

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Дошка оголошень»

Виконав студент групи ІПЗ-21-5
ПАРХОМЧУК Іван Олександрович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
КОНЦИДАЙЛО Андрій Михайлович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАРХОМЧУК Іван Олександрович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Дошка оголошень»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **об'єктом дослідження є онлайн-платформа дошка оголошень**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	27.02.2025р.	27.02.2025р.
2	Чижмотря О.Г.	27.02.2025р.	27.02.2025р.
3	Чижмотря О.Г.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-платформи «Дошка оголошень» та пояснювальної записки. Пояснювальна записка містить 83 сторінки, 28 ілюстрацій, 16 таблиць, 4 додатки і 30 використаних літературних джерел.

Метою роботи є створення зручної та безпечної веб-платформи для розміщення та пошуку некомерційних оголошень із верифікацією користувачів і захистом контактної інформації.

Для реалізації використано фреймворк Laravel, базу даних MySQL, Bootstrap для фронтенду та Vento для адмінпанелі. Застосовано об'єктно-орієнтоване програмування, реляційну модель бази даних і методи забезпечення безпеки (CSRF-токени, middleware). Розроблено функціонал реєстрації, авторизації, створення та фільтрації оголошень, обробки запитів і адміністративного управління. Інтерфейс адаптовано для різних ролей користувачів, що забезпечує інтуїтивну навігацію. Тестування підтвердило відповідність системи вимогам.

Платформа підходить для використання в некомерційних спільнотах для обміну ресурсами. Рекомендується впровадження в локальних громадах із можливістю подальшого розширення функціоналу (аналітика, рейтинги користувачів).

КЛЮЧОВІ СЛОВА: ВЕБ-ПЛАТФОРМА, ДОШКА ОГОЛОШЕНЬ, LARAVEL, MYSQL, ФІЛЬТРАЦІЯ, ПЛАТФОРМА

					ІПЗ.КР.Б – 121 – 25 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Пархомчук І.О.				Розробка онлайн-платформи Дошка оголошень	Літ.	Арк.	Аркушів
Керівник	Чижмоторя О.Г.						3	83
Рецензент	Концидайло А.М.							
Зав. каф.	Вакалюк Т.А.					Житомирська політехніка, група ІПЗ-21-5		

ABSTRACT

The bachelor's thesis consists of a web-based platform “Announcement Board” and an explanatory note. The explanatory note contains 83 pages, 28 illustrations, 16 tables, 4 appendices and 30 references.

The aim of the work is to create a convenient and secure web platform for posting and searching for non-commercial ads with user verification and protection of contact information.

The Laravel framework, MySQL database, Bootstrap for the frontend, and Vemto for the admin panel were used for the implementation. We used object-oriented programming, a relational database model, and security methods (CSRF tokens, middleware). We have developed the functionality of registration, authorization, creation and filtering of ads, request processing, and administrative management. The interface has been adapted for different user roles, providing intuitive navigation. Testing confirmed that the system meets the requirements.

The platform is suitable for use in non-profit communities to share resources. It is recommended to implement it in local communities with the possibility of further expanding the functionality (analytics, user ratings).

KEYWORDS: WEB PLATFORM, ANNOUNCEMENT BOARD, LARAVEL, MYSQL, FILTERING, PLATFORM

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Вибір архітектури веб-орієнтованої системи.....	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	18
Висновки до першого розділу.....	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОШКИ ОГолошень.....	21
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	21
2.2. Розробка бази даних системи.....	28
2.3. Проєктування та реалізація алгоритмів роботи системи	33
2.4. Реалізація функціоналу системи Дошка оголошень	34
Висновки до другого розділу	37
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДОШКА ОГолошень.....	38
3.1. Порядок встановлення та налаштування параметрів платформи	38
3.2. Структура інтерфейсу платформи Дошка оголошень.....	38
3.3. Тестування роботи платформи Дошка оголошень	47
Висновки до третього розділу	51
ВИСНОВКИ.....	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

MVC – Model-View-Controller

CRUD – Create, Read, Update, Delete

ORM – Object-Relational Mapping

CSRF – Cross-Site Request Forgery

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						6

ВСТУП

У сучасному цифровому просторі швидка та ефективна взаємодія між користувачами набуває особливого значення. Веб-додатки формату «дошка оголошень» виконують роль універсальних майданчиків для обміну інформацією, адже дають змогу оперативно розміщувати або знаходити необхідні послуги й товари. Особливу увагу привертають сервіси, подібні до OLX, де користувачі можуть публікувати свої оголошення, налаштовувати зручні фільтри пошуку та спілкуватися у форматі відкритих коментарів. Завдяки цьому суттєво розширюється коло потенційних покупців і продавців, а цифровий ринок стає ще більш прозорим і доступним.

Разом із тим, виникає потреба забезпечити такі важливі аспекти, як безпека, модерація контенту та ефективне адміністрування. Для реалізації зазначених функцій використовується прогресивний фреймворк Laravel, що надає розширені можливості розробки й масштабування. Зокрема, завдяки вбудованим інструментам аутентифікації та авторизації можна створити систему, яка підтримує різні ролі користувачів (адміністратори, розробники, зареєстровані учасники), а модульна структура Laravel полегшує інтеграцію додаткових функцій — від скарг на неприйнятну поведінку до зручного управління категоріями оголошень.

Мета даного дослідження полягає у проектуванні й реалізації надійного веб-ресурсу для розміщення оголошень різного типу («Віддаю»/«Потребую»), який забезпечуватиме зручну взаємодію між учасниками та відповідатиме сучасним вимогам безпеки й функціональності. Для досягнення поставленої мети **визначено такі завдання:**

- Дослідити наявні програмні рішення у сфері дошок оголошень, проаналізувати їх переваги та недоліки.
- Розробити концептуальну модель системи з урахуванням особливостей різних типів оголошень та вимог користувачів.
- Визначити оптимальний технологічний стек (у тому числі фреймворк Laravel) та методи реалізації веб-додатка.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						7

- Забезпечити інтеграцію функцій фільтрації й пошуку, а також розробити механізм публікації коментарів і обробки скарг на користувачів.
- Організувати систему адміністрування для контролю за контентом, управління категоріями, реєстраційними даними користувачів і модерацією оголошень.
- Перевірити працездатність створеної системи за допомогою тестування та визначити перспективи подальшого розвитку й масштабування.

Об'єктом дослідження є інтегрована модель взаємодії користувачів під час публікування та перегляду оголошень, включно з механізмами безпеки і контролю контенту.

Предметом дослідження є використання сучасних веб-технологій і методів об'єктно-орієнтованого програмування (зокрема фреймворка Laravel) для розробки універсальної дошки оголошень, що відповідає вимогам ефективності, безпеки та зручності користування.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка задачі

Проект створення веб-додатку «Дошка оголошень» спрямований на розробку зручного цифрового простору для розміщення оголошень типу «Віддаю/Потребую». Головна ідея – забезпечити користувачам інтуїтивно зрозумілий інтерфейс для публікації та пошуку оголошень без зайвих ускладнень, що дозволить ефективно організувати обмін інформацією між учасниками сервісу.

В умовах стрімкого розвитку електронних сервісів, важливо створювати платформи, що поєднують простоту використання із високою функціональністю та безпекою. У цьому контексті застосування сучасних веб-технологій (зокрема, фреймворка Laravel) дозволяє це реалізувати.

Створення персональних профілів передбачає безпечну реєстрацію та автентифікацію з можливістю використання класичного логіну/пароля або авторизації через соціальні мережі для спрощеного доступу. Організація оголошень за категоріями, додавання параметрів, таких як область чи місто, та забезпечення швидкого пошуку реалізуються через структуровану базу даних для динамічного керування контентом. Сучасний адаптивний дизайн забезпечує зручну навігацію та швидкий доступ до функцій платформи на різних пристроях, від смартфонів до десктопів. Система коментарів під оголошеннями дозволяє користувачам обмінюватися контактами, уточнювати деталі та повідомляти про неправомірні дії. Розширена адміністративна панель забезпечує контроль контенту, управління користувачами та моніторинг активності в реальному часі. Захист даних досягається через шифрування критичної інформації, валідацію даних та застосування сучасних протоколів безпеки для запобігання атакам, таким як SQL-ін'єкції.

Головною задачею є розробка надійного веб-сервісу, який дозволить користувачам легко публікувати свої оголошення та знаходити потрібні пропозиції, забезпечуючи при цьому високий рівень безпеки та зручності

використання. Для цього необхідно вирішити як технічні, так і організаційні завдання, що включають аналіз існуючих рішень, розробку унікальної архітектури системи та інтеграцію сучасних технологій для оптимізації роботи сервісу.

Таким чином, реалізація даного проекту вимагає комплексного підходу, що поєднує розробку програмного забезпечення, впровадження системи адміністрування, а також забезпечення стабільності та безпеки експлуатації платформи.

1.2. Аналіз аналогів програмного продукту

Для покращення процесу розробки веб-додатку “Дошка оголошень” варто провести аналіз аналогічних проєктів на ринку.

Одним із кращих прикладів є платформа OLX (рис. 1.1).

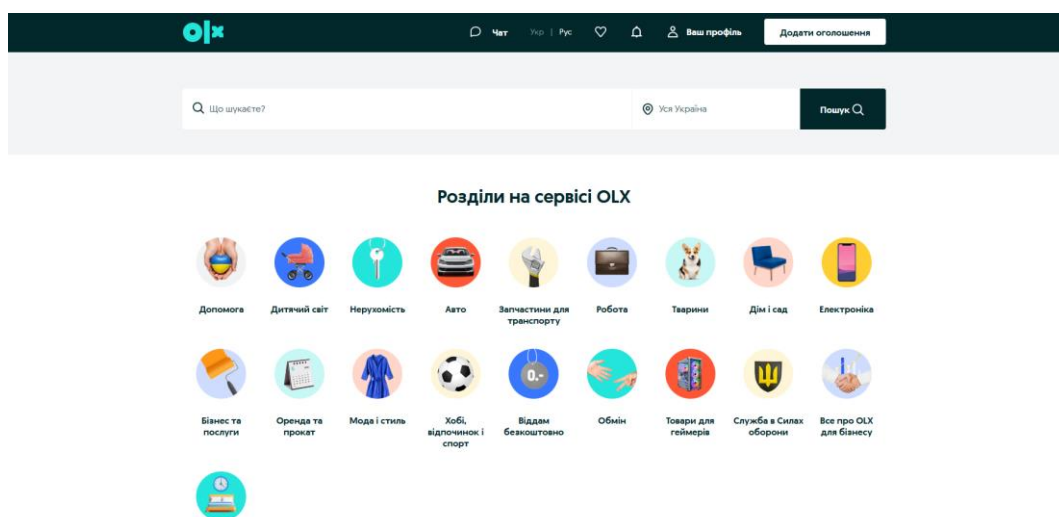


Рис. 1.1. Платформа OLX

OLX є однією з найкращих онлайн-платформ для купівлі чи продажу товарів, послуг, тощо. Має приємний дизайн та зручний функціонал, що дозволяє користувачу легко створити потрібне йому оголошення.

Платформа пропонує широкий вибір оголошень і категорій для зручного орієнтування, що робить її універсальною та доступною завдяки безкоштовності. Користувачі можуть купувати вживані товари за нижчими цінами, а також знаходити послуги, вакансії чи оренду. Зручна система фільтрації та пошуку за регіоном чи містом полегшує знаходження потрібного оголошення, а прямий

зв'язок між покупцем і продавцем спрощує комунікацію. Крім того, доступний мобільний додаток для різних систем підвищує зручність використання.

Проте платформа має певні недоліки. Через легкість розміщення оголошень будь-ким можуть траплятися випадки шахрайства. Велика кількість оголошень ускладнює пошук якісного товару, а висока конкуренція між продавцями може знижувати видимість окремих пропозицій.

Ria – є однією з провідних онлайн-платформ для публікації оголошень (рис. 1.2). В свою чергу вона має розподілення на: Ria, Autoria, Domria, Zapchasti ria, що є більш гнучким для користувача та його потреб, на кшталт пошуку авто, зйому або покупки квартири, автозапчастин, або будь яких інших речей які продають інші.

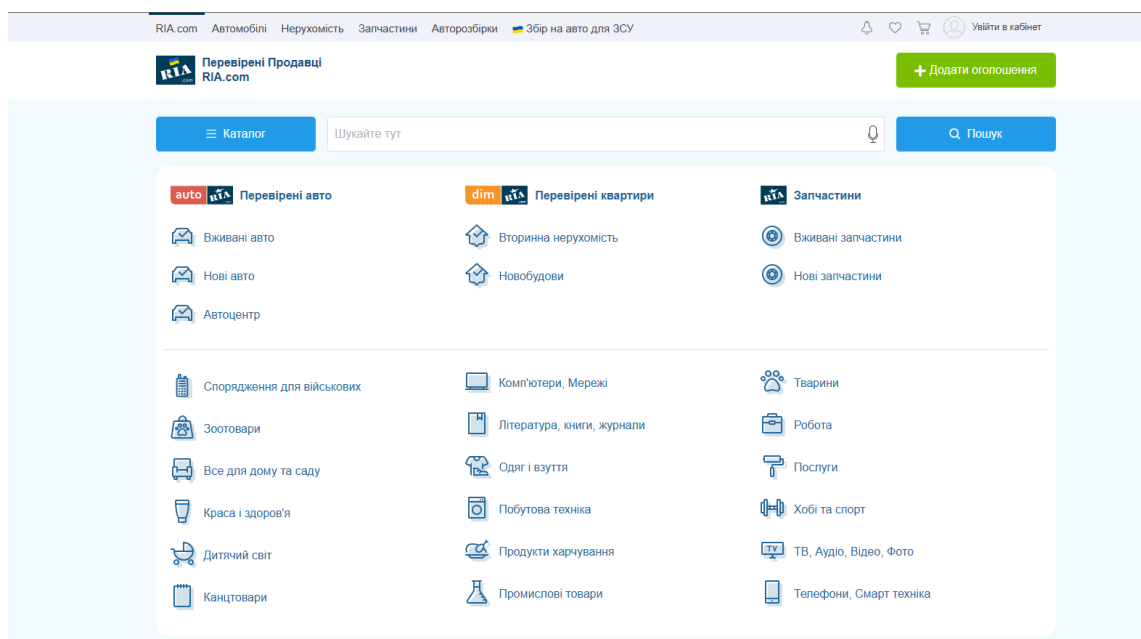


Рис. 1.2. Платформа Ria

Ria є універсальною платформою, яка охоплює найрізноманітніші категорії товарів, включаючи автомобілі, нерухомість, запчастини завдяки своїм спеціалізованим підрозділам. Чітка структура категорій полегшує пошук потрібних товарів чи послуг, а зручна система фільтрації та пошуку за регіоном робить платформу зручною для користувачів. Ria є безкоштовною, що забезпечує її доступність для всіх, а продаж вживаних товарів дозволяє економити на покупках. Крім того, платформа об'єднує товари, послуги та

оренду, що робить її універсальною, а зручний мобільний додаток підвищує комфорт використання.

Однак через відкритий доступ до розміщення оголошень можливі випадки шахрайства. Просування оголошень у топ може бути дорогим, що створює додаткові витрати для продавців. Також платформа є менш універсальною порівняно з OLX у питанні повсякденних товарів, що може обмежувати її привабливість для деяких користувачів.

Bon – раніше besplatka, українська платформа, яка дозволяє розміщувати оголошення в різних категоріях, від товарів до послуг (рис. 1.3).

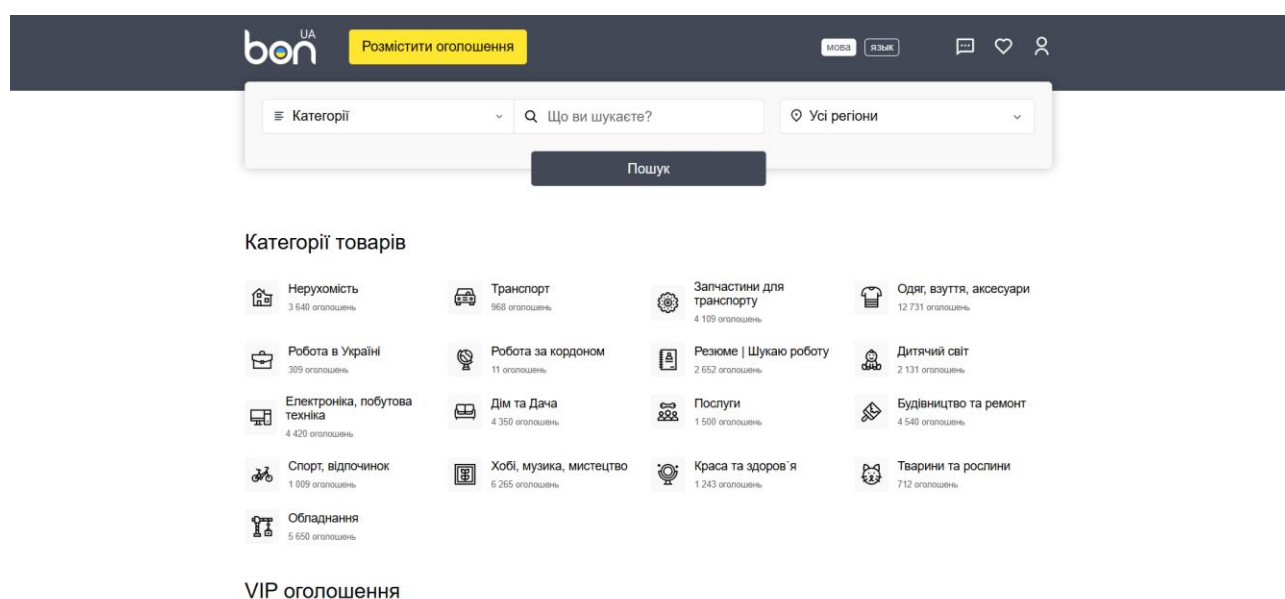


Рис. 1.3. Платформа Bon.ua

Bon є зручною платформою для розміщення оголошень, яка відзначається безкоштовністю, що робить її доступною для всіх користувачів. Простота у використанні та мінімалістичний дизайн забезпечують легкість створення та перегляду оголошень, а підтримка регіональних оголошень дозволяє знаходити пропозиції поблизу. Крім того, швидка публікація без складної реєстрації спрощує процес розміщення оголошень.

Проте обмежена функціональність фільтрації ускладнює пошук потрібних оголошень, а відсутність верифікації користувачів підвищує ризик шахрайства.

Також обмежена підтримка мобільних додатків може створювати незручності для користувачів, які надають перевагу мобільним пристроям.

Prom - це маркетплейс, який поєднує функції дошки оголошень та інтернет-магазину, орієнтований на комерційні пропозиції (рис. 1.4).

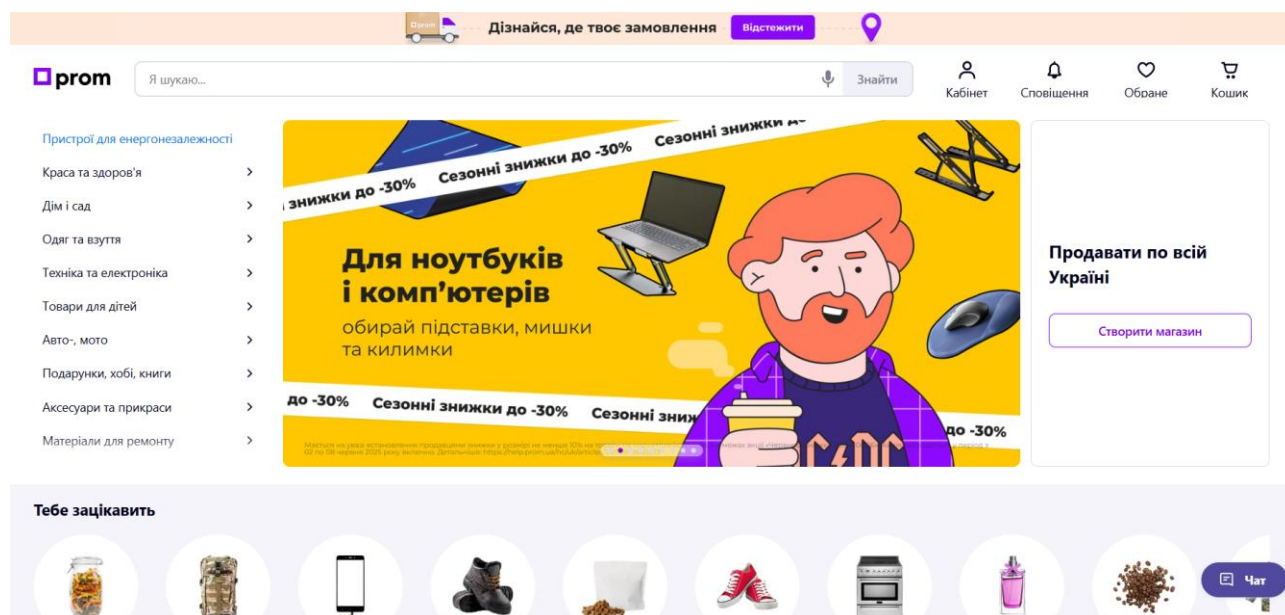


Рис. 1.4. Платформа Prom

Prom є платформою, яка забезпечує зручність транзакцій завдяки інтеграції з платіжними системами, що спрощує процес купівлі та продажу. Вона пропонує широкий вибір категорій товарів, що дозволяє користувачам знаходити різноманітні пропозиції. Можливість створення профілю продавця з відгуками підвищує довіру покупців, а активна підтримка бізнес-користувачів робить платформу привабливою для комерційних клієнтів.

Однак платформа орієнтована виключно на комерційні пропозиції, що обмежує її використання для некомерційних оголошень. Високі комісії для продавців можуть знижувати їхню вигоду, а складна структура сайту може бути незручною для звичайних користувачів, які не мають досвіду роботи з подібними платформами.

Shafa - платформа, орієнтована на продаж одягу, аксесуарів та інших товарів, переважно у форматі вживаних речей (рис. 1.5).

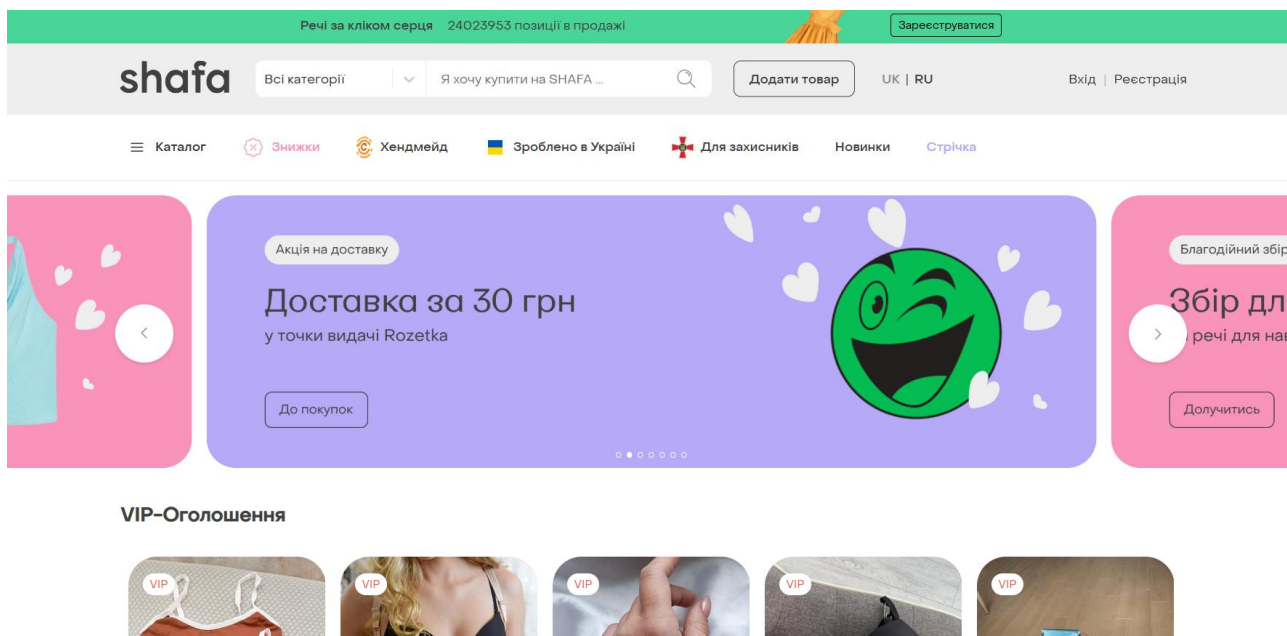


Рис. 1.5. Платформа Shafa

Shafa пропонує зручний інтерфейс для пошуку товарів, що полегшує користувачам навігацію та вибір потрібних пропозицій. Платформа фокусується на вживаних товарах, що сприяє економії коштів. Проста авторизація та створення оголошень роблять процес швидким і доступним, а можливість залишати відгуки про продавців допомагає підвищити довіру до транзакцій.

Проте категорії товарів обмежені, переважно включаючи одяг та аксесуари, що звужує вибір для користувачів. Відсутність верифікації користувачів підвищує ризик шахрайства, а обмежена підтримка регіональних фільтрів ускладнює пошук пропозицій у певних місцевостях. Крім того, платформа не підтримує некомерційні оголошення, що обмежує її універсальність.

Таблиця 1.1

Таблиця порівняння аналогів

Функція / Сервіс	OLX	Ria	Bon	Prom	Shafa
Безкоштовне розміщення	+	-	+	-	+
Широкий вибір категорій	+	+	+	+	-
Локалізований пошук за регіонами	+	+	+	-	-

Продовження таблиці 1.1

Система відгуків та рейтингу	-	-	-	+	+
Інтеграція з платіжними системами	+	+	+	+	+
Функція порівняння оголошень	-	+	-	-	-
Мобільні додатки	+	+	-	+	+

За результатами аналізу платформа OLX демонструє високу популярність завдяки великій аудиторії, широкому вибору категорій та зручності створення й пошуку оголошень, що робить її однією з найбільш затребуваних дошок оголошень. Ria.com вирізняється детальною системою фільтрації та порівняння оголошень, що забезпечує точний пошук і персоналізований підхід, проте її орієнтація на комерційні категорії, такі як автомобілі та нерухомість, обмежує широту використання. Bon пропонує безкоштовне розміщення та підтримку некомерційних оголошень, але програє через обмежену функціональність фільтрації та відсутність мобільних додатків. Prom забезпечує інтеграцію з платіжними системами та систему відгуків, але фокусується виключно на комерційних пропозиціях, що ускладнює використання для некомерційних потреб. Shafa приваблює простотою створення оголошень і системою відгуків, але обмежена категоріями, переважно одягом та аксесуарами.

Для розробки конкурентоспроможної дошки оголошень доцільно поєднати простоту та масовість, властиву OLX, з гнучкістю налаштувань і точністю пошуку, характерною для Ria.com, а також врахувати безкоштовність і підтримку некомерційних оголошень, як у Bon, та елементи довіри, такі як верифікація користувачів і система відгуків, притаманні Prom і Shafa. Власна розробка вирізняється підтримкою некомерційних оголошень («Віддаю» та «Потребую»), верифікацією користувачів для підвищення довіри, безпечною системою запитів замість інтерактивного чату та розширеними адміністративними функціями для управління контентом і користувачами. Це дозволить створити ефективний, безпечний і зручний сервіс, орієнтований на потреби широкої аудиторії.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						15

1.3. Вибір архітектури веб-орієнтованої системи

Розробка системи «Дошка оголошень» потребує ретельного вибору архітектури, яка забезпечить високу продуктивність, масштабованість і зручність для подальшого розвитку проєкту. Архітектура визначає спосіб взаємодії компонентів системи, обробку даних і здатність платформи адаптуватися до зростаючої кількості користувачів. Для забезпечення ефективної роботи системи обрано клієнт-серверну архітектуру, яка чітко розмежовує функціонал між серверною та клієнтською частинами, забезпечуючи гнучкість, безпеку та швидкодію. Клієнт-серверна модель є оптимальною для веб-додатків, оскільки дозволяє обробляти запити користувачів на сервері, зберігаючи дані в централізованій базі, тоді як клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем.

Серверна частина системи побудована на основі фреймворку Laravel, який є одним із провідних інструментів для створення веб-додатків на мові PHP. Laravel вирізняється модульною структурою, що полегшує організацію коду та його підтримку. Ключовою особливістю Laravel є використання Eloquent ORM (Object-Relational Mapping), яке забезпечує зручну роботу з базою даних через об'єктно-орієнтований підхід. Eloquent спрощує створення складних запитів, таких як фільтрація оголошень за категоріями, регіонами чи типами, а також автоматизує зв'язки між таблицями, наприклад, між користувачами, оголошеннями та запитами. Laravel також надає вбудовані інструменти для маршрутизації, аутентифікації, авторизації та обробки помилок, що підвищує безпеку системи та скорочує час розробки. Наприклад, вбудована система авторизації забезпечує захист доступу до профілів користувачів і запитів, а валідація даних гарантує коректність введеної інформації, такої як назви чи описи оголошень.

Для порівняння архітектурних патернів було розглянуто кілька альтернатив: монолітну архітектуру, мікросервісну архітектуру та серверлес-архітектуру. Монолітна архітектура об'єднує всі компоненти системи в єдиному кодовому блоці, що спрощує розробку та розгортання, але може ускладнити

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						16

масштабування при значному зростанні навантаження. Мікросервісна архітектура розбиває систему на незалежні сервіси, кожен із яких відповідає за певну функцію (наприклад, управління оголошеннями чи користувачами). Вона забезпечує високу масштабованість, але вимагає складнішого управління та більших ресурсів на початковому етапі. Серверлес-архітектура делегує управління серверною інфраструктурою стороннім сервісам, таким як AWS Lambda, але менш підходить для проєктів із чіткою структурою бази даних і складною логікою, як у «Дошки оголошень». З огляду на потреби проєкту – простоту розробки, помірне навантаження та необхідність швидкого розгортання – обрано монолітну архітектуру на базі Laravel. Вона дозволяє швидко реалізувати всі необхідні функції, забезпечує легкість у підтримці коду та дає можливість у майбутньому перейти до мікросервісів за потреби.

Клієнтська частина системи реалізована з використанням фреймворку Bootstrap, який забезпечує створення адаптивного та інтуїтивно зрозумілого інтерфейсу. Bootstrap дозволяє розробляти інтерфейс, що коректно відображається на пристроях із різними розмірами екранів – від десктопів до смартфонів. Готові компоненти Bootstrap, такі як кнопки, модальні вікна та форми, прискорюють розробку клієнтської частини, забезпечуючи сучасний і приємний вигляд платформи. Наприклад, модальні вікна використовуються для відправлення скарг, звернень до підтримки чи створення запитів до оголошень. Для адміністративної панелі застосовано інструмент Vemto, який інтегрується з Laravel і автоматично генерує шаблони та компоненти для управління користувачами, оголошеннями, регіонами статичними. Vemto значно скорочує час розробки адмінпанелі, надаючи зручний інтерфейс для виконання CRUD-операцій (Create, Read, Update, Delete) і забезпечуючи гнучке налаштування.

Обрана клієнт-серверна архітектура на базі Laravel із Eloquent ORM, Bootstrap для клієнтської частини та Vemto для адмінпанелі створює надійну, гнучку та масштабовану систему. Ця архітектура забезпечує оптимальну взаємодію між компонентами, високу продуктивність і можливість швидкого

впровадження нових функцій у майбутньому, що відповідає потребам проєкту «Дошка оголошень».

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для реалізації системи «Дошка оголошень» обрано технологічний стек, який поєднує сучасні, перевірені інструменти веб-розробки, що забезпечують високу продуктивність, безпеку та легкість масштабування. Вибір інструментів ґрунтується на потребах проєкту, зокрема на необхідності створення простої, інтуїтивної та надійної платформи, яка підтримує некомерційні оголошення та забезпечує безпечну взаємодію між користувачами.

Серверна частина розробляється на основі фреймворку Laravel, який є одним із провідних інструментів для створення веб-додатків на PHP. Laravel вирізняється модульною архітектурою, що полегшує організацію коду та його підтримку. Важливою перевагою є Eloquent ORM, який забезпечує зручну роботу з базою даних через об'єктно-орієнтований підхід. Eloquent спрощує створення складних запитів, таких як фільтрація оголошень за категоріями, регіонами чи типами («Віддаю» чи «Потребую»), а також автоматизує зв'язки між таблицями, наприклад, між оголошеннями та запитами. Laravel надає вбудовані інструменти для аутентифікації, авторизації, валідації даних і обробки помилок, що підвищує безпеку системи. Наприклад, валідація даних забезпечує коректність введеної інформації, такої як назви чи описи оголошень, а система авторизації захищає доступ до профілів і запитів, дозволяючи підтвердженим користувачам бачити контактну інформацію.

Клієнтська частина системи створюється з використанням Bootstrap – популярного фреймворку для розробки адаптивних інтерфейсів. Bootstrap забезпечує сучасний і зручний дизайн, який коректно відображається на різних пристроях, від десктопів до мобільних телефонів. Його готові компоненти, такі як форми, кнопки та модальні вікна, дозволяють швидко створювати інтуїтивно зрозумілий інтерфейс. Наприклад, модальні вікна використовуються для відправлення запитів до оголошень, скарг або звернень до підтримки.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						18

Адаптивність Bootstrap забезпечує комфортну роботу користувачів на будь-яких пристроях, що є критично важливим для доступності платформи.

Адміністративна панель системи створюється за допомогою інструменту Vemto, який інтегрується з Laravel і дозволяє швидко генерувати шаблони для управління користувачами, оголошеннями, регіонами та статичними сторінками. Vemto автоматизує створення CRUD-операцій (Create, Read, Update, Delete), що значно скорочує час розробки адмінпанелі. Наприклад, адміністратор може легко підтверджувати користувачів, видаляти чи редагувати оголошення або керувати статичними сторінками, такими як «Про нас» чи «Контакти», через зручний інтерфейс. Vemto також забезпечує гнучке налаштування, що дозволяє адаптувати адмінпанель до специфічних потреб проєкту.

Для зберігання даних обрано реляційну базу даних MySQL, яка інтегрується з Laravel через Eloquent ORM. MySQL є надійною та широко використовуваною СУБД, яка підтримує складні запити та зв'язки між таблицями, що необхідно для управління даними користувачів, оголошень, запитів і регіонів. Наприклад, таблиці оголошень і запитів пов'язані через ключі, що дозволяє швидко отримувати інформацію про запити до певного оголошення чи оголошення конкретного користувача. MySQL забезпечує швидку обробку даних і високу надійність, що критично для платформи з великою кількістю оголошень.

Обраний технологічний стек – Laravel із Eloquent ORM, MySQL, Bootstrap і Vemto – забезпечує створення стабільної, безпечної та масштабованої системи. Ці інструменти дозволяють реалізувати всі функціональні вимоги проєкту, такі як створення та фільтрація оголошень, верифікація користувачів, безпечна комунікація через запити та управління контентом. Система оптимізована для швидкої розробки, легкої підтримки та подальшого розширення, що робить її ідеальним рішенням для платформи «Дошка оголошень».

Висновки до першого розділу

У першому підрозділі окреслено ключові елементи, необхідні для розробки системи «Дошка оголошень», а також визначено основні технічні

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						19

вимоги до проєкту. Це дозволило сформувати базове розуміння компонентів, таких як механізми реєстрації, авторизації, управління оголошеннями та функціонал комунікації між користувачами, що є критично важливими для подальшої реалізації рішення.

У другому підрозділі проаналізовано платформи OLX, Ria.com, Bon, Prom і Shafa, порівняно їхній функціонал, виявлено переваги (широкі категорії, фільтрація) і недоліки (відсутність верифікації, обмежена підтримка некомерційних оголошень). Це допомогло визначити напрямки вдосконалення, зокрема простий інтерфейс, верифікацію та безпечну комунікацію.

У третьому підрозділі здійснено вибір архітектури веб-системи, оптимально адаптованої для нашої платформи. Обрано серверну частину на базі Laravel, що забезпечує надійність, масштабованість та зручне управління даними через Eloquent ORM, а адміністративний інтерфейс створюється із застосуванням Vemto. Такий підхід дозволяє ефективно організувати бізнес-логіку системи, забезпечити її безпеку та швидко впроваджувати нові функціональні можливості.

У четвертому підрозділі визначено інструментальні засоби та вимоги до апаратного забезпечення для підтримки стабільної роботи платформи. Використання Laravel, Vemto та Bootstrap дозволяє створити інтуїтивно зрозумілий та адаптивний інтерфейс, а робота з базою даних за допомогою Eloquent ORM забезпечує ефективне управління інформацією.

Ці рішення дозволяють створити стабільну, безпечну та масштабовану платформу «Дошка оголошень», яка відповідає сучасним вимогам ринку та забезпечує ефективну взаємодію з користувачами навіть при високих навантаженнях.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОШКИ ОГолошень

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Онлайн платформа Дошка оголошень надає можливість користувачам обмінюватись своїми речами абсолютно безкоштовно. Користувачів можна розділити на звичайних, які можуть лише переглядати оголошення та на підтверджених, хто створює оголошення про потребу у певному продукті або про можливість його віддати, а також на адміністраторів які керують всім, підтверджують користувачів і т.д..

Головна ціль розробки онлайн платформи Дошка оголошень, полягає у створенні веб-середовища спрямованого на безкоштовність. Платформа дозволяє користувачам абсолютно безкоштовно віддавати або отримувати певні речі та забезпечує зручне спілкування та комунікацію між користувачами, а також можливість зручного пошуку та фільтрації, включно з озподіленням по регіонах, для забезпечення швидкого знаходження потрібного оголошення.

Головна мета заключається у можливості легко та зручно займатись безкоштовністю абсолютно будь кому.

Для незареєстрованих користувачів передбачена можливість переглядати списки оголошень, застосовувати фільтри та здійснювати пошук за різними параметрами, а також ознайомлюватися з деталями оголошень та інформацією про їхніх авторів. Це дозволяє відвідувачам сайту швидко знаходити потрібну інформацію без необхідності створення облікового запису.

Зареєстровані, але непідтверджені користувачі отримують розширений доступ до функціоналу платформи. Вони можуть пройти процес авторизації, переглядати та редагувати власний профіль, створювати, редагувати або деактивувати оголошення. Також такі користувачі мають можливість переглядати список власних оголошень, відправляти запити на оголошення інших користувачів, а також подавати скарги або звернення до служби підтримки.

Підтверджені користувачі отримують додаткову перевагу — доступ до контактної інформації авторів оголошень, що спрощує комунікацію та організацію благодійної діяльності.

Адміністратори платформи мають широкий спектр можливостей для управління системою. Вони відповідають за підтвердження профілів користувачів, керування оголошеннями, перегляд інформації про запити на оголошення, а також управління користувачами, областями, містами та статичними сторінками сайту.

Функціональні вимоги до системи включають забезпечення процесів реєстрації та авторизації користувачів. Платформа дозволяє створювати, редагувати та деактивувати оголошення, а також переглядати список власних оголошень разом із їхніми статусами. Система підтримує фільтрацію оголошень за категоріями, областями, містами, типами та назвою. Користувачі можуть відправляти запити на оголошення, при цьому контактна інформація залишається прихованою для всіх, окрім автора оголошення, адміністратора та відправника запиту. Підтверджені користувачі отримують доступ до контактної інформації авторів оголошень. Для зручності подачі скарг та звернень до підтримки система використовує модальні форми.

Нефункціональні вимоги стосуються зручності, масштабованості та надійності системи. Інтерфейс платформи розроблено таким чином, щоб освоєння сайту займало від 3 до 5 хвилин для нових користувачів, а для досвідчених — менше 2 хвилин. Час відповіді системи на дії користувача не перевищує 3 секунд. Дизайн інтерфейсу є інтуїтивно зрозумілим, з чітким поділом на розділи “віддаю” та “потребую”, підтримкою модальних форм та мінімалістичним дизайном, що включає адаптивність для різних пристроїв. Система розрахована на зростання кількості оголошень та користувачів, забезпечуючи при цьому надійне збереження даних.

Системні вимоги до середовища виконання включають наявність щонайменше 2 ГБ оперативної пам’яті, процесора з тактовою частотою не нижче 1.2 GHz, підключення до мережі Інтернет та доступ через сучасні веббраузери.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						22

На серверній стороні система використовує реляційну базу даних MySQL для зберігання інформації про користувачів, оголошення, запити, області, міста та статичні сторінки. Для реалізації серверної логіки застосовується технологія Laravel.

Користувачі мають зареєструватися, якщо ще цього не зробили, або авторизуватися, якщо вже мають акаунт, для того щоб створити нове оголошення.

Таблиця 2.1

Таблиця прецеденту “Реєстрація/Авторизація”

Назва варіанту використання	Реєстрація/Авторизація
Основні актори	Користувач
Короткий опис	Користувачі мають зареєструватися якщо ще цього не зробили або авторизуватися якщо вже мають акаунт для того щоб створити нове оголошення

Керування оголошенням описує можливості користувача щодо роботи з власними оголошеннями. Ця таблиця визначає дії, які дозволяють створювати, редагувати або деактивувати оголошення.

Таблиця 2.2

Таблиця прецеденту “Керування оголошенням”

Назва варіанту використання	Керування оголошенням
Основні актори	Користувач
Короткий опис	Користувач може створювати оголошення, редагувати його та деактивувати

Функціонал перегляду та фільтрації оголошень у системі забезпечує зручний пошук і доступ до оголошень. Таблиця детально роз’яснює, як користувачі можуть ефективно застосовувати різноманітні фільтри для швидкого та точного знаходження потрібних оголошень відповідно до їхніх потреб.

Таблиця 2.3

Таблиця прецеденту “Перегляд і фільтрація оголошень”

Назва варіанту використання	Перегляд і фільтрація оголошень
Основні актори	Користувач
Короткий опис	Користувач може переглядати оголошення та фільтрувати їх за потрібними варіантами фільтрації

Функціонал відправлення запиту на оголошення детально описує процес взаємодії користувача з оголошеннями, створеними іншими користувачами системи. Таблиця наочно пояснює, як формується запит із включенням контактних даних, забезпечуючи при цьому надійне збереження конфіденційності інформації для всіх сторін, окрім автора, адміністратора та відправника.

Таблиця 2.4

Таблиця прецеденту “Відправлення запиту на оголошення”

Назва варіанту використання	Відправлення запиту на оголошення
Основні актори	Користувач
Короткий опис	Користувач може відправити запит на оголошення через форму вказавши ім'я та контактні дані. Запит відображається в коментарях оголошення, але контактна інформація прихована

Функціонал перегляду профілю та оголошень користувача забезпечує зручну та ефективну можливість ознайомлення з інформацією про інших учасників платформи. Таблиця чітко та детально описує, яким чином користувач може отримати доступ до профілю іншого користувача, переглянути створені ним оголошення, а також ознайомитися з відповідними деталями їхнього вмісту, що сприяє зручній взаємодії в системі.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						24

Таблиця 2.5

Таблиця прецеденту “Перегляд профілю та оголошень користувача”

Назва варіанту використання	Перегляд профілю та оголошень користувача
Основні актори	Користувач
Короткий опис	Користувач може переглядати профіль іншого користувача з його оголошеннями

Функціонал відправлення скарги або запиту на підтримку забезпечує зручну та ефективну взаємодію користувача зі службою підтримки системи. Таблиця чітко роз’яснює, як користувачі можуть створювати запити чи скарги, використовуючи модальну форму, яка автоматично додає контекст сторінки, що значно полегшує обробку звернень адміністраторами та сприяє швидкому вирішенню питань.

Таблиця 2.6

Таблиця прецеденту “Відправлення скарги/запиту на підтримку”

Назва варіанту використання	Відправлення скарги/запиту на підтримку
Основні актори	Користувач
Короткий опис	Користувач може відкрити модальну форму для відправлення скарги або запиту на підтримку обираючи тему та текст повідомлення, а також автоматично додається посилання на сторінку з якої відправлена форма

Адміністративне управління охоплює широкий набір інструментів і функціональних можливостей, призначених для ефективного керування платформою адміністратором. Таблиця детально роз’яснює повноваження адміністратора, які включають управління профілями користувачів, створення, редагування та видалення оголошень, а також контроль над іншими ключовими сутностями, такими як категорії, регіони, міста та статичні сторінки. Це

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						25

забезпечує повний контроль над усіма аспектами системи, сприяючи її стабільному функціонуванню та зручності для користувачів.

Таблиця 2.7

Таблиця прецеденту “Адміністративне управління”

Назва варіанту використання	Адміністративне управління
Основні актори	Адміністратор
Короткий опис	Адміністратор може підтверджувати користувачів, та керувати: користувачами, оголошеннями, категоріями, запитами, областями/містами та статичними сторінками

За допомогою аналізу функціональних вимог, ми виділяємо наступні сутності, які в свою чергу забезпечують реалізацію нашої онлайн платформи. Діаграма класів, що використовується для її роботи наведена на рисунку 2.1.

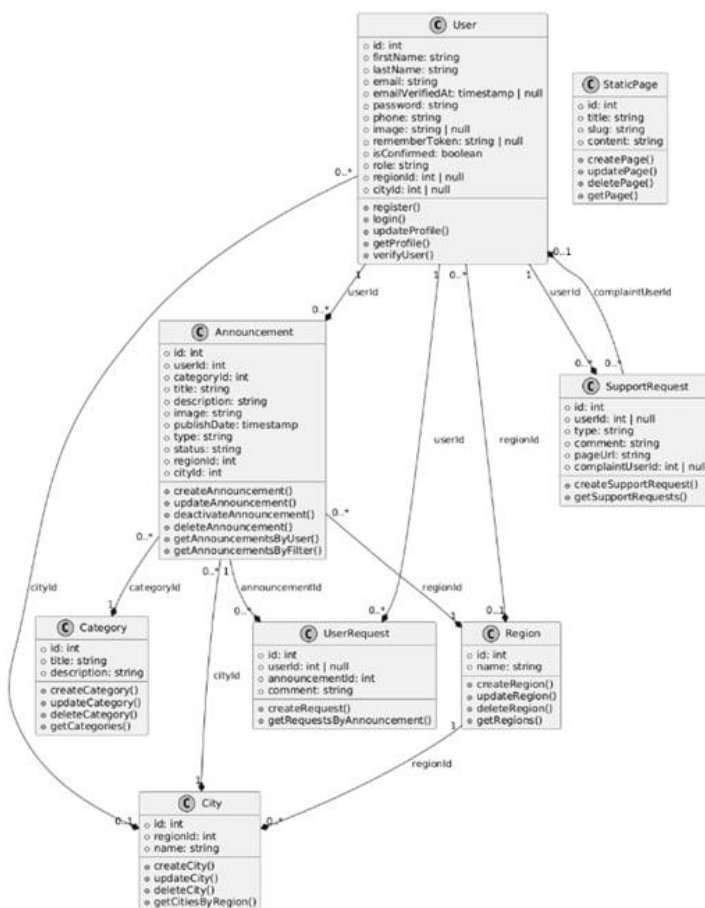


Рис. 2.1. Діаграма класів

Клас User відповідає за користувача. Метод register() забезпечує реєстрацію нового користувача з указанням імені, прізвища, email, пароля, телефону та опціонально регіону й міста. Метод login() відповідає за авторизацію користувача за email і паролем. Метод updateProfile() дозволяє редагувати дані профілю, такі як ім'я, прізвище, телефон, фото, регіон, місто. Метод getProfile() повертає дані профілю користувача. Метод verifyUser() підтверджує статус користувача адміністратором шляхом встановлення атрибута isConfirmed.

Клас Announcement відповідає за оголошення. Метод createAnnouncement() дозволяє користувачу створювати нове оголошення з указанням категорії, назви, опису, фото, регіону, міста, типу ("Віддаю" або "Потребую"). Метод updateAnnouncement() редагує існуюче оголошення. Метод deactivateAnnouncement() деактивує оголошення з указанням причини, наприклад, "Видалено" або "Віддав/Знайшов". Метод deleteAnnouncement() видаляє оголошення адміністратором. Метод getAnnouncementsByUser() повертає всі оголошення конкретного користувача. Метод getAnnouncementsByFilter() повертає оголошення за критеріями фільтрації, такими як категорія, регіон, місто, тип, ключові слова.

Клас Category відповідає за категорії. Метод createCategory() дозволяє адміністратору створювати нову категорію. Метод updateCategory() редагує категорію. Метод deleteCategory() видаляє категорію. Метод getCategories() повертає список усіх категорій.

Клас Region відповідає за регіони. Метод createRegion() дозволяє адміністратору створювати новий регіон. Метод updateRegion() редагує регіон. Метод deleteRegion() видаляє регіон. Метод getRegions() повертає список усіх регіонів.

Клас City відповідає за міста. Метод createCity() дозволяє адміністратору створювати нове місто. Метод updateCity() редагує місто. Метод deleteCity() видаляє місто. Метод getCitiesByRegion() повертає список міст для конкретного регіону.

Клас `UserRequest` відповідає за запити до оголошень. Метод `createRequest()` створює новий запит до оголошення з указанням коментаря, що включає контактну інформацію. Метод `getRequestsByAnnouncement()` повертає всі запити до конкретного оголошення, приховуючи контакт.

Клас `SupportRequest` відповідає за звернення до підтримки. Метод `createSupportRequest()` створює нове звернення до підтримки з указанням типу (скарга, підтвердження, підтримка), тексту та посилання на сторінку. Метод `getSupportRequests()` повертає всі звернення до підтримки, доступні адміністратору.

Клас `StaticPage` відповідає за статичні сторінки. Метод `createPage()` дозволяє адміністратору створювати нову статичну сторінку. Метод `updatePage()` редагує вміст сторінки. Метод `deletePage()` видаляє сторінку. Метод `getPage()` повертає вміст сторінки за її ідентифікатором або slug.

2.2. Розробка бази даних системи

Для збереження інформації в додатку "Дошка оголошень" використовується MySQL – система управління базами даних, яка базується на реляційній моделі. MySQL забезпечує ефективне зберігання та обробку даних про користувачів, оголошення, запити, звернення до підтримки, категорії, регіони, міста та статичні сторінки.

Опис кожної з таблиць бази даних наведений у таблицях 2.8 – 2.16

Таблиця `Users` призначена для зберігання інформації про користувачів додатку "Дошка оголошень". Вона містить дані, необхідні для авторизації, профілю та ролі користувача.

Таблиця 2.8

Таблиця `Users`

Поле	Тип	Призначення
<code>id</code>	Big increments (PK)	Ідентифікатор користувача
<code>firstName</code>	String	Ім'я користувача

Продовження таблиці 2.8

lastName	String	Прізвище користувача
Email	String	Email користувача
emailVerifiedAt	Timestamp	Дата підтвердження email
Password	String	Зашифрований пароль користувача
Phone	String	Номер телефону користувача
Image	String	Посилання на фото профілю
rememberToken	String	Токен для функції “Запам’ятати мене”
isConfirmed	Boolean	Статус підтвердження користувача
Role	Integer	Роль користувача
regionId	Unsigned Big Integer	Id регіону користувача
cityId	Unsigned Big Integer	Id міста користувача

Таблиця Announcements використовується для зберігання даних про оголошення, створені користувачами. Вона містить інформацію про автора, категорію, статус та географічні дані оголошення.

Таблиця 2.9

Таблиця Announcements

Поле	Тип	Призначення
Id	Big increments (PK)	Ідентифікатор оголошення
userId	Unsigned Big Integer	Id автора оголошення
categoryId	Unsigned Big Integer	Id категорії
Title	String	Назва оголошення

Продовження таблиці 2.9

Description	Text	Опис оголошення
Image	String	Посилання на фотографію
publishDate	Timestamp	Дата публікації оголошення
Type	Integer	Тип оголошення (“Віддаю”/”Потребую”)
Status	Integer	Статус оголошення
regionId	Unsigned Big Integer	Id регіону
cityId	Unsigned Big Integer	Id міста

Таблиця Categories призначена для зберігання інформації про категорії оголошень. Вона дозволяє класифікувати оголошення за їх типом та змістом.

Таблиця 2.10

Таблиця Categories

Поле	Тип	Призначення
Id	Big increments (PK)	Id категорії
Title	String	Назва категорії
description	text	Опис категорії

Таблиця Regions зберігає дані про регіони, які використовуються для географічної прив'язки оголошень та користувачів. Вона містить унікальні ідентифікатори та назви регіонів.

Таблиця 2.11

Таблиця Regions

Поле	Тип	Призначення
Id	Big increments (PK)	Id регіону
name	string	Назва регіону

Таблиця Cities використовується для зберігання інформації про міста, пов'язані з регіонами. Вона забезпечує деталізацію географічного розташування в системі.

Таблиця 2.12

Таблиця Cities

Поле	Тип	Призначення
Id	Big increments (PK)	Id міста
regionId	Unsigned Big Integer	Id регіону
name	string	Назва регіону

Таблиця UserRequests призначена для зберігання запитів користувачів до оголошень. Вона містить коментарі та пов'язує запити з користувачами та оголошеннями.

Таблиця 2.13

Таблиця UserRequests

Поле	Тип	Призначення
Id	Big increments (PK)	Id запиту користувача
userId	Unsigned Big Integer	Id користувача
announcementId	Unsigned Big Integer	Id оголошення
comment	Text	Коментар з інформацією

Таблиця SupportRequests призначена для зберігання даних про звернення користувачів до служби підтримки платформи. Вона містить інформацію про тип запиту, коментарі, контекстну інформацію про сторінку, з якої було відправлено звернення, а також додаткові деталі, що сприяють швидкому та ефективному обробленню запитів адміністраторами системи.

Таблиця 2.14

Таблиця SupportRequests

Поле	Тип	Призначення
Id	Big increments (PK)	Id запиту підтримки
userId	Unsigned Big Integer	Id користувача
type	Integer	Тип запиту (Скарга/Підтримка)
comment	Text	Текст запиту

Продовження таблиці 2.14

pageUrl	String	Посилання на сторінку з якої зробили запит
complaintUserId	Unsigned Big Integer	Id користувача, на якого подано скаргу

Таблиця StaticPages використовується для зберігання вмісту статичних сторінок додатку. Вона містить інформацію про назву, унікальний ідентифікатор та вміст сторінки.

Таблиця 2.15

Опис таблиці StaticPages

Поле	Тип	Призначення
Id	Big increments (PK)	Id запиту підтримки
Title	String	Назва сторінки
Slug	String	Унікальний ідентифікатор для URL
Content	Text	Вміст сторінки

Коли користувач створює нове оголошення, система записує його дані в таблицю Announcements, включаючи інформацію про автора (userId), категорію (categoryId), регіон (regionId) і місто (cityId). Ці дані пов'язуються з відповідними записами в таблицях Users, Categories, Regions і Cities. Після створення оголошення воно стає доступним для перегляду та фільтрації іншими користувачами.

Після створення запиту до оголошення через таблицю UserRequests, система зберігає коментар із контактною інформацією (comment), але доступ до цього поля обмежується: лише автор оголошення, адміністратор і відправник можуть переглядати ці дані. Це забезпечується на рівні програмного коду через перевірку прав доступу перед виведенням інформації.

Дані в таблицях, таких як Users, Announcements і UserRequests, є захищеними від несанкціонованого доступу. Наприклад, паролі користувачів у таблиці Users зберігаються в зашифрованому вигляді (поле password), а

контактна інформація в запитах (UserRequests.comment) приховується від сторонніх. Це забезпечує безпеку персональних даних і відповідність нефункціональним вимогам системи.

Для звернення до даних система використовує SQL-запити, які виконуються через фреймворк Laravel, зазначений у системних вимогах. Laravel забезпечує ORM (об'єктно-реляційне відображення) через Eloquent, що дозволяє працювати з даними як із об'єктами. Наприклад, для отримання оголошень користувача система виконує запит до таблиці Announcements із фільтрацією за userId. Для фільтрації оголошень за категоріями, регіонами чи містами використовуються з'єднання (JOIN) із таблицями Categories, Regions і Cities.

Зміни даних, такі як редагування оголошення чи деактивація, також виконуються через SQL-запити. Наприклад, при деактивації оголошення оновлюється поле status у таблиці Announcements. Усі операції запису та оновлення даних супроводжуються перевітками прав доступу: лише автор оголошення або адміністратор може змінювати його статус.

2.3. Проєктування та реалізація алгоритмів роботи системи

Додаток "Дошка оголошень" є простим у використанні та має всі основні функції платформи, що дозволяє користувачам легко взаємодіяти з системою, обмінюватися речами на благодійній основі та здійснювати зручний пошук і фільтрацію оголошень. Алгоритм роботи з платформою детально описано його пункти нижче.

Після авторизації адміністратор потрапляє до адміністративної панелі, де отримує доступ до всіх ключових функцій управління платформою. Адміністратор може переглядати список усіх користувачів і підтверджувати їхній статус, змінюючи відповідне поле в базі даних, якщо користувачі відповідають умовам платформи. Крім того, адміністратор має можливість переглядати та видаляти оголошення, а також ознайомлюватися із запитами до них, включаючи контактну інформацію, зазначену в коментарях. Адміністратор може створювати, редагувати або видаляти категорії, регіони та міста, а також керувати статичними сторінками, змінюючи їхній вміст або видаляючи їх. Також

адміністратор переглядає звернення до служби підтримки, включаючи скарги на користувачів, і реагує на них відповідно до потреб.

Зареєстрований користувач після авторизації отримує доступ до розширеного функціоналу платформи. Він може переглядати список активних оголошень і застосовувати фільтрацію за категоріями, регіонами, містами, типом оголошення ("Віддаю" або "Потребую") та ключовими словами. Користувач має можливість ознайомитися з профілем іншого користувача та його оголошеннями. Також зареєстрований користувач може створювати нове оголошення в розділі "Віддаю" або "Потребую", вказуючи категорію, регіон, місто, назву, опис і додаючи фото. Він має змогу редагувати або деактивувати власні оголошення, змінюючи їхні дані або статус. Крім того, користувач може переглядати список власних оголошень разом із їхніми статусами. Для взаємодії з іншими оголошеннями користувач може відправляти запити, додаючи коментар із контактною інформацією, яка залишається прихованою для всіх, крім автора оголошення, адміністратора та самого відправника. У разі потреби користувач може створити звернення до служби підтримки або подати скаргу через модальну форму, вказавши тип звернення, текст і, якщо необхідно, інформацію про користувача, на якого подається скарга.

Незареєстрований користувач, або гість, має обмежений доступ до функціоналу платформи. Він може переглядати список активних оголошень і застосовувати фільтрацію за категоріями, регіонами, містами, типом оголошення та ключовими словами. Гість також може озна21 із оголошенням та профілем його автора, але без можливості перегляду контактної інформації, такої як номер телефону чи електронна пошта.

2.4. Реалізація функціоналу системи Дошка оголошень

Розглянемо основні етапи роботи з платформою "Дошка оголошень". Для початку користувач повинен зареєструватися та авторизуватися, після чого він отримує можливість створювати оголошення, відправляти запити та взаємодіяти з іншими користувачами. Адміністратор, у свою чергу, має доступ до панелі управління для підтвердження користувачів і керування контентом.

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						34

```

public function store(AnnouncementStoreRequest $request):
RedirectResponse
{
    $this->authorize('create', Announcement::class);

    $validated = $request->validated();
    if ($request->hasFile('image')) {
        $validated['image'] = $request->file('image')-
>store('public');
    }

    $announcement = Announcement::create($validated);

    return redirect()
        ->route('announcements.edit', $announcement)
        ->withSuccess(__('crud.common.created'));
}

```

Після створення оголошення користувач може переглядати запити до нього, які надходять від інших користувачів. Одним із ключових моментів є приховування контактної інформації в запитах: поле comment у таблиці UserRequests доступне лише автору оголошення, адміністратору та відправнику запиту. Для реалізації цього функціоналу створено окремий метод, який перевіряє права доступу перед виведенням даних.

```

public function userRequests()
{
    return $this->hasMany(UserRequest::class);
}

```

Адміністратор має доступ до панелі управління, де може підтверджувати користувачів, змінюючи поле isConfirmed у таблиці Users. Підтвердження користувачів дозволяє їм отримувати доступ до контактної інформації авторів оголошень. Цей процес включає перевірку прав доступу адміністратора та оновлення запису в базі даних.

```

protected $fillable = [
    'firstname',
    'lastname',
    'email',
    'password',
    'phone',
    'image',
    'is_confirmed',
    'role',
    'region_id',
    'city_id',
];

```

```
protected $casts = [
    'email_verified_at' => 'datetime',
    'is_confirmed' => 'boolean',
];

public function store(UserStoreRequest $request): RedirectResponse
{
    $this->authorize('create', User::class);

    $validated = $request->validated();

    $validated['password'] = Hash::make($validated['password']);

    if ($request->hasFile('image')) {
        $validated['image'] = $request->file('image')-
>store('public');
    }

    $user = User::create($validated);

    return redirect()
        ->route('users.edit', $user)
        ->withSuccess(__('crud.common.created'));
}
```

Ще одним важливим аспектом є створення звернень до підтримки через модальну форму. Користувач (або гість) може відправити скаргу чи запит на підтримку, вказавши тип звернення, текст і, за потреби, користувача, на якого подається скарга (complaintUserId). Система автоматично додає URL сторінки, з якої надійшло звернення (pageUrl), і зберігає дані в таблиці SupportRequests. Цей функціонал потребує валідації вхідних даних і безпечного збереження інформації.

```
public function store(SupportRequestStoreRequest $request):
RedirectResponse
{
    $this->authorize('create', SupportRequest::class);

    $validated = $request->validated();

    $supportRequest = SupportRequest::create($validated);

    return redirect()
        ->route('support-requests.edit', $supportRequest)
        ->withSuccess(__('crud.common.created'));
}
```

Розглянемо, як відбувається взаємодія з фронтом для відображення оголошень. Для цього використовуються шаблони Blade, які отримують дані від

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						36

контролерів і формують HTML-структуру. Наприклад, сторінка перегляду оголошень відображає список оголошень із можливістю фільтрації. Фронтенд також відповідає за відображення модальних форм, таких як форма звернення до підтримки, і обробку асинхронних запитів (наприклад, через AJAX) для оновлення даних без перезавантаження сторінки.

Висновки до другого розділу

У рамках розробки онлайн-платформи "Дошка оголошень" проаналізовано вимоги та спроектовано об'єктно-орієнтовану структуру системи, включаючи моделі класів і їх взаємодію. Створено реляційну базу даних на MySQL для зберігання даних про користувачів, оголошення, запити та інші сутності. Описано алгоритми роботи для різних ролей користувачів і детально розглянуто реалізацію ключового функціоналу: створення оголошень, фільтрацію, обробку запитів із приховуванням контактів і адміністративне управління.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДОШКА ОГолошень

3.1. Порядок встановлення та налаштування параметрів платформи

Встановлення та налаштування системи «Дошка оголошень» передбачає розгортання її компонентів і конфігурацію для стабільної роботи. Процес включає підготовку серверного середовища, встановлення коду, налаштування бази даних і прав доступу.

Архітектура системи відображена в діаграмі розгортання, яка включає веб-сервер із Laravel, базу даних MySQL для зберігання даних про користувачів, оголошення, запити та регіони, клієнтську частину на Bootstrap і адмінпанель на Vemto. Усі компоненти розгортаються на сервері з Openserver.

Для розгортання встановлюється Openserver із PHP (версія 8.1+), MySQL (версія 8.0+) і Node.js. Код клонується з репозиторію на GitLab, залежності встановлюються через Composer (для PHP) і npm (для клієнтських бібліотек). У файлі .env налаштовуються параметри підключення до MySQL (хост, ім'я бази, логін, пароль) і ключ додатка (APP_KEY). Команда php artisan migrate створює таблиці для оголошень, запитів, користувачів і регіонів.

Клієнтська частина з шаблонами Blade і стилями Bootstrap розміщується в папці resources, а адмінпанель, згенерована Vemto, інтегрується з Laravel для управління контентом. Права доступу налаштовуються через middleware Laravel: звичайні користувачі створюють і редагують оголошення, підтверджені бачать контакти, адміністратори керують усіма функціями. Безпека забезпечується CSRF-токенами та HTTPS.

3.2. Структура інтерфейсу платформи Дошка оголошень

Система «Дошка оголошень» розроблена як веб-додаток із інтуїтивно зрозумілим інтерфейсом, що забезпечує зручну взаємодію для всіх категорій користувачів – звичайних користувачів, підтверджених користувачів і адміністраторів. Інтерфейс створений із використанням фреймворку Bootstrap, що гарантує адаптивність і коректне відображення на різних пристроях, від

десктопів до мобільних девайсів. Для адміністративного функціоналу використано інструмент Vemto, який забезпечує зручне управління контентом.

Взаємодія користувача із системою починається саме з головної сторінки (рис. 3.1). У верхній частині розташовані заголовок, поле пошуку та панель навігації з посиланнями на головну сторінку, профіль, вхід/реєстрацію та контакти. Нижче – блок популярних категорій із посиланням на їхній повний список, блоки оголошень «Віддаю» та «Потребую» (по 3 оголошення) із посиланнями на відповідні сторінки, а також блоки «Статистика» (кількість користувачів, оголошень, категорій) і «Try It Now» із посиланнями на категорії, реєстрацію та створення оголошення.

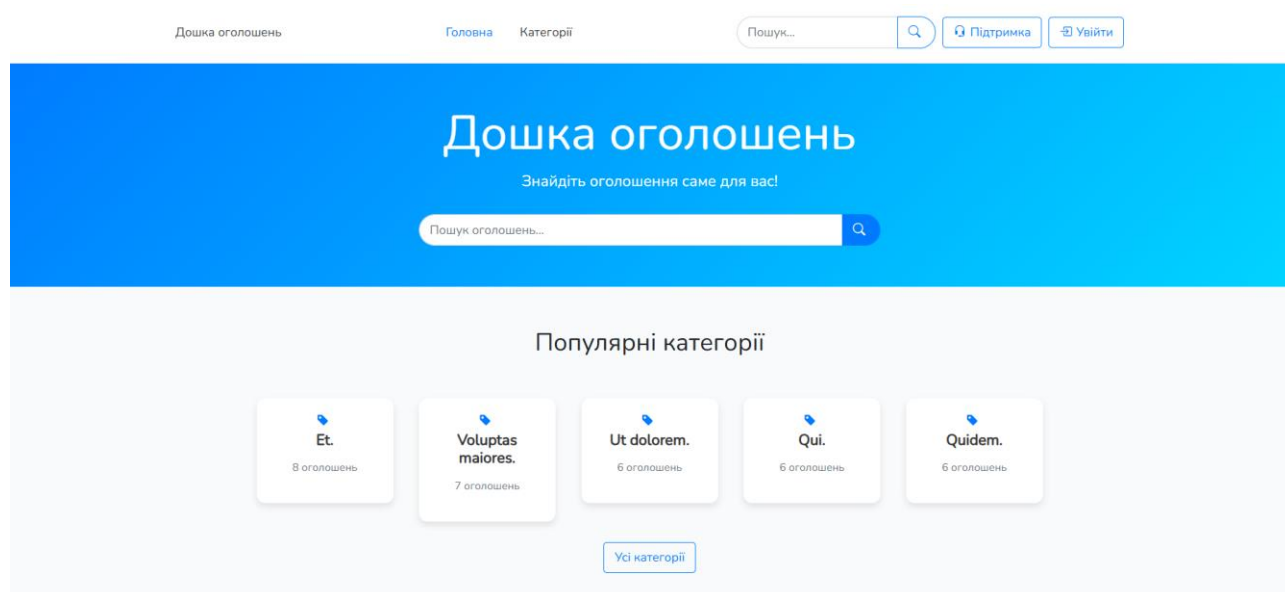


Рис. 3.1. Головна сторінка Дошки оголошень

Користувач, який не авторизований, може переглядати оголошення та використовувати фільтри, але для створення оголошення чи надсилання запиту необхідно зареєструватися (рис. 3.2). Сторінка реєстрації містить форму з полями для введення email, пароля та імені.

Регістрація

Ім'я

Прізвище

Email

Аватарка No file chosen

Телефон

Регіон

Місто

Пароль

Підтвердження паролю

Рис. 3.2. Сторінка реєстрації

Після успішної реєстрації або для наявного облікового запису користувач може увійти в систему (рис. 3.3). Сторінка авторизації містить форму з полями для введення email та пароля. Після авторизації користувач отримує доступ до профілю, де може переглядати свої оголошення, редагувати їх або деактивувати.

Увійти

Email

Пароль

☐ Запам'ятати мене

[Забули пароль?](#)

Рис. 3.3. Сторінка авторизації

Сторінка переліку категорій показує список усіх доступних категорій (наприклад, «Меблі», «Техніка», «Одяг») із кількістю оголошень у кожній і кнопкою для перегляду оголошень конкретної категорії (рис. 3.4).

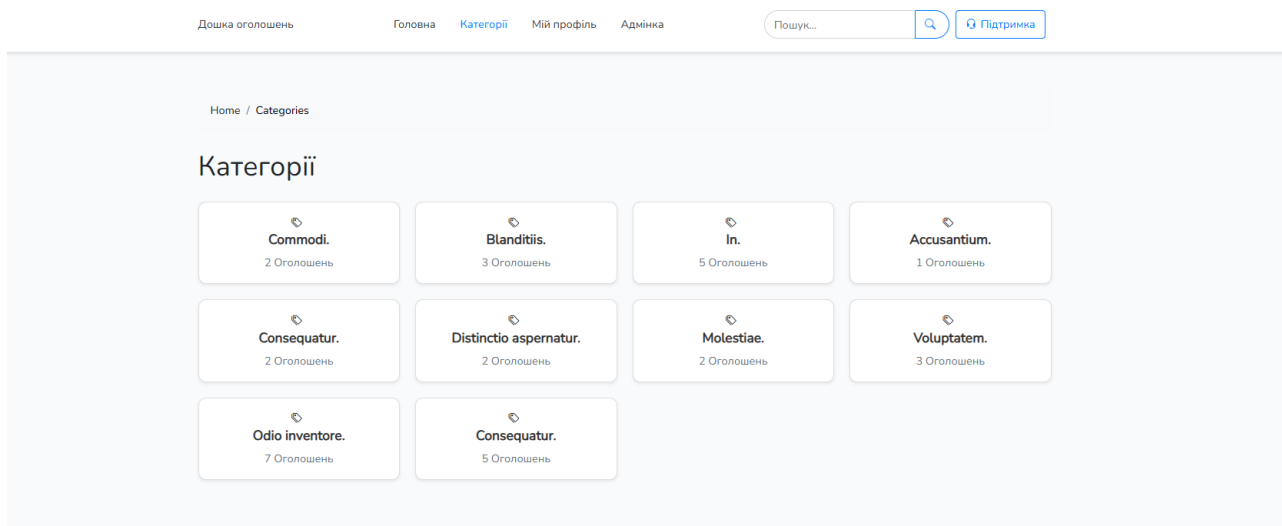


Рис. 3.4. Сторінка лістингу категорій

Сайдбар із фільтрами доступний на сторінках лістингів оголошень. Він містить фільтри за типом оголошення: «Віддаю», «Потребую», категоріями, регіонами, містами, а також пошуком за назвою, кнопки для скидання або застосування фільтрів (рис. 3.5).

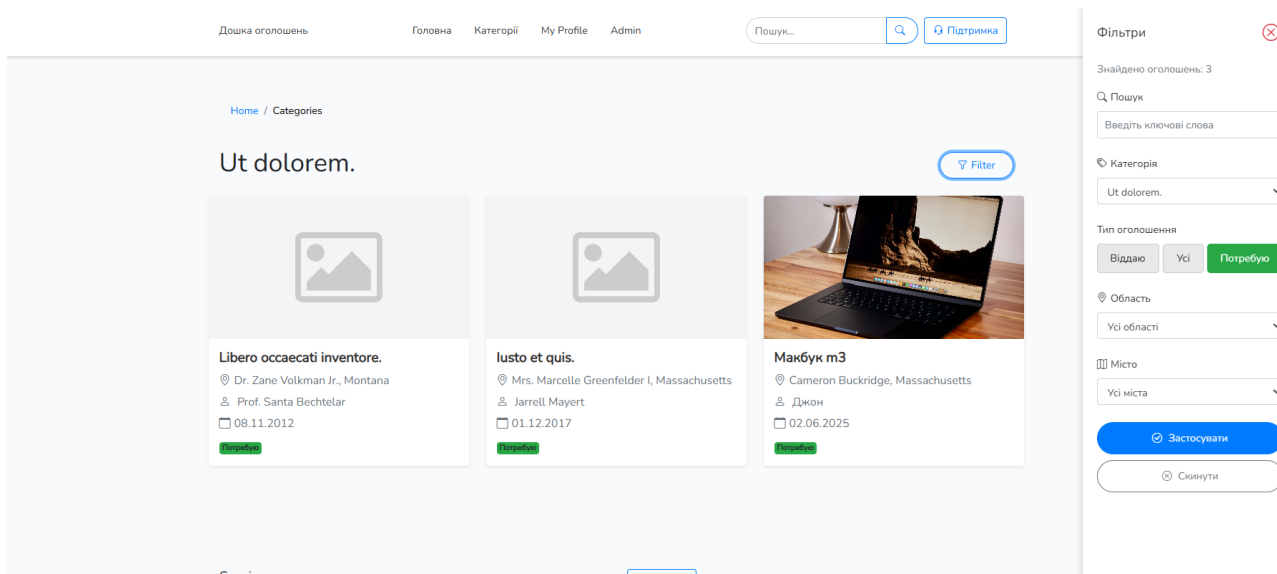


Рис. 3.5. Результат фільтрації

Для звернення на оголошення користувач натискає кнопку «Надіслати запит» на сторінці оголошення (рис. 3.6). Відкривається модальне вікно з полем для введення тексту запиту та кнопкою «Відправити». Після відправлення запит відображається в профілі автора оголошення для схвалення або відхилення. Підтверджені користувачі бачать контакти одразу.

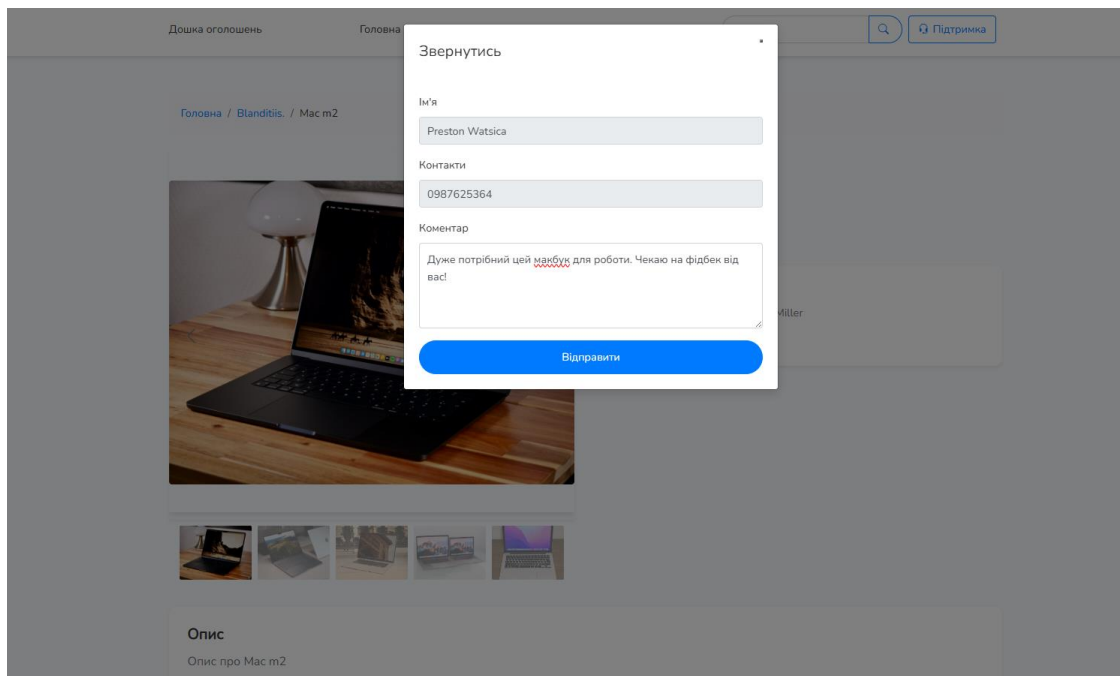


Рис. 3.6. Форма запиту на оголошення

Для створення оголошення користувач має перейти на власний профіль та перемкнутись на вкладку «Створити оголошення», після чого відкривається форма з полями: назва, опис, категорія, тип («Віддаю» чи «Потребую»), регіон, місто, контактна інформація (телефон або email) і можливість завантаження зображень (рис. 3.7). Після заповнення форми оголошення публікується та стає доступним для перегляду іншими користувачами.

Рис. 3.7. Форма створення оголошення

Користувач може переглянути деталі оголошення, натиснувши на нього. На сторінці оголошення відображається вся інформація (назва, опис, категорія, регіон, дата публікації, зображення), а також кнопка надіслання запиту для зв'язку з автором (рис. 3.8). Якщо користувач є підтвердженим (статус, який надає адміністратор), він може одразу побачити контактну інформацію автора. в іншому випадку доведеться чекати фідбеку від автора.

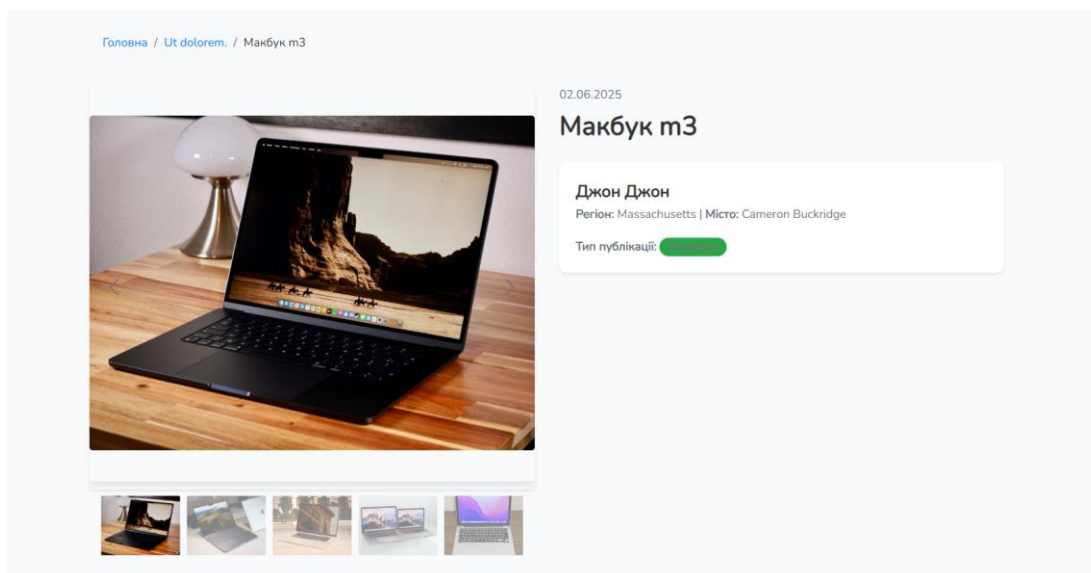


Рис. 3.8. Сторінка оголошення

Підтверджені користувачі мають додаткові привілеї, такі як перегляд контактів інших підтверджених користувачів без запитів. Звичайні користувачі можуть надсилати скарги на оголошення через модальне вікно, доступне на сторінці оголошення, вказавши причину скарги (рис. 3.9).

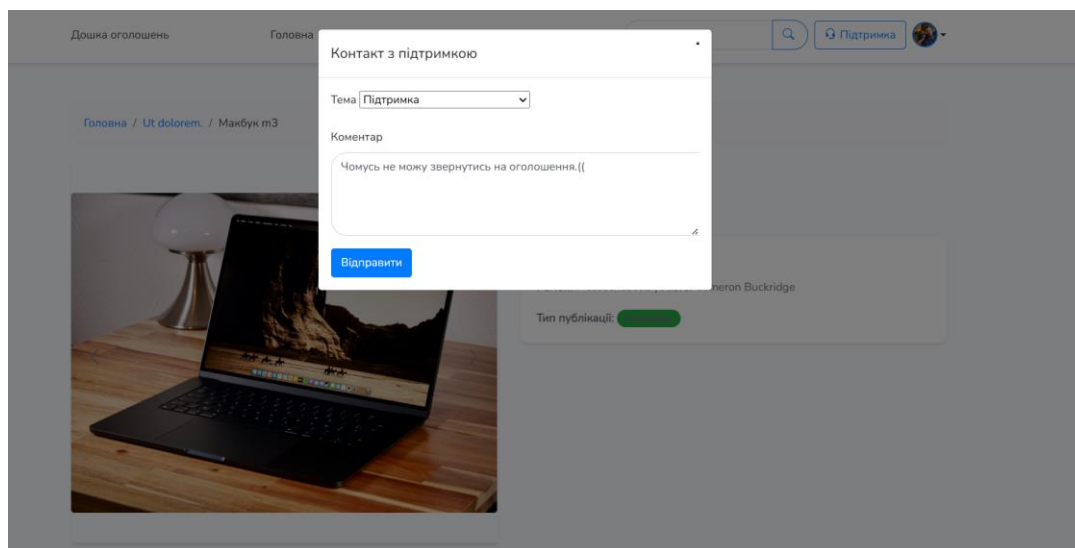


Рис. 3.9. Форма підтримки/скарги

Адміністративна панель, доступна лише користувачам із роллю адміністратора. На головній сторінці адмінпанелі відображається список усіх користувачів із можливістю підтвердження їхнього статусу, список оголошень із функціями редагування чи видалення, а також управління регіонами, статичними сторінками, наприклад «Про нас» або «Контакти». Наприклад, адміністратор може додати новий регіон, відредагувати статичну сторінку, розглянути запит підтримки, тощо (рис. 3.10).

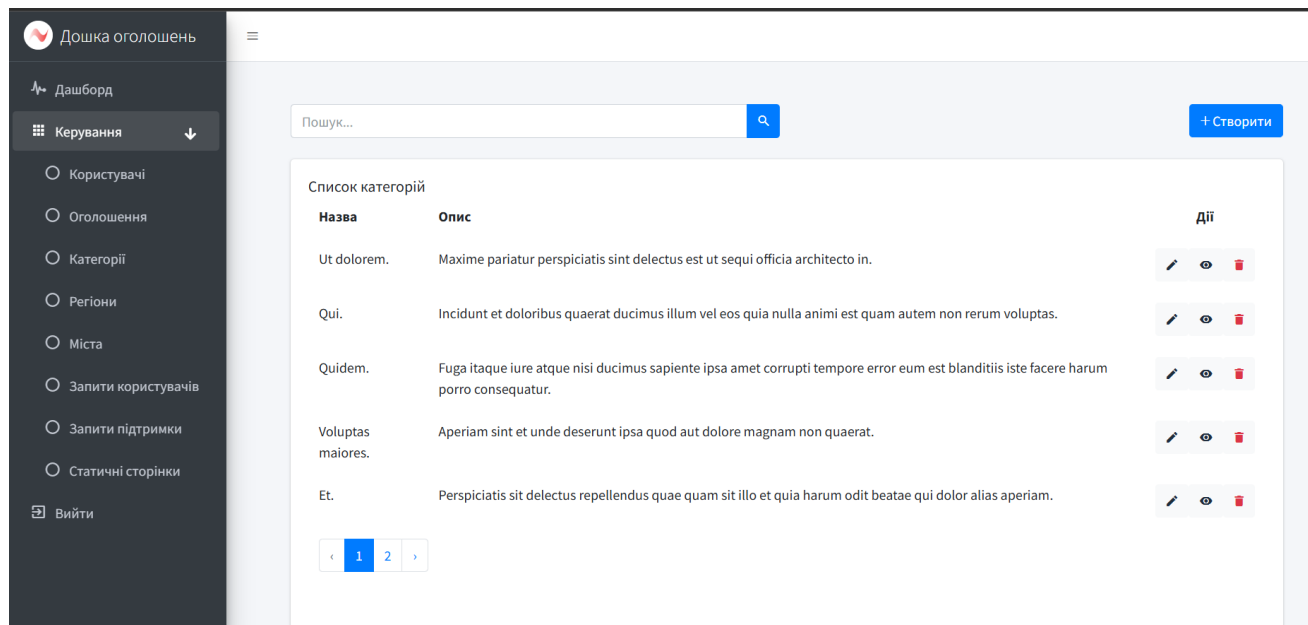


Рис. 3.10. Сторінка адмінпанелі з вкладками для управління

Адміністратор також може переглядати скарги, подані користувачами, і вживати заходів, наприклад, видаляти оголошення. Для цього в адмінпанелі є окрема сторінка зі списком скарг, де відображається інформація про оголошення, користувача, який подав скаргу, і причину (рис. 3.6).

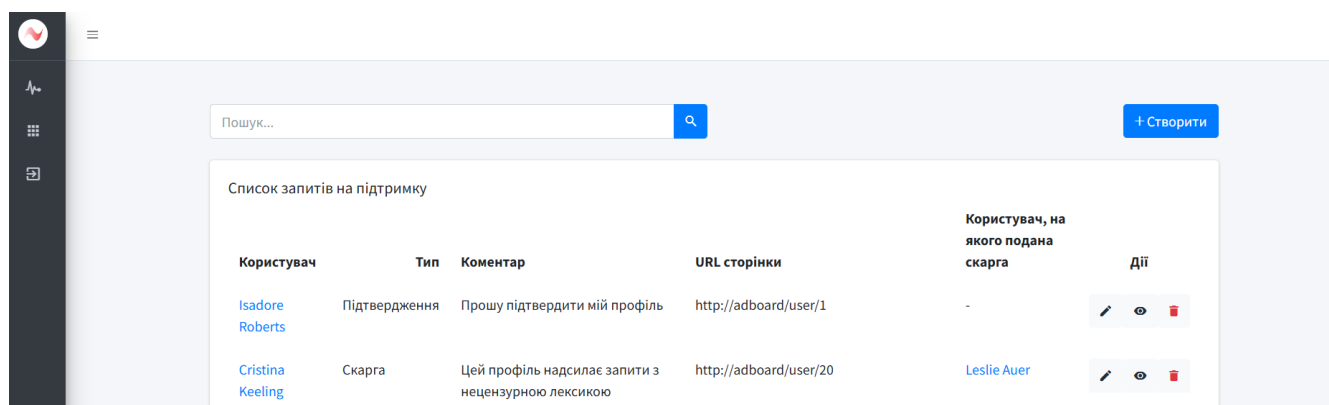


Рис. 3.11. Сторінка адмінки зі списком з запитами підтримки

Статична сторінка «Про нас» доступна у футері та містить інформацію про мету платформи, її переваги, тощо. Статичні сторінки створенні щоб наповнювати їх контентом на будь яку тему (рис. 3.12).

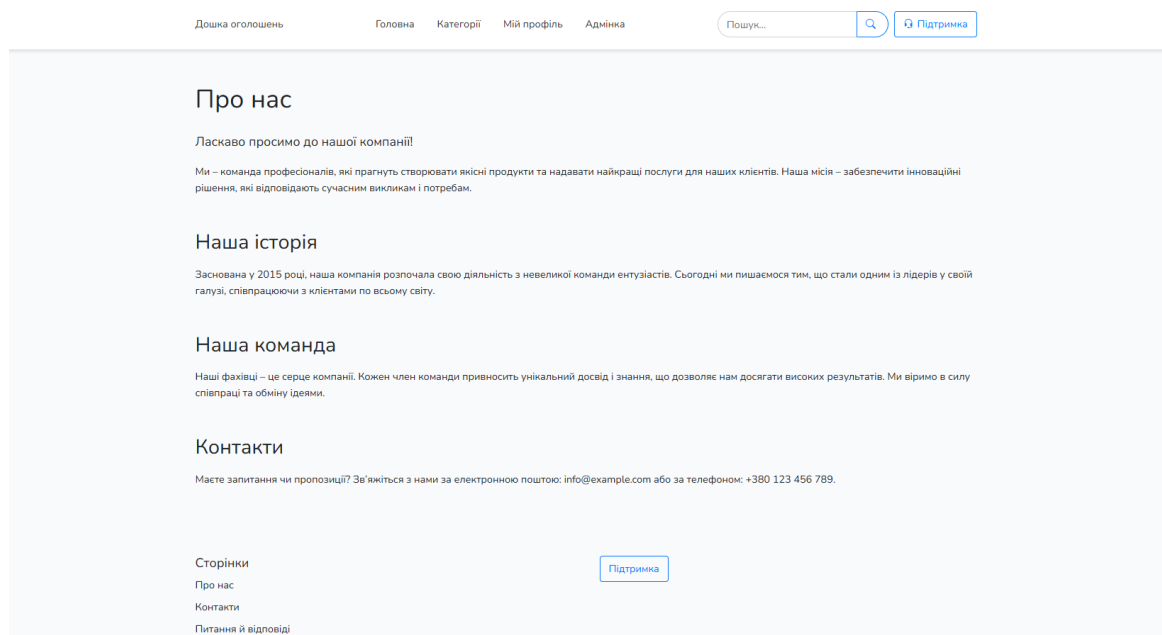


Рис. 3.12. Статична сторінка «Про нас».

Адміністративна панель, доступна через Vemto, дозволяє управляти категоріями. У вкладці «Категорії» адміністратор бачить список існуючих категорій (назва, опис) і кнопку «Додати категорію» (рис. 3.13). При натисканні відкривається форма з полями для введення назви категорії та опису. Після заповнення і натискання кнопки «Зберегти» нова категорія додається до системи і стає доступною для використання.

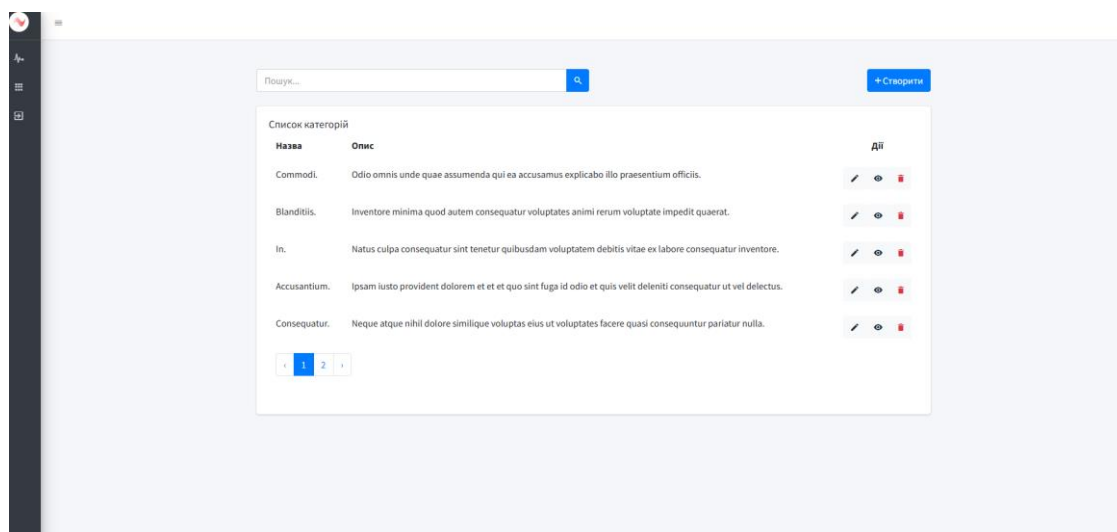


Рис. 3.13. Сторінка “Категорії” в адмінці

Для редагування категорії адміністратор обирає категорію зі списку, натискає кнопку «Редагувати» (іконка олівця), після чого відкривається форма, аналогічна до створення, із заповненими даними (назва, опис) (рис. 3.14). Адміністратор може змінити інформацію та зберегти зміни, після чого оновлена категорія відображається в системі.

Рис. 3.14. Форма редагування категорії в адмінці

Видалення категорії відбувається через кнопку «Видалити» біля кожної категорії у списку. Перед видаленням з'являється підтвердження дії (рис. 3.15). Після підтвердження категорія видаляється

Рис. 3.15. Вікно підтвердження видалення категорії

Інтерфейс системи «Дошка оголошень» розроблений з урахуванням простоти та зручності використання. Адаптивний дизайн, фільтрація оголошень, безпечна комунікація через запити та гнучке адміністрування забезпечують комфортну взаємодію для всіх користувачів, незалежно від їхнього статусу чи пристрою.

3.3. Тестування роботи платформи Дошка оголошень

Тестування системи «Дошка оголошень» проведено шляхом виконання типового сценарію використання, що відображає дії користувача від входу на сайт до звернення до оголошення. Додатково протестовано функції адміністрування для оцінки повноти реалізації системи. Мета – перевірити коректність роботи основних функцій, зручність інтерфейсу та відповідність вимогам.

Користувач заходить на сайт і потрапляє на головну сторінку. На ній відображаються заголовок, поле пошуку, блоки популярних категорій, оголошень типу «Віддаю» та «Потребую» (рис. 3.16).

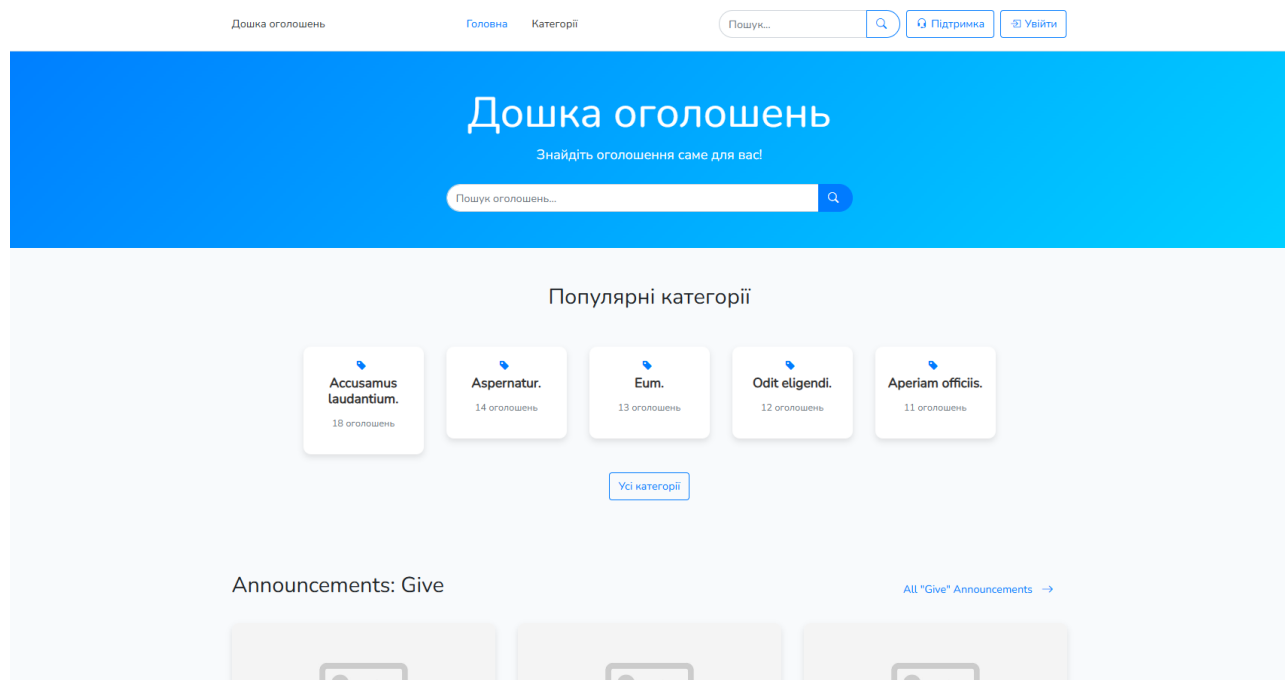


Рис. 3.16. Головна сторінка з усіма блоками: поле пошуку, категорії, оголошення

Користувач натискає на кнопку «Увійти» у панелі навігації та за потреби реєструємо новий аккаунт (рис. 3.17). Відкривається сторінка з формою, де

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						47

вводимо дані. Після натискання кнопки «Зареєструватися» система створює обліковий запис, і користувача перекидає на головну сторінку вже авторизованого. Перевірено, що дані збереглися в базі MySQL, а email унікальний (при спробі повторної реєстрації з тим же email видається помилка).

Рис. 3.17. Сторінка реєстрації з заповненою формою

На головній сторінці користувач вводить у поле пошуку запит «мас» і натискає Enter. Система перенаправляє на сторінку результатів, де відображаються оголошення, що містять слово «мас» у назві чи описі (рис. 3.18). Пошук працює коректно, результати відповідають запиту.

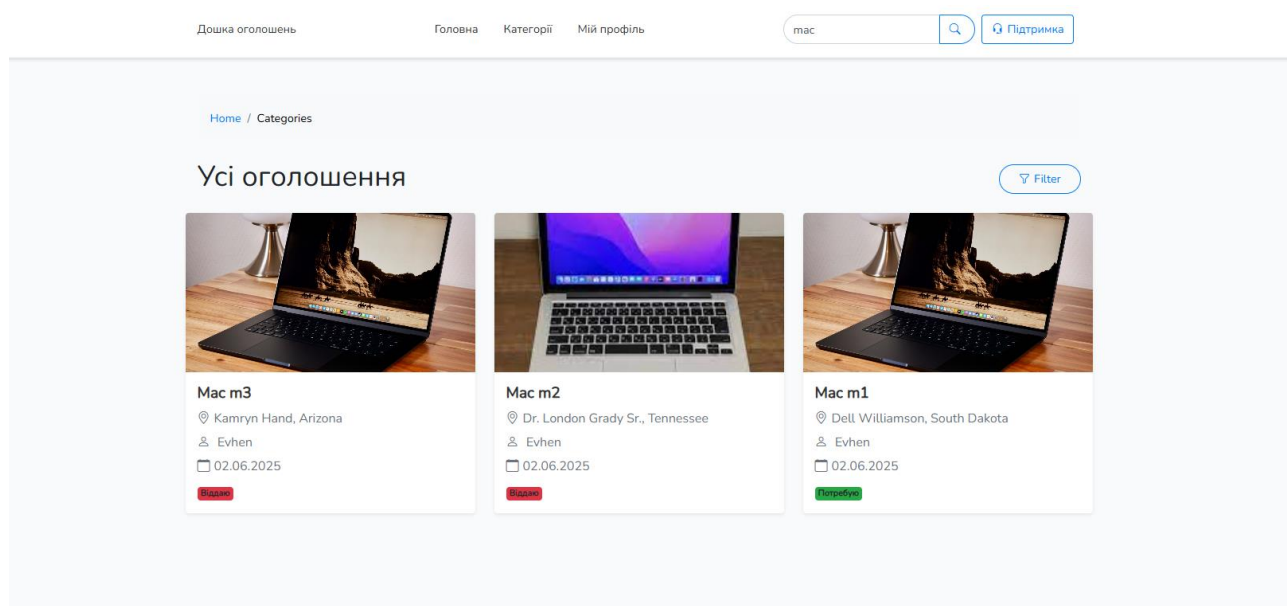


Рис. 3.18. Сторінка результатів із оголошеннями, що містять «мас».

На сторінці результатів користувач відкриває сайдбар із фільтрами, обирає тип «Віддаю» та залишаємо пошуковий запит «mac». Після натискання кнопки «Застосувати» система оновлює список, показуючи лише оголошення, що відповідають критеріям (рис. 3.19). Фільтрація працює коректно.

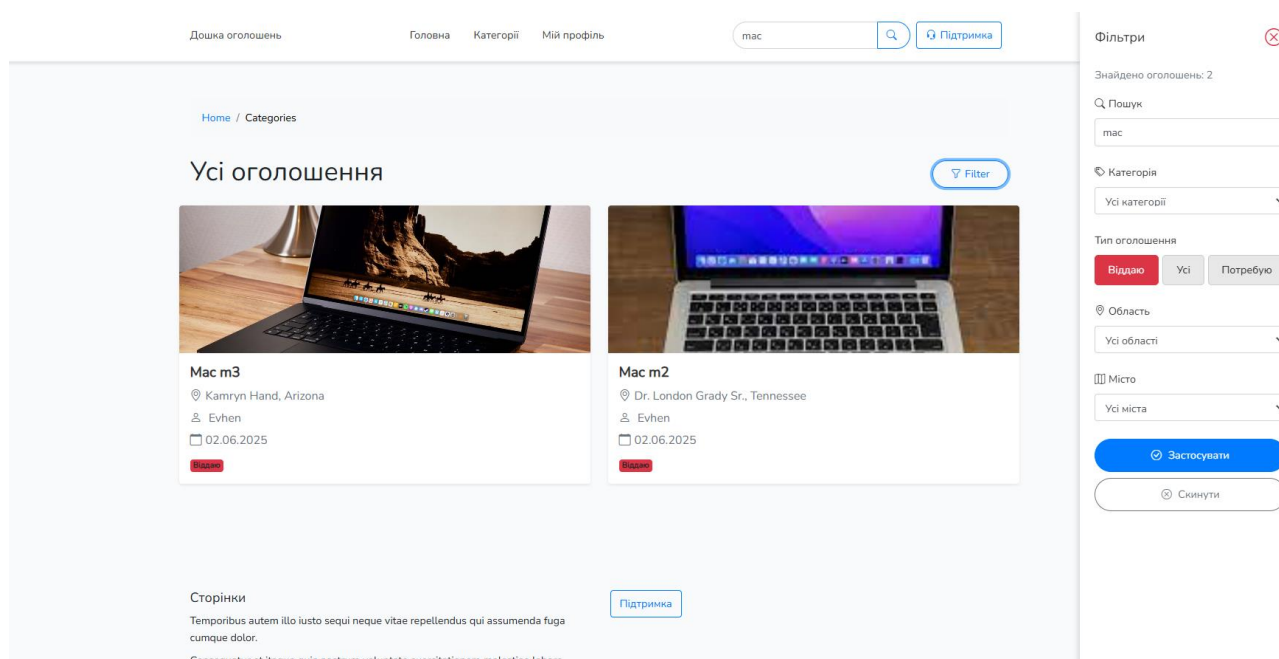


Рис. 3.19. Сторінка результатів із фільтрами

Користувач обирає оголошення і натискає. Відкривається сторінка оголошення з деталями а також кнопка «Звернутись» (рис. 3.20). Усі дані відображаються коректно, зображення завантажуються.

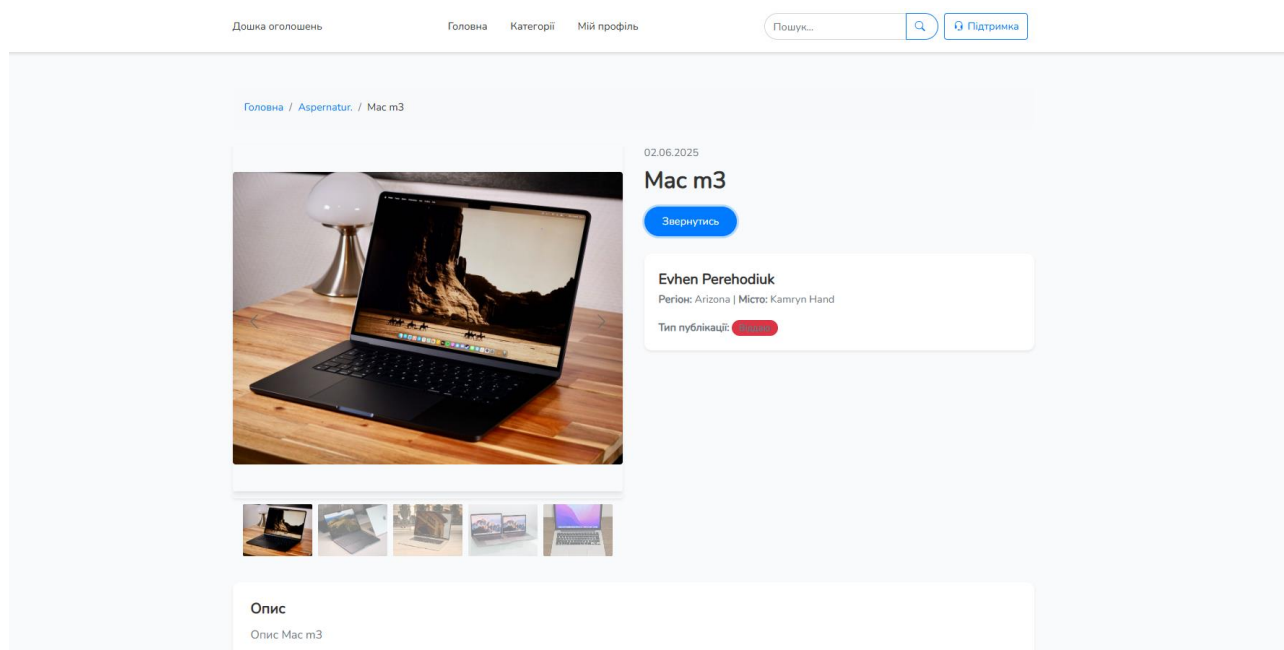


Рис. 3.20. Сторінка оголошення із деталями та кнопкою «Звернутись»

Користувач натискає «Звернутись», відкривається модальне вікно, де він вводить текст («Цікавить стан ноутбука, коли можна забрати?») і натискає «Відправити» (рис. 3.21).

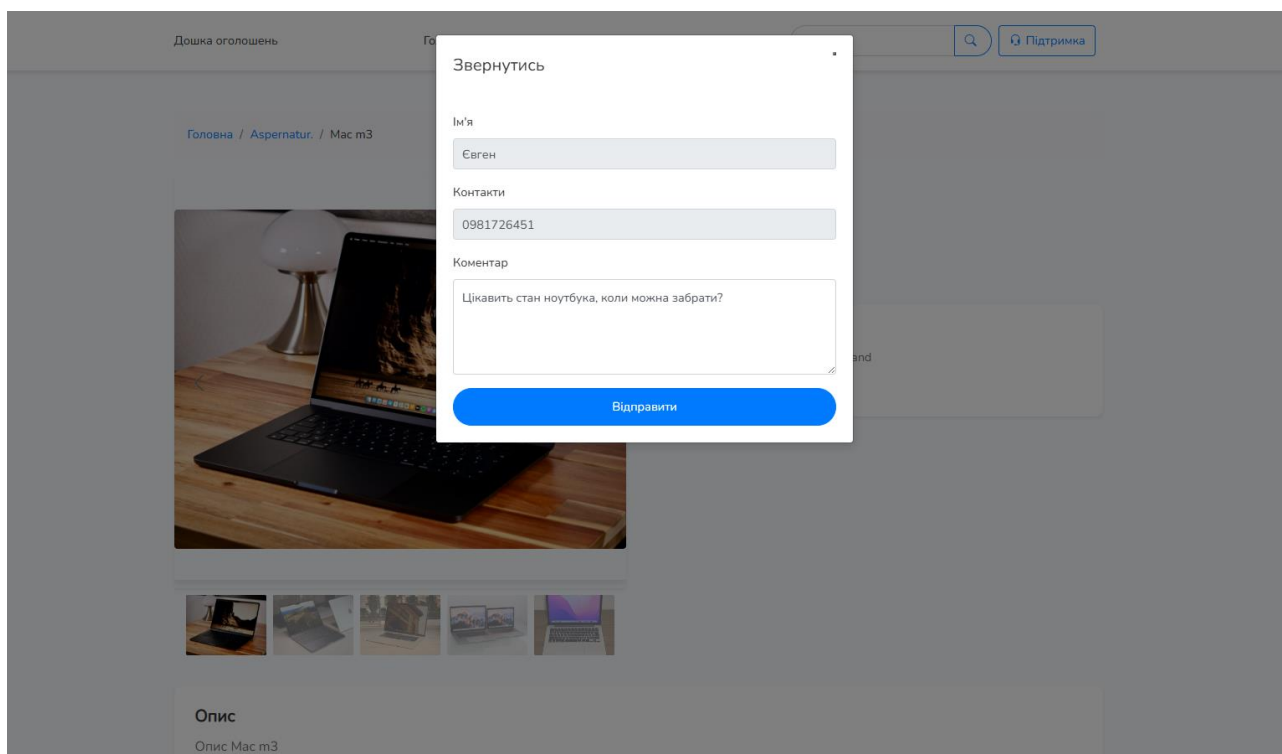


Рис. 3.21. Модальне вікно запиту

Запит надсилається автору оголошення. Для перевірки автор увійшов у профіль та побачив запит та за потреби відповів користувачу (рис. 3.22).

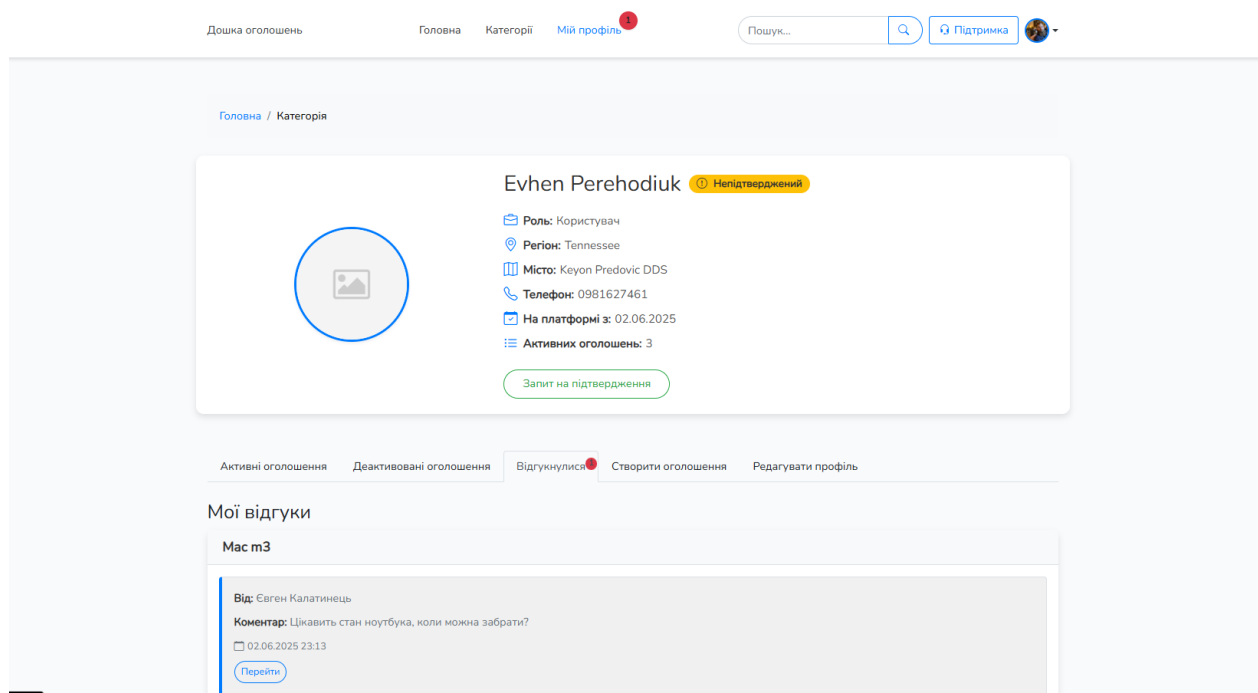


Рис. 3.22. Сторінка автора з відгуком на його оголошення

Висновки до третього розділу

У даному розділі наведено детальну схему розгортання системи «Дошка оголошень», що включає веб-сервер із Laravel, базу даних MySQL, клієнтську частину на Bootstrap і адмінпанель на Vemto, розгорнуті на сервері з Openserver. Описано процес встановлення та налаштування компонентів, а також їхню взаємодію, що дозволило зрозуміти архітектуру системи та зв'язки між її елементами. Налаштування прав доступу через middleware Laravel і забезпечення безпеки за допомогою CSRF-токенів та HTTPS підкреслили надійність платформи.

Далі надано опис інтерфейсу розробленого програмного продукту, де розглянуто функціональні можливості залежно від ролі користувача. Звичайні користувачі можуть створювати, редагувати оголошення та надсилати запити, підтверджені користувачі отримують доступ до контактів без запитів, а адміністратори мають повний контроль над системою, включаючи додавання, редагування та видалення категорій. Такий підхід допомагає кожному користувачу орієнтуватися в системі та використовувати її відповідно до своїх потреб.

Окрім того, продемонстровано тестування роботи системи через сценарії використання – від входу на головну сторінку до звернення до оголошення, а також адміністрування та обробку скарг. Перевірка коректності функцій (реєстрація, пошук, фільтрація, комунікація) підтвердила, що система відповідає вимогам і готова до практичного застосування.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра розроблено веб-платформу «Дошка оголошень». Проведено аналіз аналогів (OLX, Ria.com, Bon, Prom, Shafa), визначено вимоги до системи: простий інтерфейс, верифікація користувачів, безпечна комунікація та підтримка некомерційних оголошень.

Спроектовано об'єктно-орієнтовану структуру системи та реляційну базу даних на MySQL для зберігання даних про користувачів, оголошення, запити та категорії. Реалізовано алгоритми створення, фільтрації оголошень, обробки запитів і адміністративного управління. Забезпечено захист контактів для непідтверджених користувачів.

Описано розгортання платформи на базі Laravel, MySQL, Bootstrap і Vento. Налаштовано безпеку через CSRF-токени та middleware. Інтерфейс адаптовано для різних ролей: звичайні користувачі створюють оголошення, підтверджені отримують доступ до контактів, адміністратори керують системою. Тестування підтвердило коректність функцій: реєстрація, пошук, комунікація.

Архітектура на Laravel і MySQL забезпечує масштабованість і безпеку. Інтуїтивний інтерфейс спрощує взаємодію. Практична цінність — у створенні зручної платформи для некомерційних оголошень. Подальші вдосконалення: аналітика для адміністраторів, розширена фільтрація, система рейтингу користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт “OLX” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.olx.ua/>
2. Офіційний сайт “Ria” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ria.com/>
3. Офіційний сайт “Bon” [Електронний ресурс] – Режим доступу до ресурсу: <https://bon.ua/>
4. Офіційний сайт “Prom” [Електронний ресурс] – Режим доступу до ресурсу: <https://prom.ua/>
5. Офіційний сайт “Shafa” [Електронний ресурс] – Режим доступу до ресурсу: <https://shafa.ua/>
6. Документація Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs>
7. Документація Laravel Eloquent ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/eloquent>
8. Документація Vemto [Електронний ресурс] – Режим доступу до ресурсу: <https://vemto.studio/>
9. Документація Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/>
10. Документація jQuery [Електронний ресурс] – Режим доступу до ресурсу: <https://jquery.com/>
11. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>
12. Документація Laravel Middleware [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/middleware>
13. Документація Laravel CSRF Protection [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/csrf>
14. Документація Openserver [Електронний ресурс] – Режим доступу до ресурсу: <https://ospanel.io/>

15. Документація PHP Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/docs.php>
16. Документація W3C HTML Standards [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/standards/>
17. Документація Laravel Authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/authentication>
18. Документація Laravel Authorization [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/authorization>
19. Документація Bootstrap Modal Components [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/components/modal/>
20. Документація Laravel Blade Templating [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/blade>
21. Документація PHP Secure Coding Practices [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/security.php>
22. Документація MySQL Performance Tuning [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/optimization.html>
23. Документація W3C CSS Standards [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/Style/CSS/>
24. Документація Laravel Query Builder [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/queries>
25. Документація Bootstrap Forms [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/forms/overview/>
26. Документація JavaScript DOM Manipulation [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
27. Документація Laravel File Storage [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/filesystem>

28. Документація MySQL Foreign Keys [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html>

29. Документація Web Accessibility Guidelines [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3.org/WAI/standards-guidelines/>

30. Документація Laravel Validation [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs/validation>

ДОДАТКИ

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						56

Фільтрація оголошень

```

public function filter(Request $request)
{
    $category_id = $request->input('category_id');
    $type = $request->input('type');
    $otherFilters = $request->input('search') || $request->input('region_id') || $request->input('city_id');

    if ($category_id && !$otherFilters && !$type) {
        return redirect()->route('category', ['category' => $category_id]);
    }

    $query = Announcement::query()->where('status', 1);

    if ($search = $request->input('search')) {
        $query->where(function($q) use ($search) {
            $q->where('title', 'like', "%{$search}%")
                ->orWhere('description', 'like', "%{$search}%");
        });
    }

    if ($category_id) {
        $query->where('category_id', $category_id);
    }

    $type = $request->input('type');
    if (in_array($type, ['give', 'need'])) {
        $query->where('type', $type === 'give' ?
\App\Models\Announcement::TYPE_GIVE :
\App\Models\Announcement::TYPE_NEED);
    }

    if ($region_id = $request->input('region_id')) {
        $query->where('region_id', $region_id);
    }

    if ($city_id = $request->input('city_id')) {
        $query->where('city_id', $city_id);
    }

    $announcements = $query->with(['category', 'region', 'city', 'user'])
        ->paginate(12)
        ->appends($request->query());

    $categories = Category::whereHas('announcements', function ($q) {
        $q->where('status', 1);
    })->get();

```

```

$regions = Region::whereHas('announcements', function ($q) {
    $q->where('status', 1);
})->get();

$title = 'Усі оголошення';
if ($type && !$category_id && !$otherFilters) {
    $title = $type === 'give' ? 'Оголошення: Віддаю' :
'Оголошення: Потребую';
} elseif ($category_id) {
    $category = Category::find($category_id);
    $title = $category->title ?? 'Усі оголошення';
}

return view('main.category', compact('announcements',
'categories', 'regions', 'title'));
}

```

Конфігурація маршрутів

<?php

```

use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\UserController;
use App\Http\Controllers\CityController;
use App\Http\Controllers\RegionController;
use App\Http\Controllers\PartnerController;
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\StaticPageController as
AdminStaticPageController;
use App\Http\Controllers\UserRequestController;
use App\Http\Controllers\AnnouncementController;
use App\Http\Controllers\SupportRequestController;

Auth::routes();

Route::middleware(['auth', 'admin'])->group(function () {
    Route::get('/admin', [HomeController::class, 'index'])->
    >name('admin.home');

    Route::prefix('/admin')->group(function () {
        Route::resource('users', UserController::class);
        Route::resource('announcements',
AnnouncementController::class);
        Route::resource('categories', CategoryController::class);
        Route::resource('regions', RegionController::class);
        Route::resource('cities', CityController::class);
        Route::resource('partners', PartnerController::class);
        Route::resource('user-requests',
UserRequestController::class);
        Route::resource('support-requests',
SupportRequestController::class);
        Route::resource('static-pages',
AdminStaticPageController::class);

        Route::get('deactivated-announcements',
[AnnouncementController::class, 'deactivatedAnnouncements'])->
>name('admin.deactivated');
    });
});

```

Управління оголошеннями

```

<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Models\City;
use App\Models\Region;
use App\Models\Category;
use Illuminate\View\View;
use App\Models\Announcement;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;
use Illuminate\Support\Facades\Storage;
use App\Http\Requests\AnnouncementStoreRequest;
use App\Http\Requests\AnnouncementUpdateRequest;

class AnnouncementController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(Request $request): View
    {
        $this->authorize('view-any', Announcement::class);

        $search = (string) $request->get('search', '');

        $announcements = Announcement::search($search)
            ->latest()
            ->paginate(5)
            ->withQueryString();

        $deactivatedAnnouncements = Announcement::where('status',
0)
            ->where('deactivation_reason', '>', 0)
            ->search($search)
            ->latest()
            ->paginate(5, ['*'], 'filtered_page')
            ->withQueryString();

        return view(
            'app.announcements.index',
            compact('announcements', 'search',
'deactivatedAnnouncements')
        );
    }

    /**
     * Show the form for creating a new resource.

```

```

    */
    public function create(Request $request): View
    {
        $this->authorize('create', Announcement::class);

        $users = User::pluck('firstname', 'id');
        $categories = Category::pluck('title', 'id');
        $regions = Region::pluck('name', 'id');
        $cities = City::pluck('name', 'id');

        return view(
            'app.announcements.create',
            compact('users', 'categories', 'regions', 'cities')
        );
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(AnnouncementStoreRequest $request):
    RedirectResponse
    {
        $this->authorize('create', Announcement::class);

        $validated = $request->validated();
        if ($request->hasFile('image')) {
            $validated['image'] = $request->file('image')->store('public');
        }

        $announcement = Announcement::create($validated);

        return redirect()
            ->route('announcements.edit', $announcement)
            ->withSuccess(__('crud.common.created'));
    }

    /**
     * Display the specified resource.
     */
    public function show(Request $request, Announcement
    $announcement): View
    {
        $this->authorize('view', $announcement);

        return view('app.announcements.show',
        compact('announcement'));
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Request $request, Announcement

```

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						61

```

$announcement): View
{
    $this->authorize('update', $announcement);

    $users = User::pluck('firstname', 'id');
    $categories = Category::pluck('title', 'id');
    $regions = Region::pluck('name', 'id');
    $cities = City::pluck('name', 'id');

    return view(
        'app.announcements.edit',
        compact('announcement', 'users', 'categories',
'regions', 'cities')
    );
}

/**
 * Update the specified resource in storage.
 */
public function update(
    AnnouncementUpdateRequest $request,
    Announcement $announcement
): RedirectResponse {
    $this->authorize('update', $announcement);

    $validated = $request->validated();
    if ($request->hasFile('image')) {
        if ($announcement->image) {
            Storage::delete($announcement->image);
        }

        $validated['image'] = $request->file('image')-
>store('public');
    }

    $announcement->update($validated);

    return redirect()
        ->route('announcements.edit', $announcement)
        ->withSuccess(__('crud.common.saved'));
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(
    Request $request,
    Announcement $announcement
): RedirectResponse {
    $this->authorize('delete', $announcement);

    if ($announcement->image) {
        Storage::delete($announcement->image);
    }
}

```

```

    }

    $announcement->delete();

    return redirect()
        ->route('announcements.index')
        ->withSuccess(___('crud.common.removed'));
}

public function deactivatedAnnouncements() {
    $announcements = Announcement::where('status', 0)
        ->where('deactivation_reason', '>', 0);

    return view('admin.deactivated-announcements',
compact('announcements'));
}
}
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class AnnouncementStoreRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     */
    public function rules(): array
    {
        return [
            'user_id' => ['required', 'exists:users,id'],
            'category_id' => ['required', 'exists:categories,id'],
            'title' => ['required', 'max:255', 'string'],
            'description' => ['required', 'max:255', 'string'],
            'image' => ['nullable', 'image', 'max:1024'],
            'publish_date' => ['required', 'date'],
            'type' => ['required', 'numeric'],
            'status' => ['required', 'numeric'],
            'region_id' => ['required', 'exists:regions,id'],
            'city_id' => ['required', 'exists:cities,id'],
        ];
    }
}
}

```

```

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class AnnouncementUpdateRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     */
    public function rules(): array
    {
        return [
            'user_id' => ['required', 'exists:users,id'],
            'category_id' => ['required', 'exists:categories,id'],
            'title' => ['required', 'max:255', 'string'],
            'description' => ['required', 'max:255', 'string'],
            'image' => ['nullable', 'image', 'max:1024'],
            'publish_date' => ['required', 'date'],
            'type' => ['required', 'numeric'],
            'status' => ['required', 'numeric'],
            'region_id' => ['required', 'exists:regions,id'],
            'city_id' => ['required', 'exists:cities,id'],
        ];
    }
}

```

Шаблон адмін-панелі для оголошень

```

@extends('layouts.app')

@section('content')
    <div class="container">
        <!-- Tabs Navigation -->
        <ul class="nav nav-tabs" id="announcementTabs"
role="tablist">
            <li class="nav-item" role="presentation">
                <a class="nav-link active" id="announcements-tab"
data-toggle="tab" href="#announcements" role="tab" aria-
controls="announcements" aria-selected="true">
                    @lang('crud.announcements.index_title')
                </a>
            </li>
            <li class="nav-item" role="presentation">
                <a class="nav-link" id="deactivated-tab" data-
toggle="tab" href="#deactivated" role="tab" aria-
controls="deactivated" aria-selected="false">
                    Deactivated Announcements
                </a>
            </li>
        </ul>

        <!-- Tab Content -->
        <div class="tab-content" id="announcementTabsContent">
            <!-- Main Announcements Tab -->
            <div class="tab-pane fade show active"
id="announcements" role="tabpanel" aria-labelledby="announcements-
tab">
                <div class="searchbar mt-4 mb-4">
                    <div class="row">
                        <div class="col-md-6">
                            <form>
                                <div class="input-group">
                                    <input
                                        id="indexSearch"
                                        type="text"
                                        name="search"
                                        placeholder="{{
__('crud.common.search') }}"
                                        value="{{ $search ?? ''"
                                        class="form-control"
                                        autocomplete="off"
                                    </>
                                    <div class="input-group-
append">
                                        <button type="submit"
class="btn btn-primary">

```

```

search"></i>
                                <i class="icon ion-md-
                                </button>
                                </div>
                                </div>
                                </form>
                                </div>
                                <div class="col-md-6 text-right">
                                    @can('create',
App\Models\Announcement::class)
                                    <a
                                        href="{{
route('announcements.create') }}"
                                        class="btn btn-primary"
                                    >
                                        <i class="icon ion-md-
add"></i> @lang('crud.common.create')
                                    </a>
                                    @endcan
                                </div>
                                </div>
                                </div>
                                <div class="card">
                                    <div class="card-body">
                                        <div style="display: flex; justify-
content: space-between;">
                                            <h4 class="card-title">
@lang('crud.announcements.index_title')
                                            </h4>
                                        </div>

                                        <div class="table-responsive">
                                            <table class="table table-borderless
table-hover">
                                                <thead>
                                                    <tr>
                                                        <th class="text-
left">@lang('crud.announcements.inputs.user_id')</th>
                                                        <th class="text-
left">@lang('crud.announcements.inputs.category_id')</th>
                                                        <th class="text-
left">@lang('crud.announcements.inputs.title')</th>
                                                        <th class="text-
left">@lang('crud.announcements.inputs.description')</th>
                                                        <th class="text-
left">@lang('crud.announcements.inputs.publish_date')</th>
                                                        <th class="text-
right">@lang('crud.announcements.inputs.type')</th>
                                                        <th class="text-
right">@lang('crud.announcements.inputs.status')</th>
                                                        <th class="text-

```

```

left">@lang('crud.announcements.inputs.region_id')</th>
        <th class="text-
left">@lang('crud.announcements.inputs.city_id')</th>
        <th class="text-
center">@lang('crud.common.actions')</th>
    </tr>
</thead>
<tbody>
    @forelse($announcements as
$announcement)
        <tr>
            <td>{{
optional($announcement->user)->firstname ?? '-' }}</td>
            <td>{{
optional($announcement->category)->title ?? '-' }}</td>
            <td>{{ $announcement->title ?? '-' }}</td>
            <td>{{ $announcement->description ?? '-' }}</td>
            <td>{{ $announcement->publish_date ?? '-' }}</td>
            <td>{{ $announcement->type ?? '-' }}</td>
            <td>{{ $announcement->status ?? '-' }}</td>
            <td>{{
optional($announcement->region)->name ?? '-' }}</td>
            <td>{{
optional($announcement->city)->name ?? '-' }}</td>
            <td class="text-center"
style="width: 134px;">
                <div role="group"
aria-label="Row Actions" class="btn-group">
                    @can('update',
$announcement)
                        <a href="{{
route('announcements.edit', $announcement) }}">
                            <button
type="button" class="btn btn-light">
                                <i
class="icon ion-md-create"></i>
                            </button>
                        </a>
                    @endcan
                    @can('view',
$announcement)
                        <a href="{{
route('announcements.show', $announcement) }}">
                            <button
type="button" class="btn btn-light">
                                <i
class="icon ion-md-eye"></i>
                            </button>

```

```

</a>
@endcan
@can('delete',
$announcement)

<form
    action="{{
route('announcements.destroy', $announcement) }}"
method="POST"
onsubmit="return confirm('{{ __('crud.common.are_you_sure') }}')"
>
    @csrf
    @method('DELETE')
    <button
type="submit" class="btn btn-light text-danger">
        <i
class="icon ion-md-trash"></i>
    </button>
</form>
@endcan
</div>
</td>
</tr>
@empty
<tr>
    <td
colspan="11">@lang('crud.common.no_items_found')</td>
</tr>
@endforelse
</tbody>
<tfoot>
<tr>
    <td colspan="11">{!!
$announcements->render() !!}</td>
</tr>
</tfoot>
</table>
</div>
</div>
</div>
</div>

<!-- Deactivated Announcements Tab -->
<div class="tab-pane fade" id="deactivated"
role="tabpanel" aria-labelledby="deactivated-tab">
    <div class="mt-4 mb-4"></div>

    <div class="card">
        <div class="card-body">
            <div style="display: flex; justify-
content: space-between;">
                <h4 class="card-title">Deactivated

```

```

Announcements</h4>
</div>

<div class="table-responsive">
  <table class="table table-borderless
table-hover">
    <thead>
      <tr>
        <th class="text-
left">@lang('crud.announcements.inputs.user_id')</th>
        <th class="text-
left">@lang('crud.announcements.inputs.category_id')</th>
        <th class="text-
left">@lang('crud.announcements.inputs.title')</th>
        <th class="text-
right">@lang('crud.announcements.inputs.type')</th>
        <th class="text-
left">@lang('crud.announcements.inputs.deactivation_reason')</th>
        <th class="text-
center">@lang('crud.common.actions')</th>
      </tr>
    </thead>
    <tbody>
      @forelse($deactivatedAnnouncements
as $announcement)
        <tr>
          <td>{{
optional($announcement->user)->firstname ?? '-' }}</td>
          <td>{{
optional($announcement->category)->title ?? '-' }}</td>
          <td>{{ $announcement->title ?? '-' }}</td>
          <td>{{ $announcement->type
?? '-' }}</td>
          <td>
            @switch($announcement->deactivation_reason)
              @case(0) Не
деактивоване @break
              @case(1) Видалено
@break
              @case(2) Знайшов
@break
              @default - @break
            @endswitch
          </td>
          <td class="text-center">
            <div role="group"
aria-label="Row Actions" class="btn-group">
              @can('update',
$announcement)
                <a href="{{

```

```

route('announcements.edit', $announcement) }}">
type="button" class="btn btn-light">
class="icon ion-md-create"></i>
$announcement)
route('announcements.show', $announcement) }}">
type="button" class="btn btn-light">
class="icon ion-md-eye"></i>
$announcement)
route('announcements.destroy', $announcement) }}"
method="POST"
onsubmit="return confirm('{{ __('crud.common.are_you_sure') }}')"
@method('DELETE')
type="submit" class="btn btn-light text-danger">
class="icon ion-md-trash"></i>
colspan="6">@lang('crud.common.no_items_found')</td>
$deactivatedAnnouncements->appends(['page' =>
Request::get('page')])>render() {!!}</td>

```

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						70

```

</tr>
</tfoot>
</table>
</div>
</div>
</div>
</div>
</div>
</div>
@endsection

```

Управління профілем користувача

```

<?php

namespace App\Http\Controllers\Main;

use App\Http\Controllers\Controller;
use App\Http\Requests\Main\UpdateUserRequest;
use App\Models\Announcement;
use App\Models\Category;
use App\Models\User;
use App\Models\City;
use App\Models\Region;
use App\Models\UserRequest;
use Illuminate\Http\JsonResponse;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;
use Illuminate\Support\Facades\Storage;

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(Request $request): View
    {
        $this->authorize('view-any', User::class);

        $search = (string) $request->get('search', '');

        $users = User::search($search)
            ->latest()
            ->paginate(5)
            ->withQueryString();

        return view('app.users.index', compact('users',
'search'));
    }

    public function user(User $user): View
    {
        // $this->authorize('view', $user);

        $categories = Category::all();
        $regions = Region::withCount('cities')
            ->having('cities_count', '>', 0)
            ->get();
        $cities = City::all();

```

```

$activeAnnouncements = Announcement::with('gallery')
->where('user_id', $user->id)
    ->where('status', 1)
    ->get();

$inactiveAnnouncements = Announcement::with('gallery')
->where('user_id', $user->id)
    ->where('status', 0)
    ->get();

$announcements = auth()->user()->announcements()-
>with('requests.user')->get();

$unreadRequestsCount = 0;
if (Auth::check()) {
    $unreadRequestsCount =
UserRequest::whereIn('announcement_id', Auth::user()-
>announcements->pluck('id'))
    ->where('is_read', false)
    ->count();
}

return view('main.user', compact('user', 'categories',
'regions', 'cities', 'activeAnnouncements',
'inactiveAnnouncements', 'announcements', 'unreadRequestsCount'));
}

public function requests()
{
    $announcements = auth()->user()->announcements()-
>with('requests.user')->get();
    return view('user.profile', compact('announcements'));
}

public function update(UpdateUserRequest $request):
JsonResponse
{
    $user = Auth::user();
    $validated = $request->validated();

    if ($request->hasFile('image')) {
        if ($user->image) {
            Storage::delete($user->image);
        }
        $validated['image'] = $request->file('image')-
>store('public/profiles');
    }

    $user->update($validated);

    $imageUrl = $user->image ? Storage::url($user->image) :
asset('images/default-avatar.png');

```

```

        return response()->json([
            'success' => true,
            'message' => 'Profile successfully updated!',
            'user' => [
                'image' => $imageUrl,
            ],
        ]);
    }

    public function getCities($regionId): JsonResponse
    {
        $region = Region::findOrFail($regionId);
        $cities = $region->cities;

        return response()->json([
            'cities' => $cities->map(function ($city) {
                return [
                    'id' => $city->id,
                    'name' => $city->name,
                ];
            })
        ]);
    }
}

@extends('main.template')

@section('body')
    <section class="py-5">
        <div class="container">
            <!-- Хлібні крихти -->
            <nav aria-label="breadcrumb" class="mb-4">
                <ol class="breadcrumb bg-light p-3 rounded">
                    <li class="breadcrumb-item"><a href="{{
route('home') }}" class="text-primary">Головна</a></li>
                    <li class="breadcrumb-item active" aria-
current="page">Категорія</li>
                </ol>
            </nav>

            <!-- Інформація користувача -->
            <div class="row mb-5 align-items-center user-info">
                <div class="col-md-4 text-center text-md-start">
                    <div class="avatar-container">
                        
                    </div>
                </div>
                <div class="col-md-8">

```

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						74

```

        <div class="d-flex align-items-center mb-3">
            <h2 class="mb-0 fw-bold username-text">{{
$user->firstname }} {{ $user->lastname }}</h2>
            @if ($user->is_confirmed)
                <span class="badge bg-success ms-3"><i
class="bi bi-check-circle me-1"></i> Підтверджений</span>
            @else
                <span class="badge bg-warning ms-3"><i
class="bi bi-exclamation-circle me-1"></i> Непідтверджений</span>
            @endif
        </div>
        <p class="mb-1"><i class="bi bi-briefcase me-
2"></i><strong>Роль:</strong> {{ $user->role == 1 ? 'Продавець' :
'Користувач' }}</p>
        <p class="mb-1"><i class="bi bi-geo-alt me-
2"></i><strong>Регіон:</strong> {{ $user->region->name }}</p>
        <p class="mb-1"><i class="bi bi-map me-
2"></i><strong>Місто:</strong> {{ $user->city->name }}</p>
        <p class="mb-1"><i class="bi bi-telephone me-
2"></i><strong>Телефон:</strong> {{ $user->phone ?? 'Не вказано'
}}</p>
        <p class="mb-1"><i class="bi bi-calendar-check
me-2"></i><strong>На платформі з:</strong> {{ $user->created_at-
>format('d.m.Y') }}</p>
        <p class="mb-1"><i class="bi bi-list-ul me-
2"></i><strong>Активних оголошень:</strong> {{ $user-
>announcements()->where('status', 1)->count() }}</p>
        @if (auth()->check())
            @if (auth()->id() === $user->id && !$user-
>is_confirmed)
                <button class="btn btn-outline-success
rounded-pill px-4 mt-3"
                    data-bs-toggle="modal"
                    data-bs-target="#supportModal"
                    data-type="2">
                    Запит на підтвердження
                </button>
            @elseif (auth()->id() !== $user->id)
                <button class="btn btn-outline-danger
rounded-pill px-4 mt-3"
                    data-bs-toggle="modal"
                    data-bs-target="#supportModal"
                    data-type="1"
                    data-complaint-user-id="{{
$user->id }}">
                    Скарга
                </button>
            @endif
        @endif
    </div>
</div>

<!-- Вкладки -->

```

					ДУ «Житомирська політехніка».25.121.18.000 - ПЗ	Арк.
						75

```

        <ul class="nav nav-tabs mb-4" id="profileTab"
role="tablist">
            <li class="nav-item">
                <a class="nav-link active"
id="userActiveAnnouncements-tab" data-bs-toggle="tab"
href="#userActiveAnnouncements" role="tab" aria-
controls="userActiveAnnouncements" aria-selected="true">Активні
оголошення</a>
            </li>
            @if (auth()->id() === $user->id)
                <li class="nav-item">
                    <a class="nav-link"
id="userInactiveAnnouncements-tab" data-bs-toggle="tab"
href="#userInactiveAnnouncements" role="tab" aria-
controls="userInactiveAnnouncements" aria-
selected="false">Деактивовані оголошення</a>
                </li>
            @endif
            <li class="nav-item position-relative">
                <a class="nav-link {{ request()-
>routeIs('user.requests') ? 'active' : '' }}" id="userRequests-
tab" data-bs-toggle="tab" href="#userRequests" role="tab" aria-
controls="userRequests" aria-selected="false">
                    Відгукнулися
                    @if ($unreadRequestsCount > 0)
                        <span class="badge bg-danger rounded-
circle position-absolute top-0 end-0">{{ $unreadRequestsCount
}}</span>
                    @endif
                </a>
            </li>
            @if (auth()->id() === $user->id)
                <li class="nav-item">
                    <a class="nav-link"
id="createAnnouncement-tab" data-bs-toggle="tab"
href="#createAnnouncement" role="tab" aria-
controls="createAnnouncement" aria-selected="false">Створити
оголошення</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" id="editProfile-tab"
data-bs-toggle="tab" href="#editProfile" role="tab" aria-
controls="editProfile" aria-selected="false">Редагувати
профіль</a>
                </li>
            @endif
        </ul>

        <!-- Вміст вкладок -->
        <div class="tab-content" id="profileTabContent">
            @include('main.template-parts.user-active-
announcements')
            @include('main.template-parts.user-inactive-

```

```

announcements')
    @include('main.template-parts.user-requests')
    @include('main.template-parts.edit-profile')
    @include('main.template-parts.create-
announcement')
    </div>
  </div>
</section>

<script>
  window.userUpdateRoute = '{{ route('user.update') }}';
  window.announcementStoreRoute = '{{
route('announcement.store') }}';
  window.announcementUpdateRoute = '{{
route('announcement.update') }}';
  window.csrfToken = '{{ csrf_token() }}';
</script>
@endsection
<div class="tab-pane fade show active"
id="userActiveAnnouncements" role="tabpanel" aria-
labelledby="userActiveAnnouncements-tab">
  <h4 class="mb-4 fw-bold">{{ $user->firstname }} {{ $user-
>lastname }} Active Announcements</h4>
  <div class="row g-4">
    @forelse($activeAnnouncements as $announcement)
      @include('main.template-parts.announcement-card')
    @empty
      <p class="text-muted">No active announcements
found.</p>
    @endforelse
  </div>
</div>

@include('main.template-parts.edit-announcement')
<div class="tab-pane fade {{ request()->routeIs('user.requests') ?
'show active' : '' }}" id="userRequests" role="tabpanel" aria-
labelledby="userRequests-tab">
  <h3>Мої відгуки</h3>
  @if ($announcements->isEmpty())
    <p class="text-muted">У вас немає оголошень із
відгуками.</p>
  @else
    @foreach ($announcements as $announcement)
      <div class="card request-card mb-3">
        <div class="card-header">
          <h5 class="card-title mb-0">
            <a href="{{ route('announcement',
$announcement->id) }}">{{ $announcement->title }}</a>
          </h5>
        </div>
        <div class="card-body">
          @if ($announcement->requests->isEmpty())
            <p class="text-muted">Немає відгуків для

```

```

цього оголошення.</p>
        @else
            @foreach ($announcement->requests as
$request)
                <div class="request-item {{ $request-
>is_read ? '' : 'unread' }} mb-3 p-3 rounded">
                    <p><strong>Бід:</strong> {{
$request->user->firstname }} {{ $request->user->lastname }}</p>
                    <p><strong>Коментар:</strong> {{
$request->comment }}</p>
                    <p><small><i class="bi bi-calendar
me-2"></i>{{ $request->created_at->format('d.m.Y H:i')
}}</small></p>
                    @if (!$request->is_read)
                        <a href="{{
route('announcement', $announcement->id) }}"
                            class="btn btn-sm btn-
outline-primary mark-as-read"
                            data-request-id="{{
$request->id }}"
                            data-url="{{
route('request.markAsRead') }}">Перейти</a>
                    @endif
                </div>
            @endforeach
        @endif
    </div>
</div>
<div class="tab-pane fade" id="editProfile" role="tabpanel" aria-
labelledby="editProfile-tab">
    <div class="card shadow-sm rounded p-4">
        <h4 class="mb-4 fw-bold">Edit Profile</h4>
        <form id="profileUpdateForm" enctype="multipart/form-
data">
            @csrf
            @method('PUT')

            <!-- Ім'я -->
            <div class="mb-4">
                <label for="firstName" class="form-label"><i
class="bi bi-person me-2"></i>Firstname</label>
                <input type="text" class="form-control rounded
@error('firstname') is-invalid @enderror" id="firstName"
name="firstname" value="{{ old('firstname', $user->firstname) }}">
                <div class="invalid-feedback" id="firstname-
error"></div>
            </div>

            <!-- Прізвище -->
            <div class="mb-4">

```

```

        <label for="lastName" class="form-label"><i
class="bi bi-person me-2"></i>Lastname</label>
        <input type="text" class="form-control rounded
@error('lastname') is-invalid @enderror" id="lastName"
name="lastname" value="{{ old('lastname', $user->lastname) }}">
        <div class="invalid-feedback" id="lastname-
error"></div>
    </div>

    <!-- Email -->
    <div class="mb-4">
        <label for="email" class="form-label"><i class="bi
bi-envelope me-2"></i>Email</label>
        <input type="email" class="form-control rounded
@error('email') is-invalid @enderror" id="email" name="email"
value="{{ old('email', $user->email) }}">
        <div class="invalid-feedback" id="email-
error"></div>
    </div>

    <!-- Телефон -->
    <div class="mb-4">
        <label for="phone" class="form-label"><i class="bi
bi-telephone me-2"></i>Phone</label>
        <input type="text" class="form-control rounded
@error('phone') is-invalid @enderror" id="phone" name="phone"
value="{{ old('phone', $user->phone) }}">
        <div class="invalid-feedback" id="phone-
error"></div>
    </div>

    <!-- Зображення профілю -->
    <div class="mb-4">
        <label for="image" class="form-label"><i class="bi
bi-image me-2"></i>Profile Image</label>
        <input type="file" class="form-control rounded
@error('image') is-invalid @enderror" id="image" name="image"
accept="image/*">
        <div class="invalid-feedback" id="image-
error"></div>
    </div>

    <!-- Region i місто -->
    <div class="row">
        <div class="col-md-6 mb-4">
            <label for="region" class="form-label"><i
class="bi bi-geo-alt me-2"></i>Region</label>
            <select name="region_id" data-form-id="edit-
profile" class="form-control rounded region @error('region_id')
is-invalid @enderror">
                <option value="">Select Region</option>
                @foreach($regions as $region)
                    <option value="{{ $region->id }}" {{

```

```

old('region_id', $user->region_id) == $region->id ? 'selected' :
'' }}>
        {{ $region->name }}
    </option>
    @endforeach
</select>
<div class="invalid-feedback" id="region_id-
error"></div>
</div>
<div class="col-md-6 mb-4">
    <label for="city" class="form-label"><i
class="bi bi-map me-2"></i>City</label>
    <select name="city_id" data-form-id="edit-
profile" class="form-control rounded city @error('city_id') is-
invalid @enderror">
        <option value="">Select City</option>
        @foreach($user->region->cities as $city)
            <option value="{{ $city->id }}" {{
old('city_id', $user->city_id) == $city->id ? 'selected' : '' }}>
                {{ $city->name }}
            </option>
        @endforeach
    </select>
    <div class="invalid-feedback" id="city_id-
error"></div>
</div>
</div>

    <button type="submit" class="btn btn-primary rounded-
pill px-4" id="submitButton">
        <i class="bi bi-check-circle me-2"></i>Save
Changes
    </button>
    <div class="mt-3 alert alert-success d-none rounded"
id="successMessage"></div>
</form>
</div>
</div>
<div class="tab-pane fade" id="createAnnouncement" role="tabpanel"
aria-labelledby="createAnnouncement-tab">
    <div class="card shadow-sm rounded p-4">
        <h4 class="mb-4 fw-bold">Створити оголошення</h4>
        <form id="createAnnouncementForm" enctype="multipart/form-
data" method="POST" action="{{ route('announcement.store') }}">
            @csrf

            <!-- Категорія -->
            <div class="mb-4">
                <label for="category_id" class="form-label"><i
class="bi bi-tags me-2"></i>Категорія</label>
                <select name="category_id" id="category_id"
class="form-control rounded @error('category_id') is-invalid
@enderror">

```

```

        <option value="">Виберіть категорію</option>
        @foreach($categories as $category)
            <option value="{{ $category->id }}" {{
old('category_id') == $category->id ? 'selected' : '' }}>
                {{ $category->title }}
            </option>
        @endforeach
    </select>
    @error('category_id')
    <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>

<!-- Заголовок -->
<div class="mb-4">
    <label for="title" class="form-label"><i class="bi
bi-type me-2"></i>Заголовок</label>
    <input type="text" class="form-control rounded
@error('title') is-invalid @enderror" id="title" name="title"
value="{{ old('title') }}" placeholder="Введіть заголовок
оголошення">
    @error('title')
    <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>

<!-- Опис -->
<div class="mb-4">
    <label for="description" class="form-label"><i
class="bi bi-text-paragraph me-2"></i>Опис</label>
    <textarea class="form-control rounded
@error('description') is-invalid @enderror" name="description"
id="description" rows="5" placeholder="Введіть опис оголошення">{{
old('description') }}</textarea>
    @error('description')
    <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>

<!-- Галерея -->
<div class="mb-4">
    <label for="images" class="form-label"><i
class="bi bi-camera me-2"></i>Фотографії</label>
    <div class="image-upload-container
@error('images.*') is-invalid @enderror">
        <div class="image-upload-dropzone"
id="dropzone">
            <div class="upload-placeholder">
                <i class="bi bi-cloud-upload fs-1 mb-
2"></i>
                <p>Перетягніть фото сюди або натисніть
для вибору</p>
                <small class="text-muted">Максимум 5

```

```

фото, до 5MB кожне (JPG, PNG)</small>
        </div>
        <input type="file" name="images[]"
id="imageInput" multiple accept="image/jpeg,image/png" class="d-
none">
        </div>
        <div class="image-preview-container mt-3 d-
flex flex-wrap gap-2" id="imagePreview"></div>
        </div>
        @error('images.*')
        <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>

    <!-- Тип -->
    <div class="mb-4">
        <label for="type" class="form-label"><i class="bi
bi-tag me-2"></i>Тип</label>
        <select name="type" id="type" class="form-control
rounded @error('type') is-invalid @enderror" required>
            @foreach (\App\Models\Announcement::getTypes()
as $value => $label)
                <option value="{{ $value }}" {{
old('type') == $value ? 'selected' : '' }}>
                    {{ $label }}
                </option>
            @endforeach
        </select>
        @error('type')
        <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>

    <!-- Періон і місто -->
    <div class="row">
        <div class="col-md-6 mb-4">
            <label for="region" class="form-label"><i
class="bi bi-geo-alt me-2"></i>Періон</label>
            <select name="region_id" data-form-id="create-
announcement" class="form-control rounded region
@error('region_id') is-invalid @enderror">
                <option value="">Виберіть періон</option>
                @foreach($regions as $region)
                    <option value="{{ $region->id }}" {{
old('region_id') == $region->id ? 'selected' : '' }}>
                        {{ $region->name }}
                    </option>
                @endforeach
            </select>
            @error('region_id')
            <div class="invalid-feedback">{{ $message
}}</div>
            @enderror

```

```

        </div>
        <div class="col-md-6 mb-4">
            <label for="city" class="form-label"><i
class="bi bi-map me-2"></i>Місто</label>
            <select name="city_id" data-form-id="create-
announcement" class="form-control rounded city @error('city_id')
is-invalid @enderror">
                <option value="">Виберіть місто</option>
            </select>
            @error('city_id')
            <div class="invalid-feedback">{{ $message
}}</div>
            @enderror
        </div>
    </div>

    <button type="submit" class="btn btn-success rounded-
pill px-4" id="submitButton">
        <i class="bi bi-plus-circle me-2"></i>Створити
оголошення
    </button>
    <div class="mt-3 alert alert-success d-none rounded"
id="successMessage"></div>
</form>
</div>
</div>

```