            this.showChat = !this.showChat;
            this.chatToShow = chatParam
        }
        this.api_token = this.data.data.api_token;
```

```
            this.items = this.data.data
        },
        methods: {
            setChatToShow(id) {
                this.chatToShow = id
            },
            updateSeenStatus(chat) {
                axios.post(`/api/chat/${chat.chat.id}/viewed`, {}, {
                    headers: {
                        Authorization: `Bearer ` + chat.api_token,
                        Accept: 'application/json',
                    }
                })
            },
            accessReport(id) {
                $('#modal-application-id').val(id)
            },
            updateMessagesCount(count){
                var counter = document.getElementById('messages_counter');
                counter.innerText = counter.innerText - count;

            }
        },
    }
    </script>

    <style scoped>

    </style>
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА
РОЗРОБКА WEB-СИСТЕМИ ДЛЯ ПОШУКУ АДВОКАТІВ ТА ONLINE КОНСУЛЬТАЦІЙ

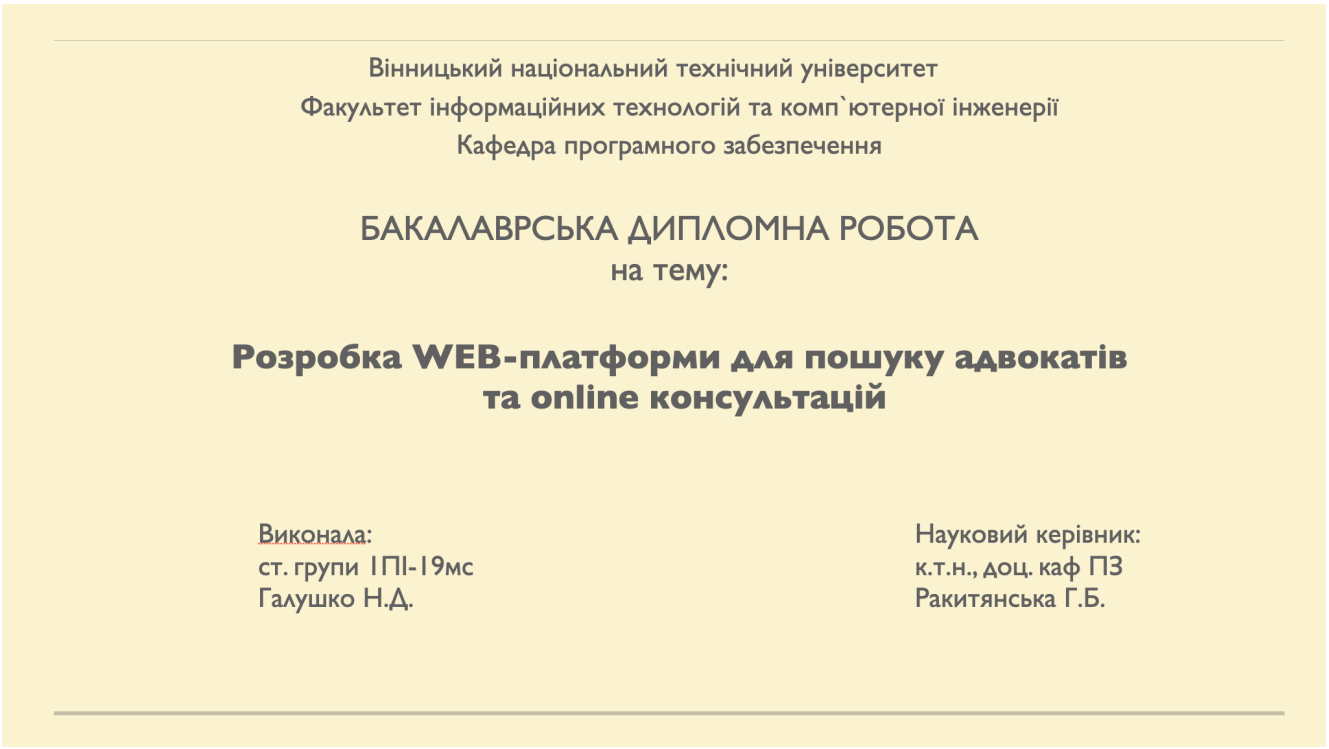

Рисунок Г.1 – Назва роботи

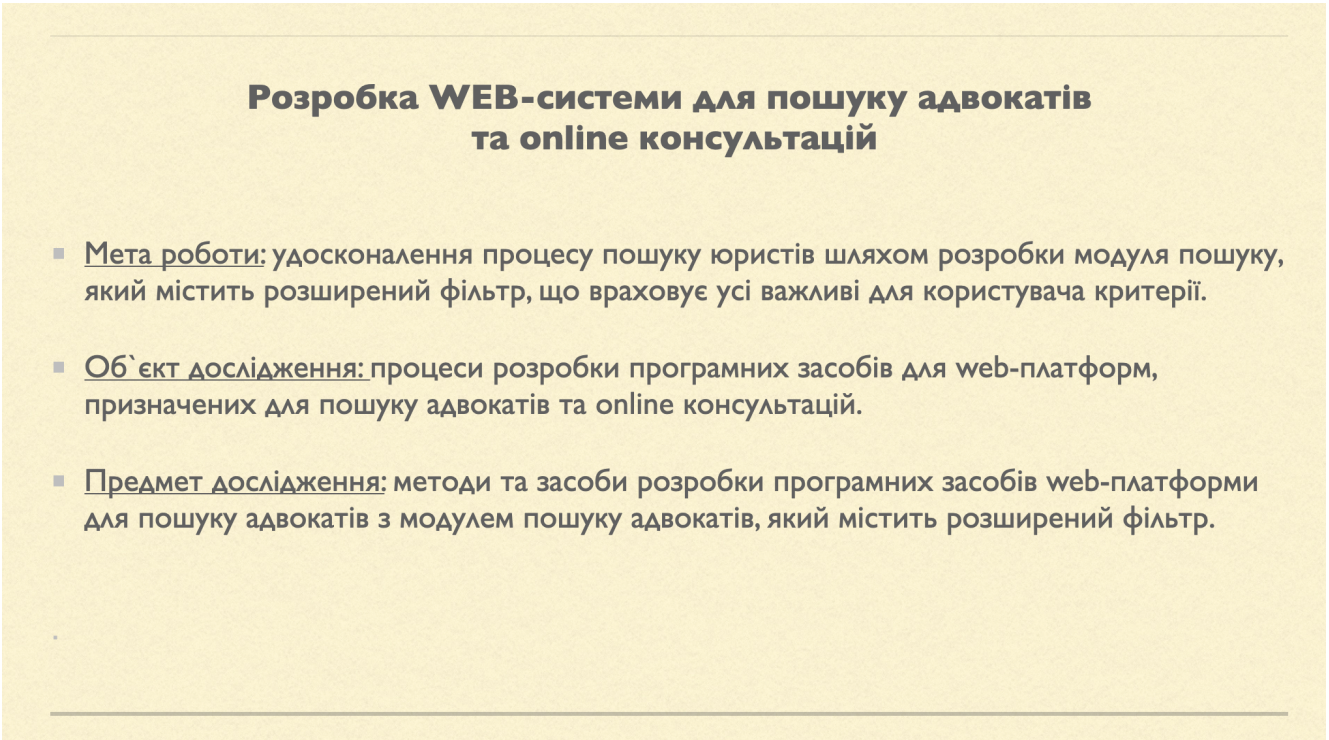

Рисунок Г.2 – Мета роботи, об’єкт та предмет дослідження

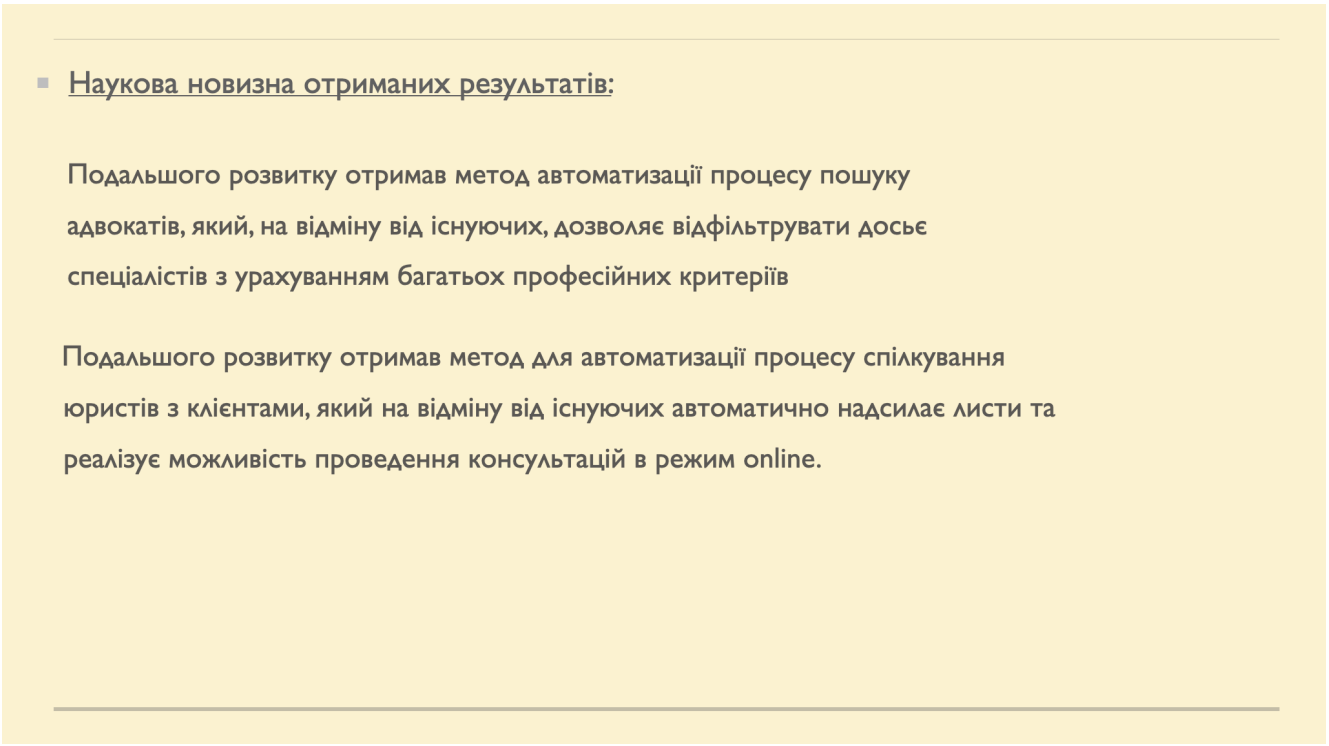

Рисунок Г.3 – Наукова новизна

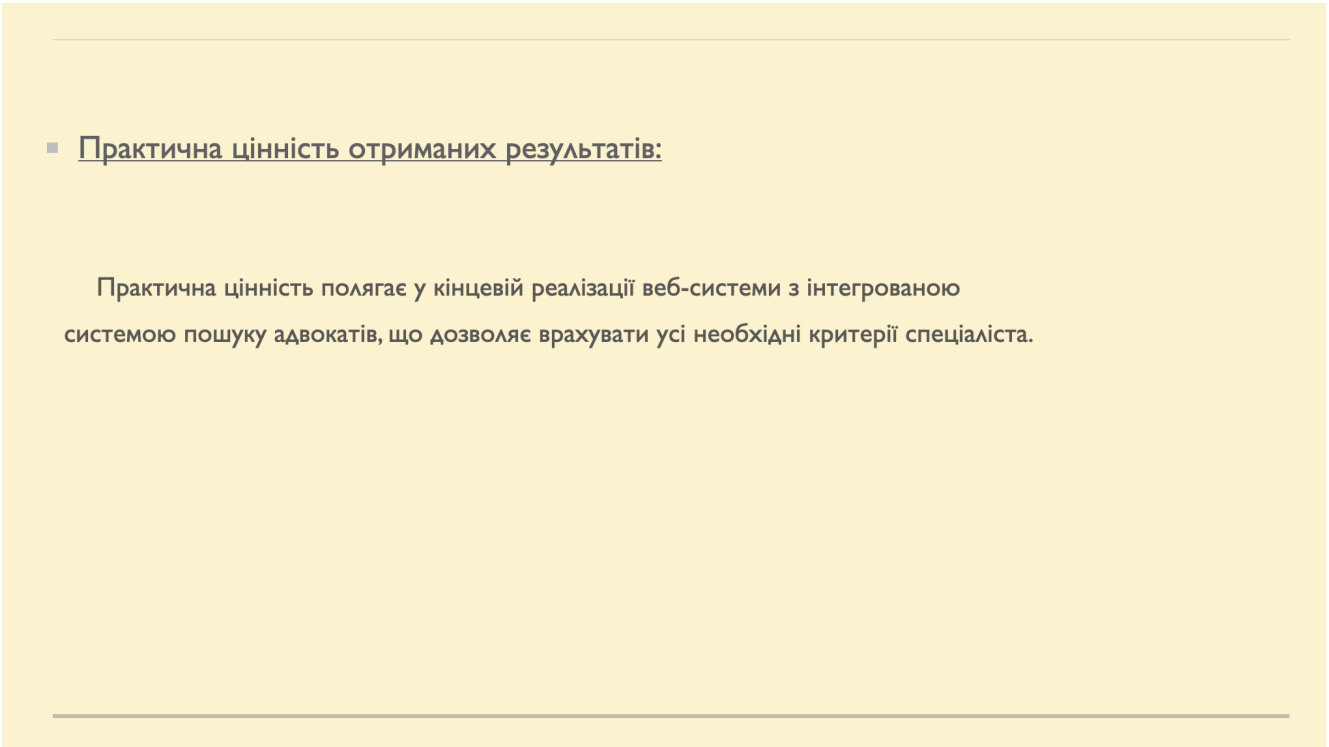

Рисунок Г.4 – Практична цінність роботи

- Порівняльний аналіз аналогів:

Критерії	Legarithm	Legist	Protocol.ua	Lawyer.ua	Власний модуль
Адаптивний інтерфейс	+	+	+	+	+
Можливість розміщувати досьє для усіх бажаючих спеціалістів	-	-	+	+	+
Наявність модуля пошуку юристів	-	-	+	+	+
Наявність розширених фільтрів для пошуку юристів	-	-	-	-	+
Наявність можливості проведення консультацій online	-	+	+	+	+
Загальна оцінка	20 %	40 %	80 %	80 %	100 %

Рисунок Г.5 – Порівняльний аналіз аналогів

- Загальна діаграма роботи web-системи

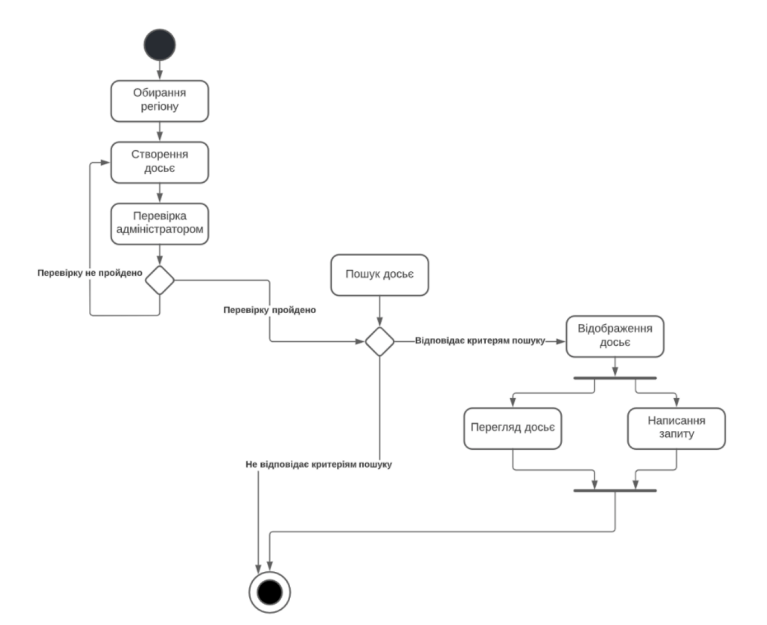

Рисунок Г.6 – Загальна діаграма роботи web-системи

- Головна сторінка web-системи

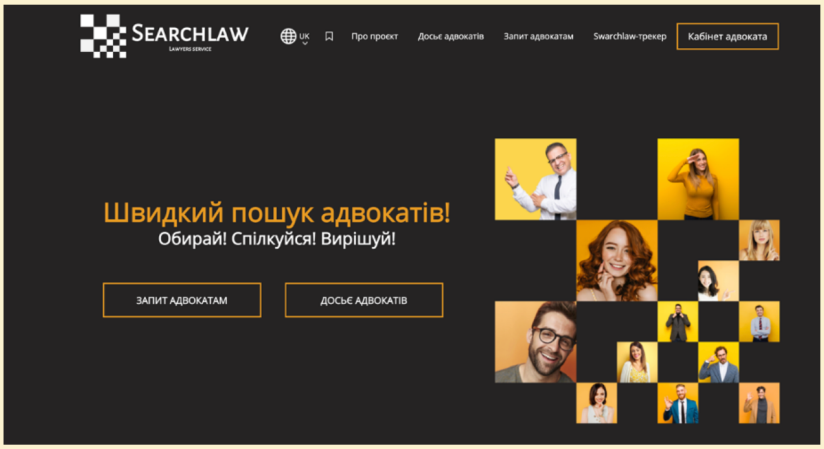

Рисунок Г.7 – Головна сторінка веб-системи

- Кабінет адвоката

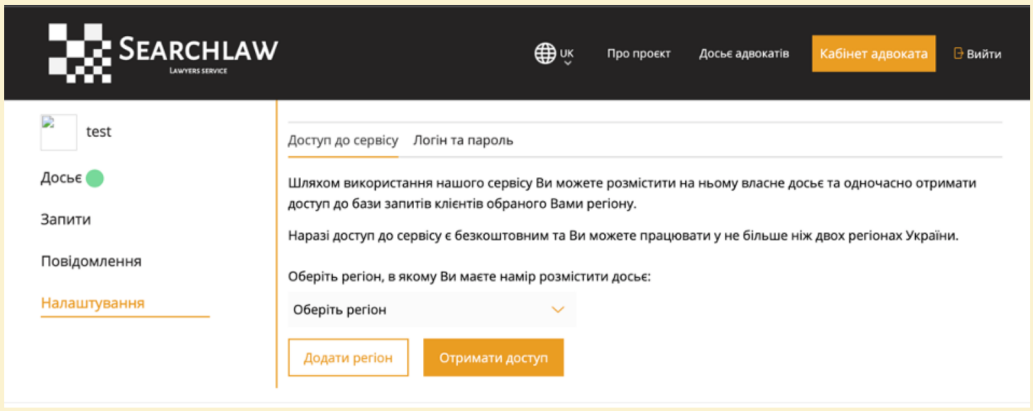

Рисунок Г.8 – Кабінет адвоката

- Сторінка створення досьє адвоката

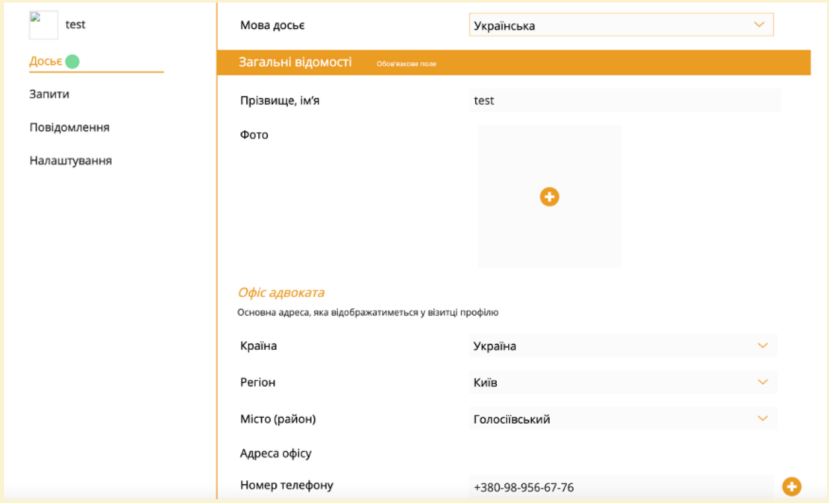

Рисунок Г.9 – Сторінка створення досьє адвоката

- Збережене досьє

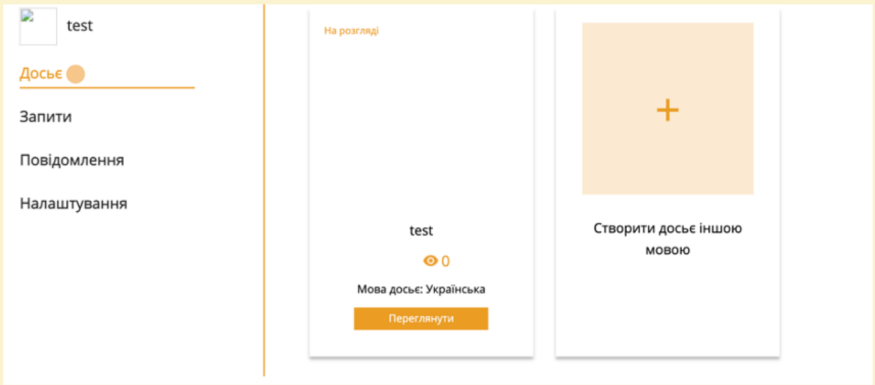

Рисунок Г.10 – Збережене досьє

- Досьє в панелі адміністратора

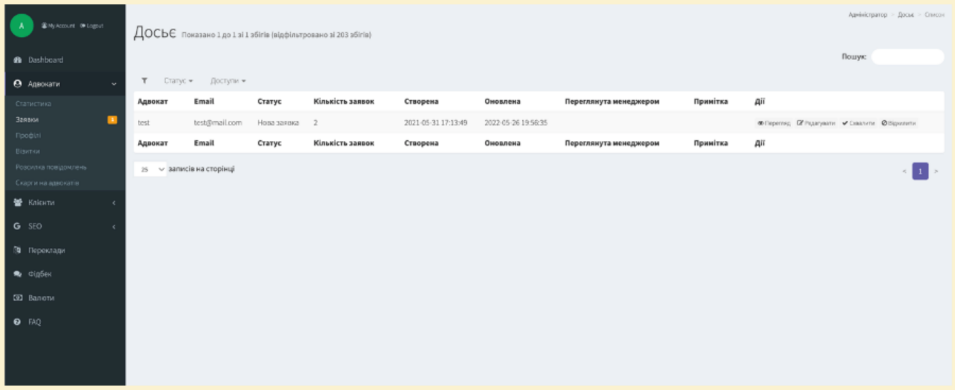

Рисунок Г.11 – Досьє в панелі адміністратора

- Пошук адвоката

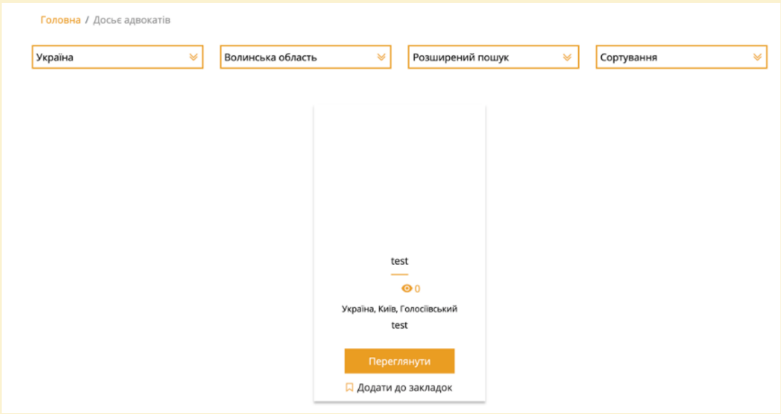

Рисунок Г.12 – Пошук адвоката

- Висновки

  - розроблено базу даних відповідно до вимог веб-системи;
  - розроблено модуль пошуку адвокатів;
  - розроблено модуль для online консультацій;
  - розроблено графічний інтерфейс для веб-середовища;
  - налаштовано взаємодію між модулями пошуку адвокатів, online консультацій та графічним інтерфейсом;
  - проведено тестування веб-системи згідно поставлених задач.

Рисунок Г.13 – Висновки

- Публікації

Основні результати дослідження опубліковані в науковій роботі – в тезах доповіді на науково-технічній конференції підрозділів Вінницького національного технічного університету факультету інформаційних технологій та комп'ютерної інженерії (2022).

https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15436

Рисунок Г.13 – Публікації