

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:

**«Платформа онлайн-оголошень з елементами
аукціону»**

Виконала студентка групи ІПЗ-20-1
БОНДАРЧУК Олександра Андріївна

Керівник роботи:
КОНЦЕДАЙЛО Валерій Валерійович

Консультант:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

Житомир – 2024

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
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БОНДАРЧУК Олександра Андріївна**

Керівник роботи: **КОНЦЕДАЙЛО Валерій Валерійович**

Тема роботи: **«Платформа онлайн-оголошень з елементами аукціону»**,
затверджена Наказом закладу вищої освіти від «23» лютого 2024 р. №74с

Вихідні дані для роботи: **розробка платформи онлайн-оголошень с елементами аукціону.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Концедайло В.В., Чижмотря О.В.	27.02.2024р.	27.02.2024р.
2	Концедайло В.В., Чижмотря О.В.	27.02.2024р.	27.02.2024р.
3	Концедайло В.В., Чижмотря О.В.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмної реалізації платформи онлайн-оголошень з елементами аукціону та пояснювальної записки. Пояснювальна записка до дипломної роботи містить 66 сторінок, 28 ілюстрацій, 7 таблиць, 1 додаток та 30 літературних джерел.

Метою цієї роботи є розробка WEB-орієнтованої платформи онлайн-оголошень з можливістю проводити аукціони, що дозволить користувачам проводити онлайн торги.

В роботі наявний опис аналізу ТЗ та вимог до системи, проведено порівняльний аналіз з існуючими аналогами. Описано процес реалізації бази даних та розробки програмної платформи онлайн-оголошень з елементами аукціону. В кінці зазначено порядок встановлення платформи, описано інтерфейс та надано інструкції до роботи з інтерфейсом.

КЛЮЧОВІ СЛОВА: АУКЦІОН, СЕРВЕР, REACT, КЛІЄНТ, API, NODE.JS

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Бондарчук О.А.			«Платформа онлайн-оголошень з елементами аукціону» Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		Концедайло В.В.					3	66
Рецензент		Вакалюк Т.А.				Житомирська політехніка, група ІПЗ-20-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

Bachelor's qualification thesis consists of the software implementation of an online advertisement service with auction elements and an explanatory note. The explanatory note to the diploma thesis contains 66 pages, 28 illustrations, 7 tables, 1 appendix, and 30 literature sources.

The purpose of this work is to develop a web-oriented online advertisement service with the ability to conduct auctions, which will allow users to hold online auctions.

The work includes a description of the analysis of technical specifications and system requirements, a comparative analysis with existing analogs, and a description of the process of database implementation and the development of the online advertisement service with auction elements. At the end, the installation procedure of the service is provided, the interface is described, and instructions for working with the interface are given.

KEYWORDS: AUCTION, SERVER, REACT, CLIENT, API, NODE.JS

		<i>Бондарчук О.А.</i>			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		<i>Кониедайло В.В</i>				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПЛАТФОРМИ	8
1.1 Технічне завдання на розробку платформи.....	8
1.2 Аналіз існуючого програмного забезпечення за тематикою проекту	11
1.3 Аналіз вимог системи та користувачів платформи онлайн-оголошень з елементами аукціону.....	15
1.4 Обґрунтування вибору інструментальних засобів та архітектури платформи онлайн-оголошень з елементами аукціону.....	16
Висновки до першого розділу.....	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ОНЛАЙН-ОГОЛОШЕНЬ З ЕЛЕМЕНТАМИ АУКЦІОНУ	21
2.1 Опис акторів та алгоритмів роботи	21
2.2 Розробка бази даних.....	25
2.3 Комунікації і послідовність взаємодії об'єктів платформи.....	29
2.4 Реалізація платформи онлайн-оголошень з елементами аукціону	32
Висновки до другого розділу	35
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ОНЛАЙН-ОГОЛОШЕНЬ З ЕЛЕМЕНТАМИ АУКЦІОНУ	36
3.1 Порядок встановлення та налаштування платформи	36
3.2 Опис інтерфейсу програмного продукту	38
3.3 Тестування платформи	44
Висновки до третього розділу.....	46
ВИСНОВКИ.....	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних;

ОС – операційна система

ПЗ – Програмне забезпечення

СКБД – Система керування базами даних

ТЗ – технічне завдання

API – application programming interface

JS – Java Script

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

За останні десятиліття інтернет та віртуальний простір стали важливими платформами для здійснення різноманітних трансакцій та обміну інформацією. Одним із найбільш поширених віртуальних інструментів, що значно змінив спосіб комунікації та торгівлі, є платформи онлайн-оголошень. Завдяки їм, користувачі можуть легко знаходити товари та послуги, а підприємці знаходять ефективний спосіб реклами своїх пропозицій.

Метою цієї роботи є розробка WEB-орієнтованої платформи онлайн-оголошень з можливістю проводити аукціони.

Встановлена мета обумовлює наступні завдання:

- Розробка технічного завдання на розробку системи.
- Аналіз існуючого програмного забезпечення.
- Аналіз вимог системи та користувача.
- Проектування платформи.
- Реалізація системи.
- Опис інтерфейсу.
- Тестування платформи.
- Налаштування проекту для встановлення.

Об'єктом дослідження є WEB-орієнтована платформа онлайн-оголошень з елементами аукціону, що дозволяє кожному користувачу додати власне оголошення та провести аукціон

Предметом дослідження є використання Web-орієнтованих технологій для проектування та розробки платформи онлайн-оголошень з елементами аукціону.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюсдайло В.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПЛАТФОРМИ

1.1 Технічне завдання на розробку платформи

Назва системи – платформа онлайн-оголошень з елементами аукціону.

Мета – розробка та впровадження платформи онлайн-оголошень з елементами аукціону, який надає інструменти для продажу власних товарів. Дана платформи може використовуватися як фізичними особами для продажу власного майна, так і компаніями для продажу товарів.

Актуальність - інтернет-аукціони та платформи онлайн-оголошень стають все більш популярними серед користувачів, які шукають спосіб продати або купити товари та послуги. У світлі зростаючої популярності електронної комерції та змін у способі споживання, створення такої платформи може задовольнити попит на зручний та ефективний інструмент для торгівлі та обміну різноманітними товарами та послугами. Крім того, він може сприяти розвитку малого та середнього бізнесу, надаючи їм можливість ефективно просувати свої продукти та залучати нових клієнтів через онлайн-канали.

Технічні вимоги:

- Платформа повинен бути здатен підтримувати мінімум 400 одночасно працюючих користувачів, що зв'язані з загальною БД.
- Час відгуку для звичайних задач складає не більше 5 секунд, а для складних задач – 20 секунд.
- Платформа має відповідати усім стандартам інтерфейсу користувача Microsoft® Windows® або ОС UNIX.
- Інтерфейс платформи має бути представлений у вигляді WEB-додатку, а сервер має обробляти запити, що надходять та надсилати відповіді.

Функціональні вимоги.

Авторизація та аутентифікація користувачів.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюдаило В.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

На платформі повинна бути можливість реєструвати нових користувачів, а також авторизація користувачів, що уже мають облікові записи.

Створення оголошень,

Користувач повинен мати змогу розміщувати на платформі власні оголошення. При додаванні оголошення можна ввести назву товару, що продається, мінімальну ставку та ціну викупу товару. Також є можливість додати фотографію, що ілюструє товар, що продається. Інші користувачі повинні мати змогу переглядати додані оголошення. Адміністратор повинен мати змогу проводити верифікацію оголошень.

Участь в аукціонах

Користувач повинен мати можливість зробити ставку на будь-яке оголошення, яке він забажає. Ставка завжди є сумою поточної ставки та мінімальної. В ситуаціях, коли поточної ставки не існує ставкою буде задана продавцем мінімальна ставка. Будь-який товар є можливість викупити в один клік без участі в аукціоні. Ціну на такий викуп задає продавець товару.

Фільтрація оголошень

Користувач може використовувати зручну комбіновану фільтрацію оголошень на головній сторінці, шукаючи потрібні йому оголошення по назві, категорії або проміжку між максимальною та мінімальною ставками.

Перемога в аукціоні

Користувач, що переміг в аукціоні і запропонував найбільшу ставку, або викупив його, може побачити повідомлення про те, що потрібно додати свої персональні дані для адреси доставки. Власник оголошення може переглянути оголошення, що були придбані, але ще не були надіслані. При перегляді адреси власник може відмітити оголошення як відправлене.

Застосовність:

1. Зручність використання:

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс повинен бути представлений у форматі WEB-додатку і мати простий та мінімалістичний інтерфейс. Головною вимогою є інтуїтивна зрозумілість інтерфейсу

Надійність:

1. Доступність:

Платформа має бути доступною для користувачів протягом усього часу. Винятками можуть бути ситуації технічного обслуговування або надзвичайні події.

2. Напрацювання на відмову:

Платформа повинна безвідмовно працювати протягом 30 робочих днів.

3. Норма дефектів:

Максимальною нормою помилок є 1 помилка на 10000 рядків коду.

Продуктивність:

1. Одночасно працюючі користувачі:

Система повинна підтримувати 400 одночасно працюючих користувачів, що пов'язані однією БД

2. Час відгуку:

Час відгуку для звичайних задач складає не більше 5 секунд, а для складних задач – 20 секунд

Придатність до експлуатації:

1. Масштабованість

Система повинна підтримувати 400 одночасно працюючих користувачів, що пов'язані однією БД. Повинна бути можливість збільшення числа одночасних користувачів у разі, коли популярність платформи зростає. Оскільки неможливо сказати точну кількість користувачів на платформі такого типу, система повинна мати можливість обробляти велику кількість одночасних користувачів.

2. Оновлення версій

Оновлення версій мають автоматизовано відбуватися через систему контролю версій та систему оновлення версій на комп'ютерах користувачів.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Обмеження проектування:

1. Застосовувані стандарти:

Система має відповідати стандартам інтерфейсу користувача як для операційних систем Microsoft® Windows® та ОС UNIX, так і сучасним нормам дизайну веб-додатків.

2. Вимоги до середовища виконання:

Система повинна задовольняти зазначеним нижче вимогами на комп'ютері в наступній мінімальній комплектації:

- 2048 МВ оперативної пам'яті
- 12 ГБ вільного дискового простору
- Процесор з тактовою частотою 2 ГГц
- ОС Windows або Linux
- Інтернет з'єднання

3. Вимоги до СКБД і доступу до даних:

В ядрі системи має бути використана промислова СКБД, орієнтована на документи, така як MongoDB. Усі звернення до інформації повинні здійснюватися через інтеграцію з MongoDB.

Антивимоги:

1. Заборонено використовувати інформацію, заборонену нормами моралі та законодавством.

2. Заборонено використовувати інформацію, що не є дійсною і/або може задати психологічної шкоди користувачам.

Бізнес логіка:

Реклама платформи на інших сайтах. Умова: власні кошти.

- Продукти та послуги: проект надає можливість проводити торгівлю.
- Користування платформою є доступною для будь-якого зареєстрованого користувача.

1.2 Аналіз існуючого програмного забезпечення за тематикою проекту

На даний момент існує безліч WEB-додатків, що реалізують мету даної роботи. Кожен з додатків має унікальні можливості та направлений на

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

конкретну аудиторію. Розроблювана платформи буде розглядатися в контексті порівняння з існуючими проектами, що користуються великою популярністю.

OLX

Платформа OLX (рис. 1.1) є найпопулярнішим в Україні платформою онлайн-оголошень. Платформа дозволяє усім зареєстрованим користувачам розміщувати оголошення про власний товар, вести переписку з другим учасником покупки [2].

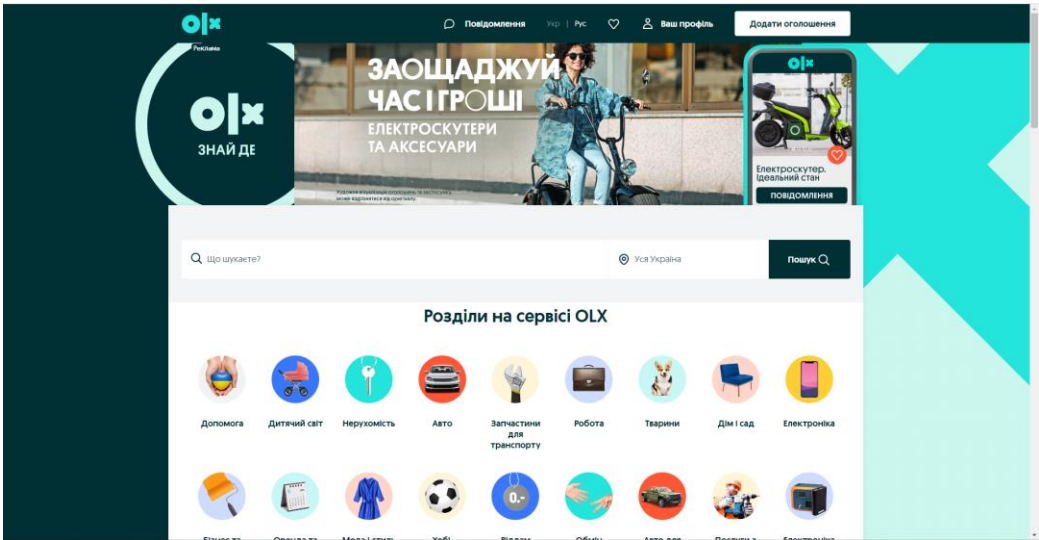


Рис. 1.1 – Інтерфейс OLX

Переваги:

- Можливість вести чати
- Зручний інтерфейс

Недоліки:

- Повільне завантаження контенту
- Обмежені можливості фільтрації
- Неповна оптимізація для телефонів

Proton.ua

Українська електронна торгівельна платформа (рис. 1.2), яка дозволяє підприємствам і фізичним особам продавати та купувати товари та послуги в Інтернеті. Платформа надає можливість розміщення оголошень, а також функціонал для здійснення покупок і продажів [3].

Переваги:

- Швидкодія

- Гнучкість пошуку

Недоліки:

- Обмежена мобільна оптимізація

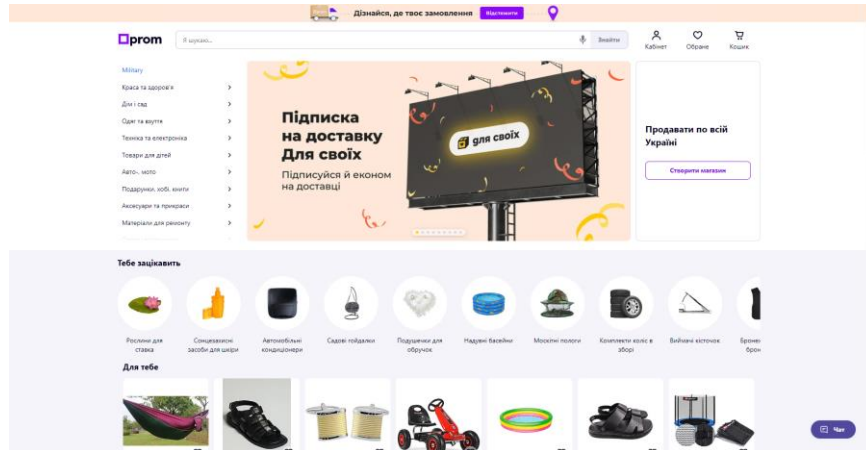


Рис. 1.2 – Інтерфейс Prom.ua

Ebay

Ebay (рис. 1.3) – це глобальна онлайн-платформа для купівлі та продажу товарів. Вона дозволяє користувачам розміщувати оголошення, брати участь в аукціонах та здійснювати покупки з різних куточків світу. Ебай пропонує широкий спектр категорій товарів та послуг.

Переваги:

- Міжнародний охоплення
- Велика кількість категорій товарів
- Функціонал аукціону

Недоліки:

- Складний інтерфейс для нових користувачів
- Високі комісії за продаж
- Повільне завантаження сторінок при великій кількості оголошень

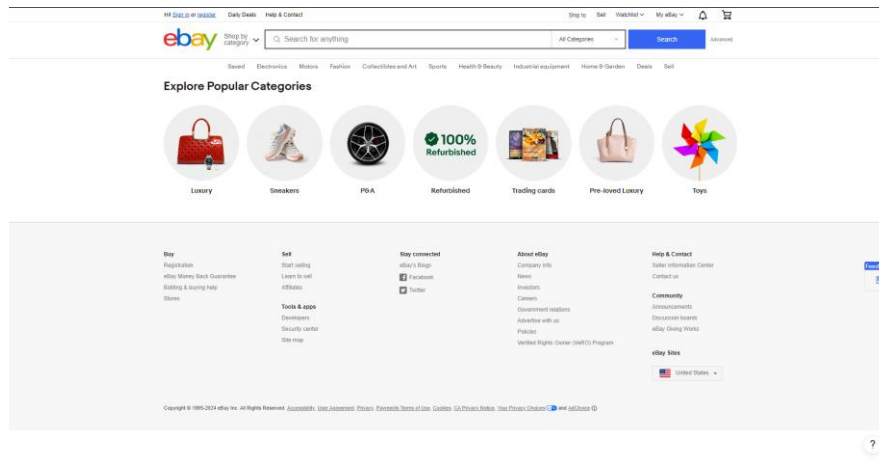


Рис. 1.3 – Інтерфейс Ebay

Перераховані платформи для онлайн-оголошень надають користувачам можливість купівлі та продажу товарів, але кожна з них відрізняється підходами та реалізацією. Важливим фактором є цільова аудиторія, на яку орієнтована кожна платформа. OLX спрямований на широку аудиторію, забезпечуючи зручний інтерфейс для повсякденного використання. Prom.ua фокусується на підприємствах та фізичних особах, пропонуючи швидкодію та гнучкість пошуку. Ebay має міжнародний охоплення та забезпечує функціонал аукціону, що приваблює глобальних покупців та продавців [4].

Для кращого розуміння розглянутих систем, порівнюємо їх функціональні можливості (табл. 1.1).

Таблиця 1.1

Функціональні можливості

Назва	OLX	Prom.ua	Ebay	Платформа онлайн-оголошень з елементами аукціону
Можливість розміщення оголошень	+	+	+	+
Можливість проведення аукціонів	-	-	+	+
Ведення чатів	+	+	+	-
Можливість фільтрації оголошень	+	+	+	+

1.3 Аналіз вимог системи та користувачів платформи онлайн-оголошень з елементами аукціону

Метою аналізу вимог є збір та аналіз високорівневих потреб і можливостей платформи онлайн-оголошень з елементами аукціону. Вимоги акцентують увагу на можливостях, необхідних співвласникам і цільовим користувачам, і на те, чому ці потреби існують.

Бізнес-вимоги:

1. Основні вимоги: проект створюється для надання функціоналу для розташування оголошень, проведення аукціонів та покупки продуктів.
2. Представлення проекту: проект буде реалізовано у форматі WEB-додатку, що містить набір інструментів для розміщення оголошень та проведення онлайн-трогів
3. Прискорення пошуку інформації: зручний інтерфейс для комбінованого пошуку оголошень
4. Підтримка великої кількості користувачів: сервер має можливість витримувати навантаження великої кількості користувачів та обробляти усі запити.

Вимоги користувачів:

Продавець:

1. Має можливість створювати власні оголошення
2. Має можливість переглядати адреси користувачів, що перемогли в аукціонах
3. Має можливість відмітити оголошення, як «відправлене»
4. Має можливість переглядати оголошення інших користувачів

Покупець:

1. Має можливість приймати участь в аукціонах
2. Має можливість викупити товар
3. Має можливість залишати адресу на оголошеннях, в яких переміг
4. Має можливість переглядати оголошення інших користувачів

Адміністратор:

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В				15
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Має усі можливості звичайних користувачів
 2. Може переглядати нові оголошення та пропускати їх, або блокувати
- На основі проаналізованих вимог була створена діаграма варіантів

Діаграма варіантів використання (рис. 1.4) відображає проаналізовані вимоги користувачів [5]. На рисунку видно усі дії, що може виконати кожен користувач платформи, зважаючи на те, яка саме роль йому належить. Можна побачити, що деякі ролі мають суміжні дії, тобто не обмежені певним набором. До таких можна віднести адміністратора, що має можливість виконати дії усіх інших ролей.

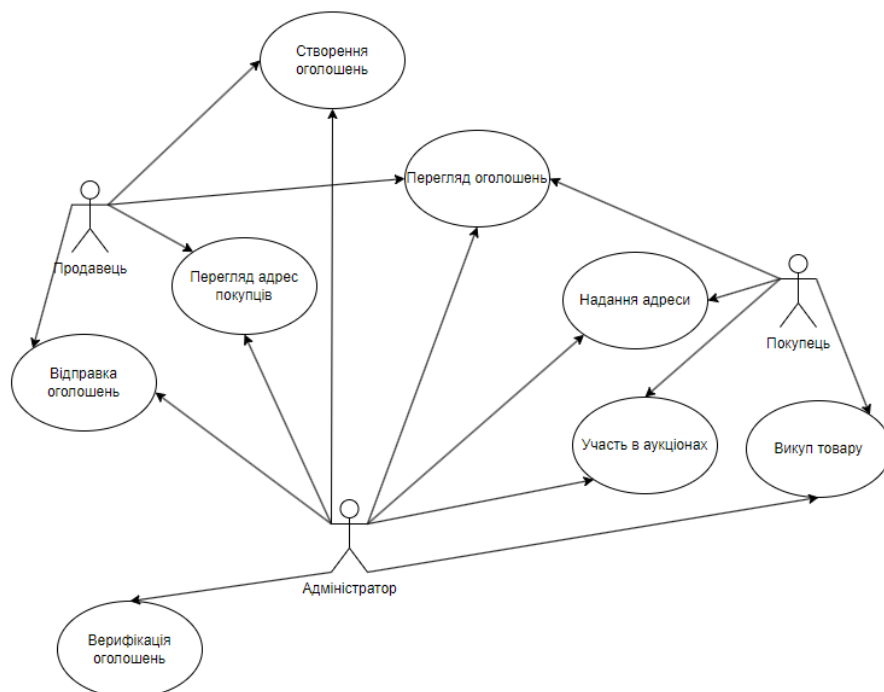


Рис. 1.4 – Діаграма варіантів використання

1.4 Обґрунтування вибору інструментальних засобів та архітектури платформи онлайн-оголошень з елементами аукціону

Існує велика кількість можливих архітектур для розробки додатків, серед яких вибір упав на клієнт-серверну архітектуру. Дана архітектура поділяється на наступні компоненти:

- Сервер – приймає запити від клієнтів, обробляє їх та надсилає відповіді.

- Клієнти – спілкуються з сервером використовуючи призначені для цього застосунки, у даному випадку – інтернет браузер.
- Мережа – зв'язок клієнт з сервером відбувається в інтернет мережі.
- База даних – відповідає за збереження даних.

Дана модель визначає три рівні операцій:

- Рівень представлення даних – інтерфейс, що допомагає даним потрапити до користувачів
- Прикладний рівень – реалізація логіки сервера, обробка даних.
- Рівень управління даними – забезпечення збереження даних та доступу до них. Реалізується за допомогою СКБД.

Обрану архітектуру необхідно реалізувати за допомогою інструментів розробки веб-інтерфейсу та веб-серверу.

Для розробки веб-інтерфейсу платформи онлайн-оголошень з елементами аукціону було обрано наступні засоби:

Розмітка WEB-сторінок: HTML5, CSS;

Мова програмування: Node.js;

Фреймворк: React.js;

Додаткові бібліотеки: уір, axios, react-router-dom.

Розмітка веб-сторінок за допомогою HTML5 і CSS забезпечує сучасний і адаптивний дизайн, який є основою будь-якого веб-додатку. Node.js було обрано як основну мову програмування завдяки його швидкодії та можливості використовувати одну мову як на стороні сервера, так і на стороні клієнта.

React.js – потужний фреймворк для створення динамічних і інтерактивних користувацьких інтерфейсів. Він дозволяє компонувати сторінки з окремих компонентів, що полегшує підтримку та масштабування коду. React.js забезпечує високу продуктивність і гнучкість, що є важливим для створення інтерактивного веб-інтерфейсу [7].

Бібліотека уір використовується для валідації даних, axios – для здійснення HTTP-запитів до сервера, а react-router-dom [8] дозволяє

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

реалізувати навігацію між сторінками веб-застосунку. Ці бібліотеки доповнюють функціонал React.js і спрощують процес розробки [9].

Для розробки серверної частини використовуються наступні засоби:

- Мова програмування: Node.js;
- Фреймворк: Express;
- Бібліотека для роботи з базою даних: mongoose;
- Додаткові бібліотеки: bcrypt, jsonwebtoken, multer.

Node.js забезпечує високу продуктивність і неблокуючу модель введення/виведення (I/O), що дозволяє ефективно обробляти велику кількість одночасних запитів. Express – мінімалістичний і гнучкий фреймворк для Node.js, що значно спрощує розробку серверних застосунків. Він забезпечує всі необхідні інструменти для швидкого створення RESTful API та налаштування маршрутизації для обробки запитів.

MongoDB – це документно-орієнтована система керування базами даних, яка написана на мові C++. Вона забезпечує високу продуктивність, гнучкість та масштабованість, що робить її ідеальним вибором для сучасних веб-додатків [11].

Характеристики MongoDB:

Документно-орієнтована: Замість реляційних таблиць, MongoDB зберігає дані у вигляді JSON-подібних документів (BSON), що дозволяє зберігати складні структури даних.

Гнучка схема: MongoDB не вимагає заздалегідь визначати структуру даних, що дозволяє легко змінювати схему бази даних у процесі розробки.

Масштабованість: Підтримує горизонтальне масштабування шляхом шардінгу, що дозволяє розподіляти дані між кількома серверами для підвищення продуктивності.

Вбудовані реплікації: Підтримує реплікаційні набори для забезпечення високої доступності та відмовостійкості даних.

Потужні запити: Підтримує складні запити, індексування та агрегації, що дозволяє ефективно обробляти великі обсяги даних.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Висока продуктивність: Оптимізована для роботи з великими обсягами даних та високими навантаженнями.

Кросплатформність: Підтримується на різних операційних системах, включаючи Windows, Linux та macOS.

MongoDB має потужну інтеграцію з Node.js через бібліотеку mongoose, яка забезпечує зручний інтерфейс для взаємодії з базою даних. Mongoose дозволяє визначати схеми даних, виконувати валідацію, а також забезпечує простий доступ до даних через модельно-орієнтований підхід.

Mongoose є бібліотекою-драйвером для роботи з MongoDB, яка дозволяє легко визначати схеми даних і взаємодіяти з базою даних. MongoDB було обрано як база даних завдяки її гнучкості та можливості масштабування, що є критично важливим для динамічних веб-додатків.

Бібліотека bcrypt використовується для захисту паролів користувачів шляхом хешування, що забезпечує безпеку зберігання паролів. JSON Web Token (jsonwebtoken) [6] використовується для створення та верифікації токенів доступу, що дозволяє реалізувати систему авторизації та аутентифікації користувачів. Multer забезпечує обробку файлів, які завантажуються на сервер, що є важливим для реалізації функціоналу додавання зображень до оголошень.

Висновки до першого розділу

У першому розділі було детально розглянуто технічне завдання на розробку платформи онлайн-оголошень з елементами аукціону, визначено його ціль та мету, а також акцентовано увагу на актуальності даного проекту. Було проведено аналіз функціональних і технічних вимог до системи, враховуючи потреби користувачів та бізнесу.

Оглянуті та порівняні існуючі платформи, такі як OLX, Prom.ua та Ebay, що дозволило виявити їх переваги та недоліки. Це дало можливість визначити конкурентні переваги та необхідні функції для розроблюваної платформи.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Проаналізовано вимоги користувачів та системи, що дало змогу створити діаграму варіантів використання. Це допомогло чітко зрозуміти, як різні користувачі будуть взаємодіяти з системою та які функції їм необхідні.

На основі аналізу було обрано клієнт-серверну архітектуру для реалізації платформи. Описано інструменти та технології, що будуть використовуватися для розробки веб-інтерфейсу та серверної частини.

Загалом, цей розділ закладає міцну основу для подальшого проектування та розробки платформи онлайн-оголошень з елементами аукціону, забезпечуючи глибоке розуміння вимог та технологій, необхідних для досягнення поставлених цілей.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ОНЛАЙН-ОГОЛОШЕНЬ З ЕЛЕМЕНТАМИ АУКЦІОНУ

2.1 Опис акторів та алгоритмів роботи

Наступним етапом розробки платформи онлайн-оголошень з елементами аукціону є аналіз алгоритму програми та створення діаграм активностей.

Основними акторами (табл. 2.1) є користувачі з такими ролями: продавець, покупець та адміністратор.

Таблиця 2.1

Користувачі

Актор	Короткий опис
Продавець	Має можливість створювати оголошення, переглядати оголошення що були продані та отримувати дані від користувачів, що придбали ці оголошення
Покупець	Має можливість переглядати оголошення інших користувачів, приймати участь в аукціонах, викупати лоти та надавати свої дані для відправки товарів
Адміністратор	Може виконувати усі дії, що й зазначені вище актори. Додатково має можливість проводити верифікацію новостворених оголошень.

Актори мають можливість взаємодіяти між собою за допомогою WEB-інтерфейсу та можуть спілкуватися з сервером. Далі розглянемо основні алгоритми роботи, використовуючи діаграми активностей [20].

Першою буде створена діаграма активності для додавання нового оголошення (рис. 2.1).

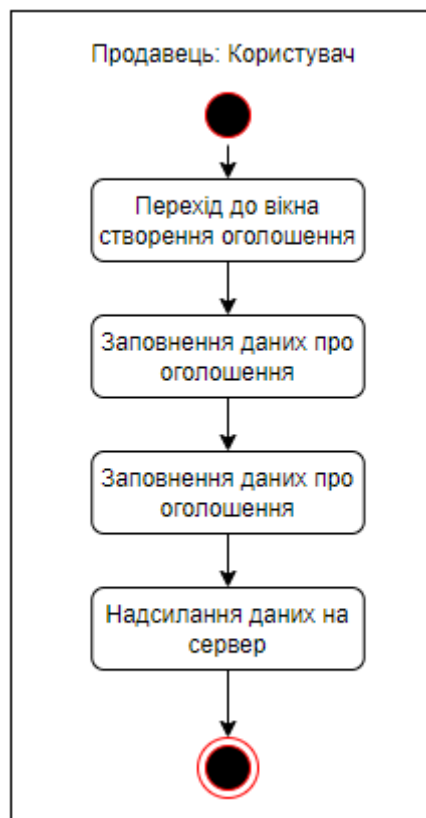


Рис. 2.1 – Діаграма активності для додавання оголошень

Створення оголошення ініціює продавець, що хоче розмістити власний товар на платформі. Для цього він натискає кнопку «Додати оголошення», після чого відкривається модальне вікно, де знаходиться форма, яку користувач має заповнити. Після заповнення і надсилання форми вся інформація надходить на сервер, де і створюється нове оголошення.

Далі бачимо діаграму (рис. 2.2) активності для верифікації оголошення. Як видно з діаграми, адміністратор може провести верифікацію оголошення після того, як користувач створить оголошення. Для проведення верифікації оголошень користувач має перейти на сторінку адміністратор, що знаходиться в профілі. На даній сторінці адміністратор може побачити усі новостворені записи, що не були затверджені жодним з адміністраторів. Переглянувши інформацію про оголошення адміністратор повинен прийняти рішення про те,

чи підлягає дане оголошення додаванню на платформі. У разі якщо так, адміністратор натискає кнопку «Пропустити», інакше ж – «Заблокувати».

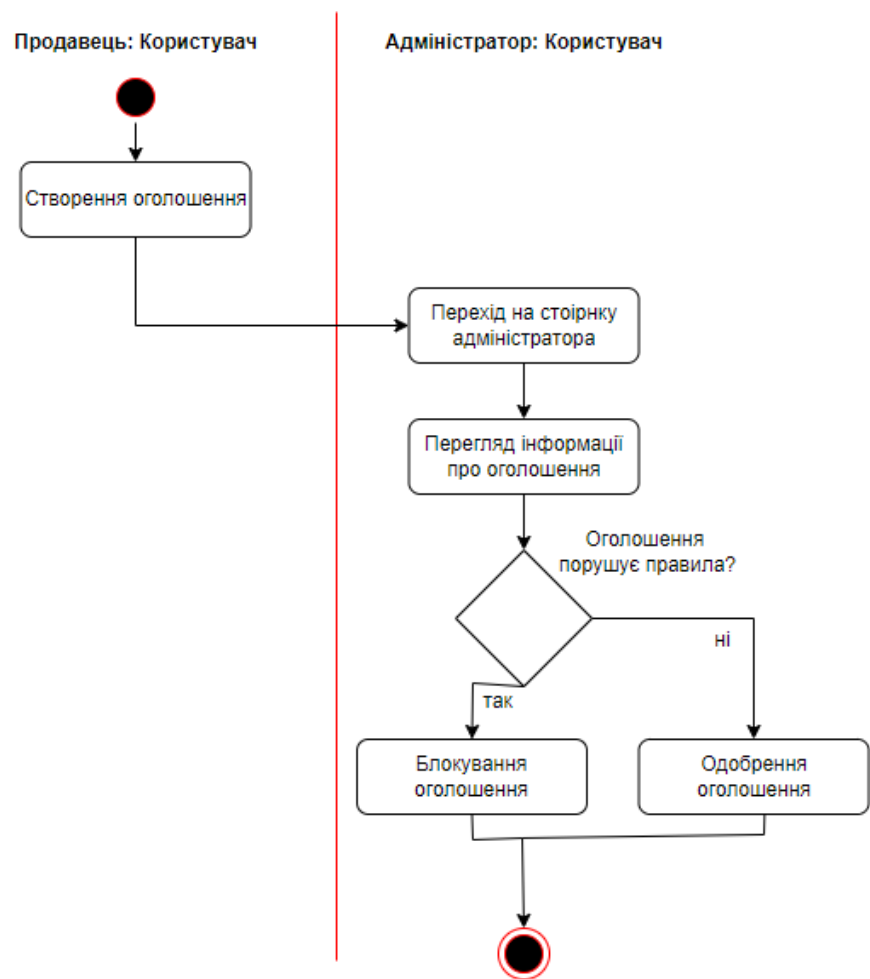


Рис. 2.2 – Діаграма активності для верифікації оголошення

Як видно з діаграми (рис. 2.3), для пошуку оголошень користувач повинен перейти на головну сторінку, де відображаються усі можливі оголошення. По замовчуванню оголошення відображаються по даті додавання до бази даних. У разі, якщо для користувача є потреба у фільтрації оголошень, наприклад за назвою або за ціною, він може заповнити параметри пошуку в верхній частині екрану. Після заповнення форми набір оголошень, що відображаються, буде змінено, в залежності від заданих параметрів. Переглянувши сторінку користувач має вирішити, чи є потрібний товар на екрані. У разі, якщо потрібного товару немає користувач повинен перемикнути сторінку в нижній частині екрану.

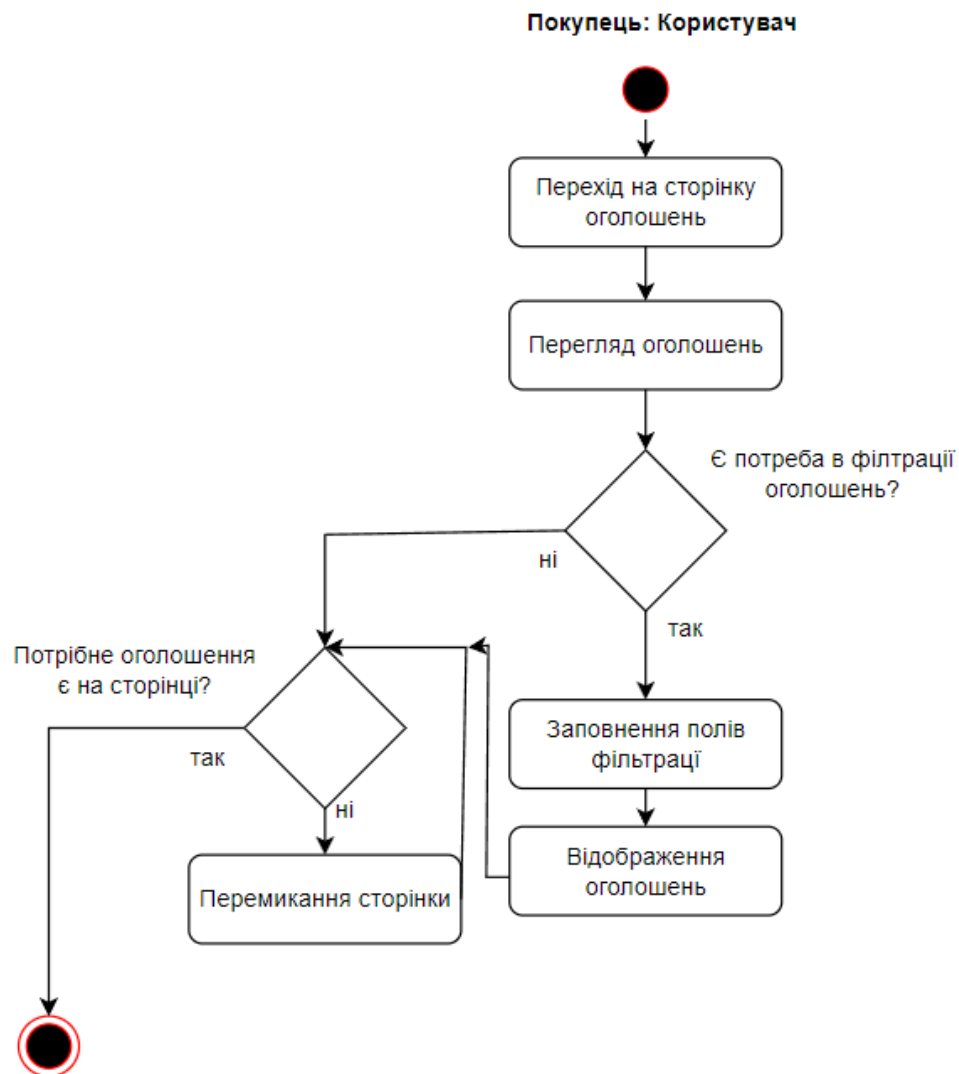


Рис. 2.3 – Діаграма активності пошуку оголошення

На діаграмі активності участі в аукціоні (рис. 2.4) можна побачити те, як відбувається участь в аукціоні, як проходить перевірка на переможця аукціону і те, як відбувається викуп товару. Переглянувши діаграму бачимо, що користувач має вибрати потрібне оголошення та перейти на його сторінку. На сторінці оголошення користувач може натиснути одну з вух кнопок. У разі, якщо користувач обирає кнопку для викупу товару, оголошення одразу отримує статус проданого та передається на сторінку додавання адреси у користувача. У випадку, коли користувач обирає зробити ставку він записується як поточний лідер аукціону. Надалі система постійно перевіряє чи не настала дата, коли закінчується аукціон. У разі, якщо аукціон закінчився поточний лідер стає переможцем аукціону. Якщо аукціон ще не закінчився,

то перевіряється чи не змінився лідер. У разі, якщо лідер уже змінився користувач може зробити ставку знову, перейшовши на сторінку оголошення.

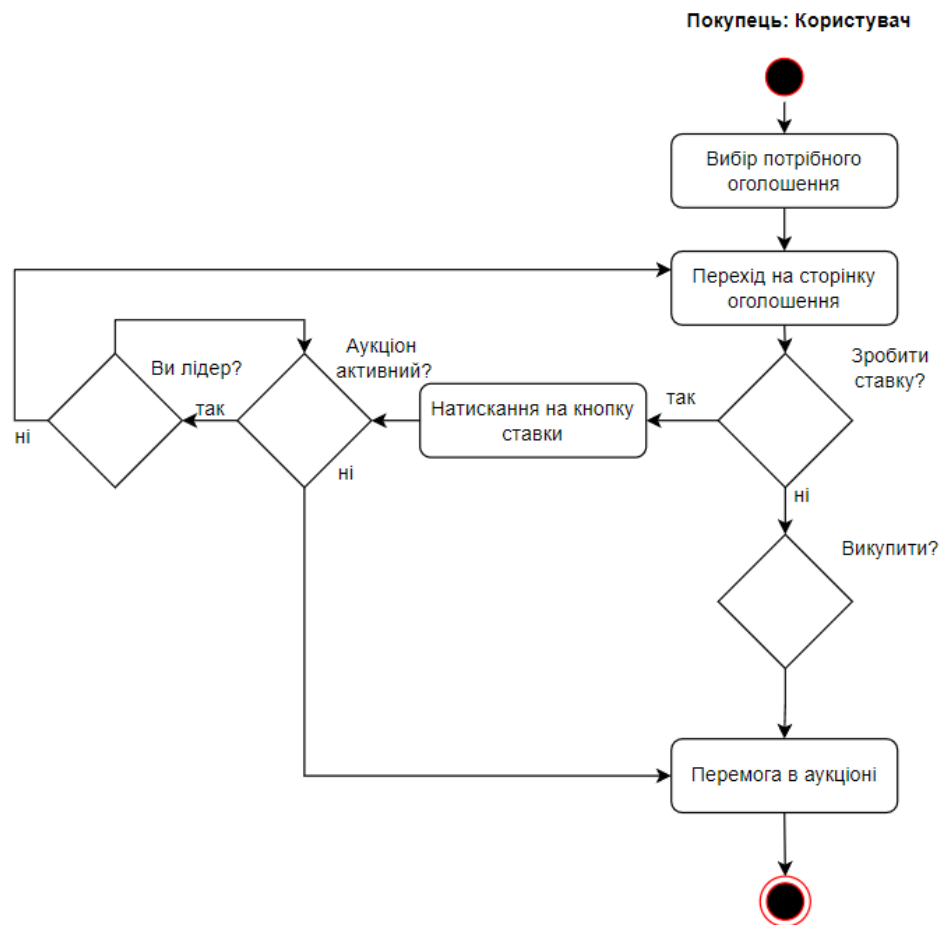


Рис. 2.4 – діаграма активності участі в аукціоні

2.2 Розробка бази даних

Наступним кроком в розробці платформи онлайн-оголошень з елементами аукціону буде опис та створення БД. База даних складається з 4 таблиць (табл. 2.2), кожна з яких зберігає окремі дані.

Таблиця 2.2

Опис таблиць

Назва таблиці	Короткий опис
Address	Містить в собі ідентифікатор оголошення та надану користувачем адресу
Category	Містить в собі назви категорій
Lot	Містить в собі дані про оголошення
User	Містить в собі інформацію про користувачів

Усі таблиці створені з використання бібліотеки-драйверу mongoose, що дозволило описувати структуру БД на мові програмування Node.js, завдяки чому значно пришвидшується процес розробки. Завдяки партерну програмування MVC усі моделі зберігаються в окремій папці, а кожна модель представлена у форматі об'єкта в окремому файлі [19].

Розберемо зв'язки таблиць БД, використовуючи діаграму візуалізації БД (рис. 2.5). На діаграмі видно, що таблиця Lot має зв'язки з усіма іншими таблицями, оскільки усі операції на сервісу так чи інакше пов'язані з роботою з оголошеннями. Таблиця пов'язана з таблицею User, по ідентифікатору користувача, що створив оголошення. З таблицею категорій по ідентифікатору категорії, до якої відноситься даний товар, а до таблиці з адреса по власному ідентифікатору, завдяки чому можна швидко отримати доступ до оголошення зі сторінки адрес. Усі інші таблиці не мають зв'язків між собою, оскільки їх зв'язок також представлений через таблицю з оголошеннями.

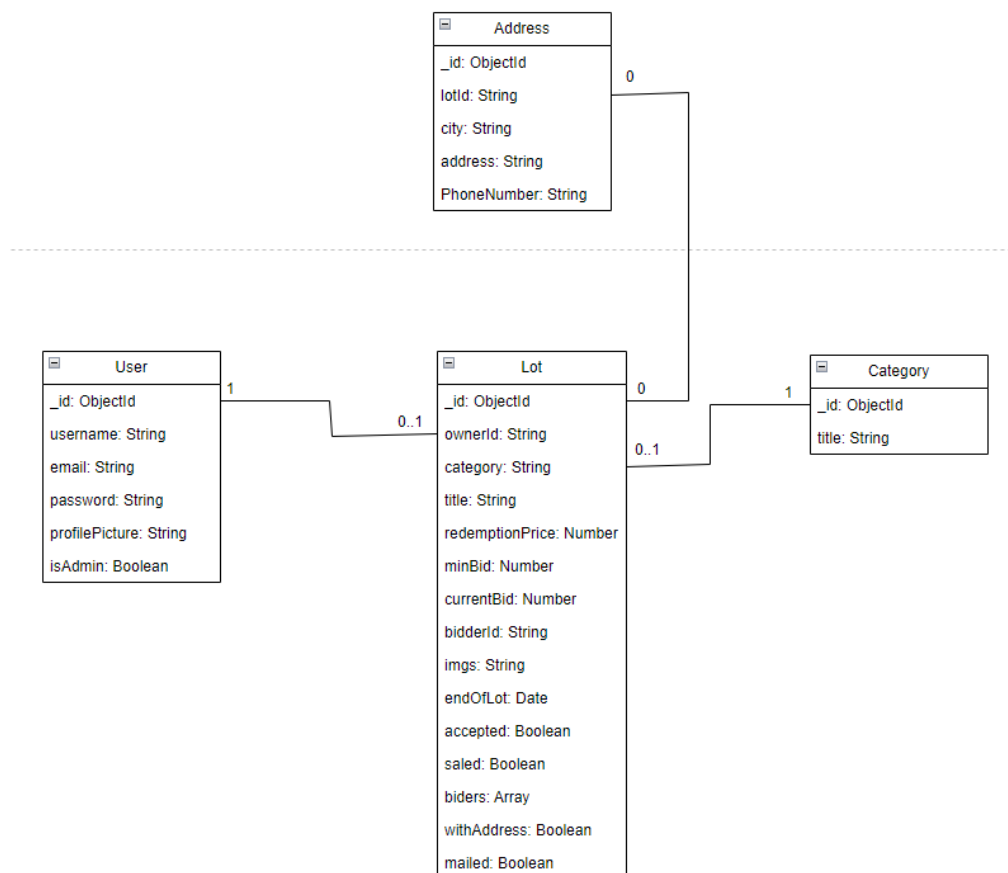


Рис. 2.5 – Візуалізація бази даних

Для кращого розуміння діаграми розберемо кожну таблицю детально.
Основною таблицею є таблиця оголошень (табл. 2.3).

Таблиця 2.3

Структура таблиці «Lot»

Lot			
Назва поля	Тип даних	PK/FK	Опис поля
_id	ObjectId	PK	Унікальний ідентифікатор запису
ownerId	String	FK	Унікальний ідентифікатор власника оголошення
category	String	FK	Унікальний ідентифікатор категорії
title	String	-	Назва оголошення
redemptionPrice	Number	-	Ціна викупу
minBid	Number	-	Мінімальна можлива ставка
currentBid	Number	-	Поточна ставка
bidderId	String	FK	Ідентифікатор учасника з найбільшою ставкою
imgs	String	-	Назва фотографії оголошення
endOfLot	Date	-	Дата завершення аукціону
Accepted	Boolean	-	Одобрення адміністратором
saled	Boolean	-	Поле що вказує на стан оголошення, чи продане воно
bidders	Array	FK	Масив усіх учасників аукціону
withAddress	Boolean	-	Поле, що вказує чи була додана адреса
Mailed	Boolean	-	Поле, що вказує, чи було товар надіслано

В даній таблиці реалізовано зберігання даних про усі оголошення, а також про їх статуси. Завдяки полю category таблиця поєднується з таблицею категорій, завдяки чому оголошення можна фільтрувати по різним

параметрам, а поле ownerId дає змогу користувачам швидко отримувати доступ до свої оголошень. Важливим є поле bidderId, яке зберігає ідентифікатор користувача, що зробив останню, а значить найбільшу, ставку, або ж ідентифікатор користувача, що викупив товар. Таким чином видно, що саме дана таблиця є основою БД та використовується для проведення більшої частини операцій на платформі.

Одна з основних таблиць (табл. 2.4), в якій організовано зберігання даних про усіх користувачів. За допомогою даної таблиці відбувається авторизація та автентифікація користувачів у системі. Майже усі контролери працюють безпосередньо з даною таблицею, для додавання власника та покупця до оголошення, відображення картинки профілю поточного користувача тощо.

Таблиця 2.4

Структура таблиці «User»

User			
Назва поля	Тип даних	PK/FK	Опис поля
_id	ObjectId	PK	Унікальний ідентифікатор користувача
username	String	-	Ім'я користувача, зазначене при реєстрації
email	String	-	Пошта користувача, зазначена при реєстрації
password	String	-	Хеш паролю
profilePicture	String	-	Картинка профілю користувача
isAdmin	Boolean	-	Ідентифікатор, що вказує, чи адміністратор користувач

Таблиця категорій (табл. 2.5), зберігає можливі категорії товарів, зв'язуючись з кожним оголошенням. Також з цією таблицею працює відображення назви в інтерфейсах платформи.

Структура таблиці «Category»

Category			
Назва поля	Тип даних	PK/FK	Опис поля
_id	ObjectId	PK	Унікальний ідентифікатор користувача
title	String	-	Назва категорії

Таблиця Address (табл. 2.6.) використовується для збереження адреси та персональних даних користувача, які будуть використовуватися продавцем для відправлення замовлення. Дана таблиця зв'язується з таблицею оголошень, що дозволяє легко отримати дані власнику оголошення та внести дані переможця аукціону.

Таблиця 2.6

Структура таблиці «Address»

Address			
Назва поля	Тип даних	PK/FK	Опис поля
_id	ObjectId	PK	Унікальний ідентифікатор запису
lotId	String	FK	Унікальний ідентифікатор лоту, до якого прив'язується адреса
city	String	-	Місто, яке зазначив користувач
address	String	-	Адреса, яку зазначив користувач
PhoneNumber	String	-	Номер телефону, який зазначив користувач

2.3 Комунікації і послідовність взаємодії об'єктів платформи

Етапом, що буде розглянуто наступним, є створення діаграм послідовностей, які значно покращать розуміння того, як буде відбуватися комунікація в системі [18].

Діаграма послідовностей додавання оголошень (рис. 2.6) починається з того, що користувач заповню форму. Після заповнення форма надсилається на контролер, де проходить збереження усіх переданих даних до відповідної моделі, у випадку даної діаграми збереження відбувається в моделі оголошень. Після успішного збереження інформація про підтвердження надходить на контролер. Фінальним етапом є запит контролера до шаблонізатора на відображення інтерфейсу. Оскільки шаблонізатором в нашому випадку є React, то запит представляє собою відповідь серверу з інформацією, яку потрібно відобразити на сторінці для користувача.

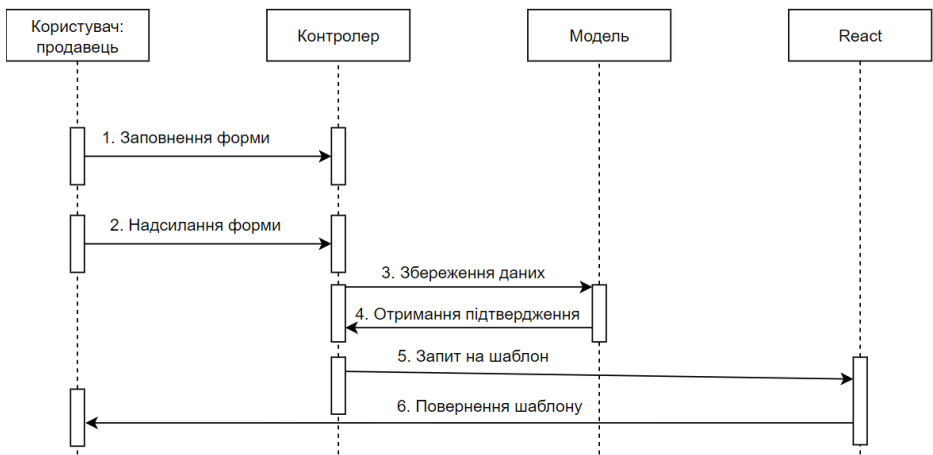


Рис. 2.6 – Діаграма послідовностей додавання оголошення

Діаграма послідовностей створення ставки (рис. 2.7) ілюструє, що створення ставки починається з того, що користувач знаходить потрібне йому оголошення та відкриває його сторінку, після чого контролер повертає на React всю потрібну інформацію про обраний шаблон. Отримавши сторінку з оголошенням користувач натискає кнопку «Зробити ставку» і надсилає свої дані і дані про свою ставку на сервер. Контролер робить запит до моделі, перевіряючи коректність даних про ставку, що прийшли від користувача і після отримання підтвердження зберігає нові дані. Отримавши підтвердження про успішне збереження контролер передає клієнту оновлені дані.

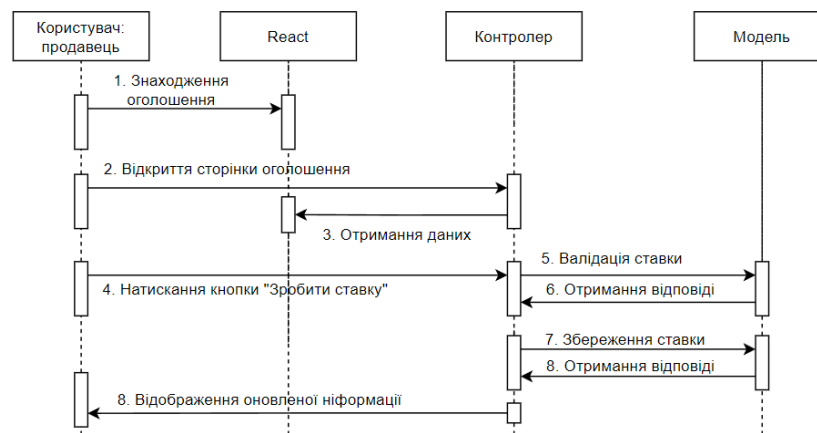


Рис. 2.7 – Діаграма послідовностей створення ставки

Діаграма послідовностей верифікації оголошення (рис. 2.8) показує, що у випадку, коли адміністратор переходить на сторінку для верифікації оголошень він відправляє запит про отримання усіх непідтверджених на сервер. На сервері контролер звертається до бази даних, отримує усі потрібні запити та відсилає їх шаблонізатору, який в свою чергу, відображає їх для користувача. Переглянувши нове оголошення, адміністратор приймає рішення про пропуск або видалення оголошення. Натиснувши одну з кнопок він надсилає відповідний запит на сервері. Контролер передає дані від користувача до моделі і, або змінює запис на підтверджений, або видаляє оголошення, після чого отримує відповідь від БД. Отримавши відповідь контролер надсилає дані для відображення шаблонізатору, який в свою чергу відображає дані клієнту

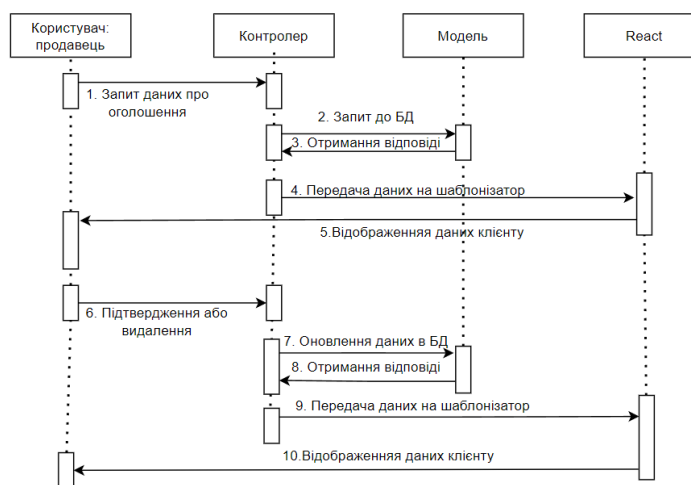


Рис. 2.8 – Діаграма послідовностей верифікації оголошення

2.4 Реалізація платформи онлайн-оголошень з елементами аукціону

Враховуючи, що система розроблена з використанням патерну MVC її можливо поділити на 3 складові.

Моделі (Model) Mongoose є способом визначення структури даних для колекцій у MongoDB. Вони є об'єктами, які представляють конкретну колекцію в базі даних. Моделі дозволяють виконувати операції з базою даних, такі як збереження, оновлення, видалення та запити.

Для прикладу візьмемо модель User

```
const mongoose = require("mongoose")

const UserSchema = new mongoose.Schema({
  username:{
    type:String,
    required: true,
    min: 3,
    max: 50,
  },
  email:{
    type: String,
    required: true,
    unique: true
  },
  password:{
    type: String,
    required: true,
    min: 8
  },
  profilePicture:{
    type: String,
    default: "default.jpg"
  },
  isAdmin:{
    type: Boolean,
    default: false
  }
})

module.exports = mongoose.model("User", UserSchema)
```

Кожна модель складається з потрібної кількості полів, а для кожного поля задаються параметри, такі як тип даних, стандартне значення,

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюсдайло В.В				32
Змн.	Арк.	№ докум.	Підпис	Дата		

унікальність і так далі. В даній частині коду можна побачити, що, наприклад, поле username має строковий тип даних, є обов'язковим та його довжина має бути від 3 до 50 символів.

Вид (View) відповідає за відображення користувацького інтерфейсу. З використанням React.js ми отримуємо змогу писати складну логіку одразу на стороні клієнта, що знімає навантаження з серверу, звільняючи його від опрацювання великої кількості однакових задач. Для прикладу розглянемо код відображення сповіщень.

```
import React, {useEffect} from "react";
import "./notification.css";

const Notification = ({ notification, setNotification }) => {

  useEffect(() => {
    if(notification){
      const timer = setTimeout(() => {
        setNotification('')
      }, 2000)

      return () => clearTimeout(timer);
    }
  })

  return (
    <div className="notificationContainer">
      <div className="notificationItem">
        {notification}
        <button onClick={() => setNotification('')}
className="notificationButton">x</button>
      </div>
    </div>
  );
};

export default Notification;
```

В даній частині коду важливим є таймер, що через 2 секунди ховає повідомлення. Для цього використовується хук useState без залежностей, завдяки чому він постійно прослуховується. Якщо додати пусту залежність, то хук відпрацює лише на завантаженні сторінки, якщо в залежність додати змінну, то хук буде відпрацьовувати лише при зміні змінної. В даному випадку через 2 секунду після того, як компонент було змонтовано набір даних для

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

відображення буде очищено, і повідомлення зникне, оскільки компонент монтується лише при наявності в ньому повідомлення.

Контролер (Controller) відповідає за обробку даних на сервері. Він приймає дані від користувача та займається їх обробкою. Контролер зв'язує між собою модель та вид.

```
router.post("/", auth, upload.single('image'), async (req, res) => {
  const imageFile = req.file
  const randomFileName = `${Date.now()}-${
    Math.random().toString(36).substring(7)}${path.extname(imageFile.originalname)}
`;
  fs.rename(imageFile.path, path.join(imageFile.destination, randomFileName),
    (err) => {
      if (err) {
        console.error('Помилка при збереженні файлу:', err);
      } else {
        console.log('Файл збережено:', randomFileName);
      }
    });
  try{
    const newLot = new Lot({
      ownerId: req.user.userId,
      title: req.body.productName,
      redemptionPrice: req.body.price,
      minBid: req.body.minBid,
      category: req.body.category,
      endOfLot: req.body.endDate,
      imgs: randomFileName
    })

    console.log()

    const lot = await newLot.save()
    res.status(200).json({message:"Ваш лот додано та відправлено на
перевірку"})
  }catch(err){
    return res.status(500).json(err)
  }
})
```

Даний код відповідає за додавання нового оголошення на платформу. На початку генерується випадковий рядок, який буде використовуватися як ім'я для файлу, що буде додано. Далі файл зберігається в папку з фотографіями товарів. Після цього створюється об'єкт нового оголошення, в поля якого записуються дані, що надходять від користувача на сервер. Після наведених вище дій об'єкт зберігається в таблицю Lot в базі даних і на клієнтську частину

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В				34
Змн.	Арк.	№ докум.	Підпис	Дата		

надсилається повідомлення про успішне збереження оголошення. У разі, якщо на одному з етапів виявляється помилка на сервер надсилається повідомлення про помилку.

Висновки до другого розділу

У даному розділі було здійснено детальний аналіз та проектування платформи онлайн-оголошень з елементами аукціону, що включає опис акторів, алгоритмів роботи, розробку бази даних, а також комунікації та послідовність взаємодії об'єктів платформи. Було розглянуто процеси додавання та верифікації оголошень, пошуку товарів і участі в аукціонах, що візуалізовано через діаграми активностей.

Таким чином, даний розділ забезпечує комплексний підхід до проектування та розробки платформи, включаючи як теоретичний аналіз, так і практичні аспекти реалізації, що закладає основу для створення функціонального та ефективного онлайн-платформи оголошень з елементами аукціону.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ОНЛАЙН-ОГОЛОШЕНЬ З ЕЛЕМЕНТАМИ АУКЦІОНУ

3.1 Порядок встановлення та налаштування платформи

Для розуміння того, як елементи в системі платформи онлайн-оголошень з елементами аукціону взаємодіють та взаємозалежать, була розроблена діаграма компонентів (рис. 3.1). Ця діаграма відображає взаємовідносини між різними частинами системи [16].

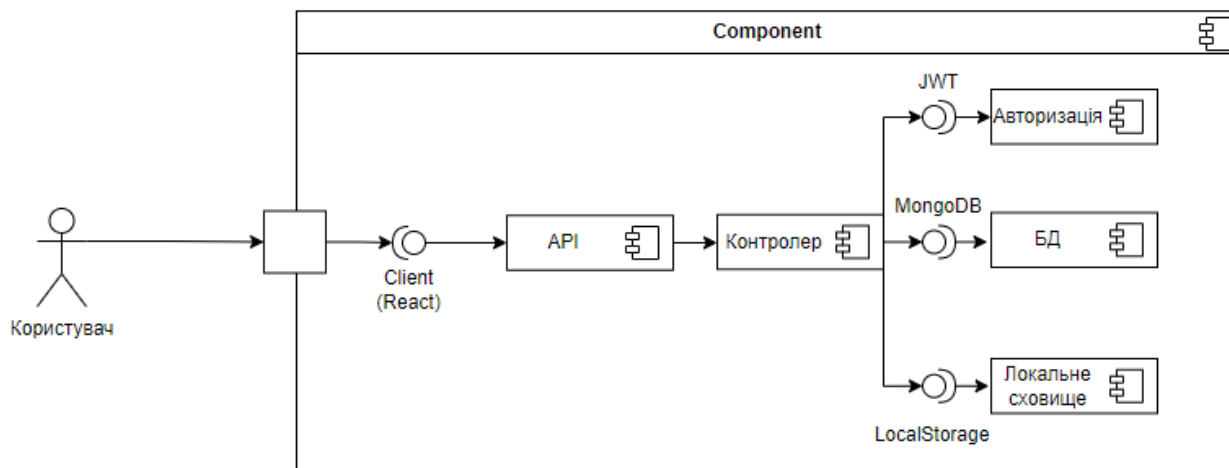


Рис. 3.1 – Діаграма компонентів

Основним джерелом даних, з якими працює платформа, є користувач. Усі запити, що надходять до серверу відправляються безпосередньо користувачем. Користувач спілкується з сервером, використовуючи React клієнт. Усі звернення до серверу відбуваються шляхом надсилання запитів на API серверу. В ситуаціях, коли контролеру треба зберегти, або змінити записи в базі даних він використовує компонент БД. Якщо ж контролеру потрібно авторизувати користувача він звертається до компонента авторизації, який використовує Json WebToken. У випадках, коли контролеру треба зберегти файли, наприклад картинки товарів, він звертається до компоненту локального сховища, за допомогою якого і відбувається збереження файлів.

Таким чином відбувається взаємодія між усіма компонентами платформи.

Також варто звернути увагу на створену діаграму розгортання, яка розроблена для надання інформації про фізичну будову системи [17].

На діаграмі (рис. 3.2) видно, що використовуючи комп'ютер користувач, за допомогою мережі інтернет, має можливість обмінюватися інформацією з сервером. Сервер, обробляючи інформацію, що надходить в запитах клієнта, звертається до потрібних йому компонентів. Якщо брати до уваги локальне сховище, то воно є частиною серверу і зберігає усі завантажені користувачами файли. Сервер БД є віддаленим, з ним сервер платформи зв'язується за допомогою спеціального драйверу.

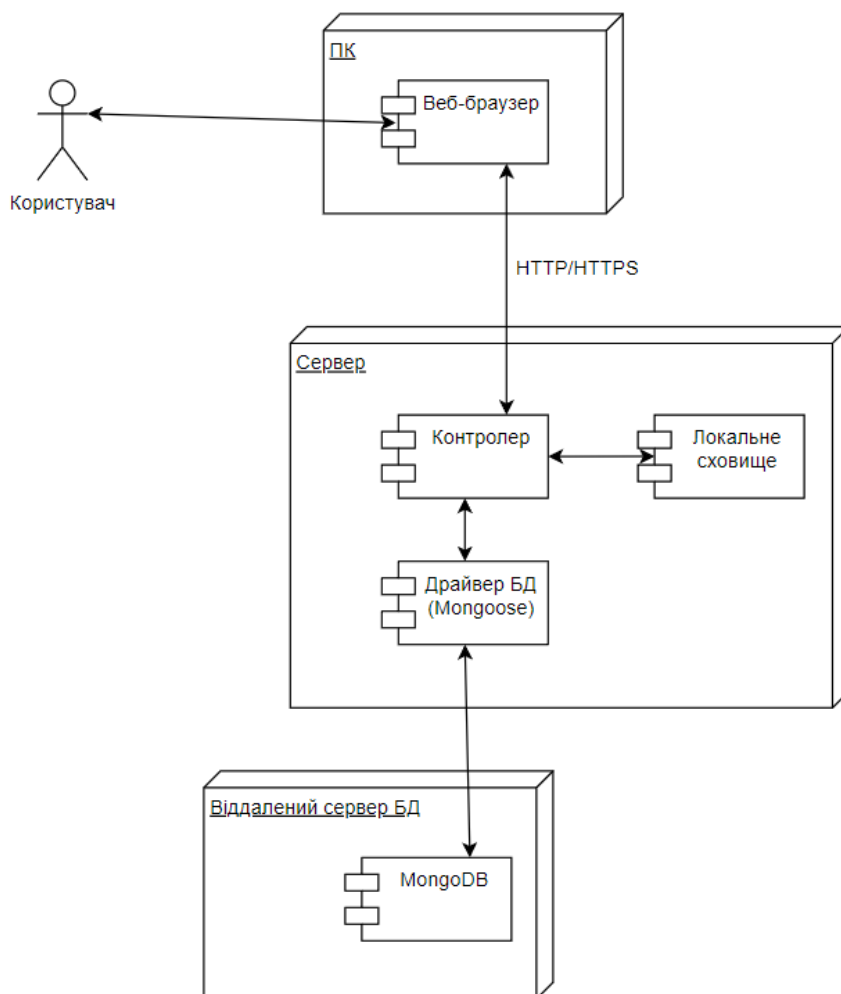


Рис. 3.2 – Діаграма розгортання

Розробивши діаграму розгортання видно, що для запуску платформи потрібно використовувати хостинг, що надає місце для збереження файлів у локальному сховищі. Оскільки база даних є віддаленою, то вона не потребує додаткового розгортання разом з сервером, але варто пересвідчитись, що хостинг дозволяє виконувати запити з серверу до зовнішніх джерел.

3.2 Опис інтерфейсу програмного продукту

При першому заході на сайт користувач потрапляє на сторінку реєстрації (рис. 3.3), де він має можливість створити власний аккаунт, оскільки для роботи з сайтом реєстрація є обов’язковою. Також на сторінці реєстрації можна побачити кнопку «Увійти в свій аккаунт», яка перенаправить користувача на сторінку авторизації.

Рис. 3.3 – Сторінка реєстрації

Сторінка авторизації (рис. 3.4) з точки зору дизайну наслідує сторінку реєстрації. На сторінці є кнопка «Створити аккаунт», яка перенаправляє користувача на сторінку реєстрації.

Рис. 3.4 – Сторінка авторизації

Після авторизації на кожній сторінці користувач буде бачити “шапку” (рис. 3.5). Тут відображаються усі основні елементи роботи з сайтом. Першим

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюдаило В.В				38
Змн.	Арк.	№ докум.	Підпис	Дата		

розташований напис з логотипом сайту, при натисканні на який користувач завжди потрапить на головну сторінку. Далі розташоване поле пошуку, заповнюючи дані в якому, користувач може фільтрувати дані оголошення, що відображаються на головному екрані. Після пошуку знаходиться кнопка «Додати оголошення», яка відкриває модальне вікно (рис. 3.6), де користувач може додати нове оголошення. Центр сповіщень, що знаходиться далі відкриває модальне вікно, де покупець може побачити придбані товари, а продавець – продані. І останній елемент шапки це фото профілю користувача, що перенаправляє на сторінку користувача.

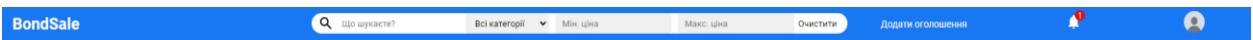


Рис. 3.5 – Шапка сайту

Рис. 3.6 – Модальне вікно додавання оголошення

У вікні, що представлене на рисунку 3.4 можна побачити форму додавання товару. Форма складається з 5 полів, в яких користувач вказує основну інформацію про свій товар. Також є кнопка «Додати фото», після натискання якою завантажене фото відразу відобразиться на екрані користувача.

При виконанні дій з базою даних користувач завжди бачить сповіщення (рис. 3.7), яке можна закрити натиснувши на хрестик, або ж почекати 2 секунди і воно зникне автоматично.

Ваш лот додано та відправлено на перевірку

X

Рис. 3.7 – Сповіщення

На скріншоті (рис. 3.8) можна побачити вікно адміністратора де відбувається верифікація оголошень, що додаються користувачами. Адміністратор може побачити усю інформацію про лот, а натиснувши на назву картинку може побачити картинку, яку завантажив користувач. Праворуч знаходяться кнопки «Видалити» та «Прийняти», які відповідають за видалення товару, що не пройшов критерії перевірки, та за прийняття товару, що не порушує правил.

BondSale					
Що шукаєте?		Всі категорії	Мін. ціна	Макс. ціна	Очистити
Додати оголошення					
Профіль			Панель Адміністратора		
Назва лоту	Категорія	Ціна викупу	Мінімальна ставка	Зображення	Дія
Samsung Galaxy A24 6/128GB Dark Red	Мобільні телефони	28000	3000	1717600766941-dcf69.jpg	Видалити Прийняти

Рис. 3.8 – Сторінка адміністратора

На головній сторінці (рис. 3.9) відображаються усі можливі для перегляду оголошення, одночасно на сторінці може знаходитися до 8 оголошень, а для перегляду більшої кількості використовується перемикач сторінок (рис. 3.10). При натисканні на назву оголошення користувачу буде представлено модальне вікно, на якому він може приймати участь в аукціоні.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюдаило В.В.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

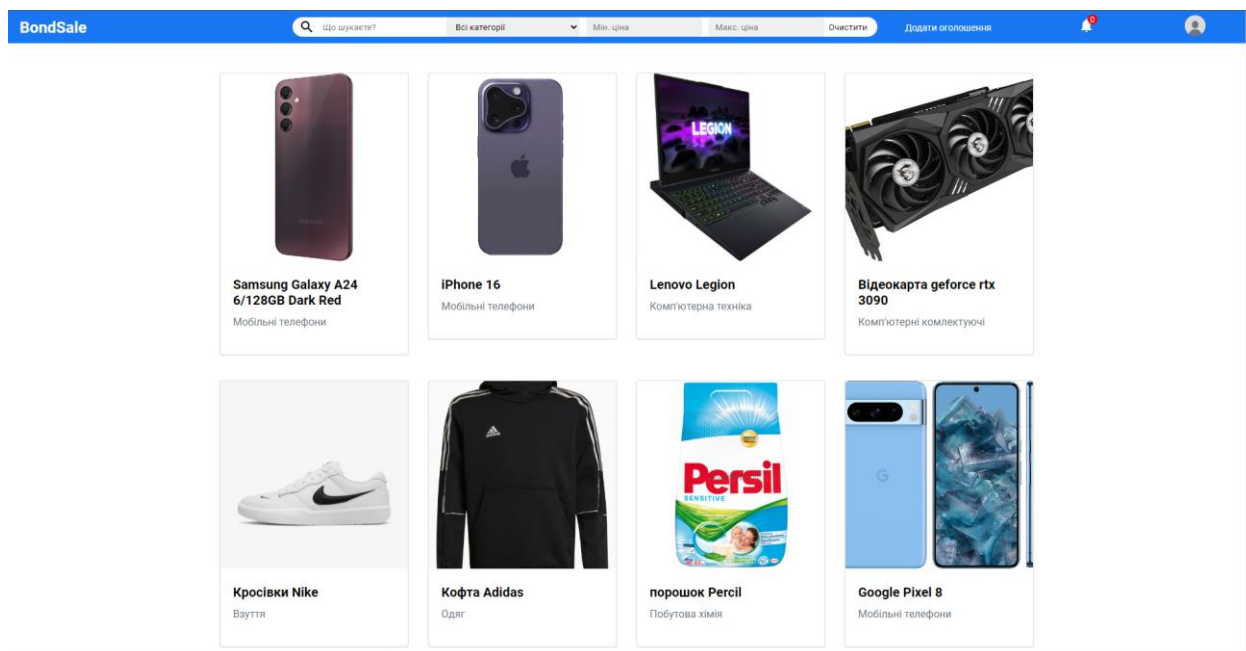


Рис. 3.9 – Головна сторінка

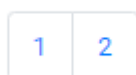


Рис. 3.10 – Перемикач сторінок

Звертаючи увагу на модальне вікно товару (рис. 3.11) можна побачити, що тут знаходиться фото товару та дві кнопки. Перша дає користувачу можливість зробити ставку, друга – викупити товар. Розмір ставки задається автоматично за формулою поточна ставка + мінімальна ставка. В верхній частині вікна відображається назва товару, а знизу праворуч розташована кнопка для закриття модального вікна.

Коли користувач купує товар, він може побачити товари, що придбав, але ще не залишив адресу у спеціальному модальному вікні (рис. 3.12).

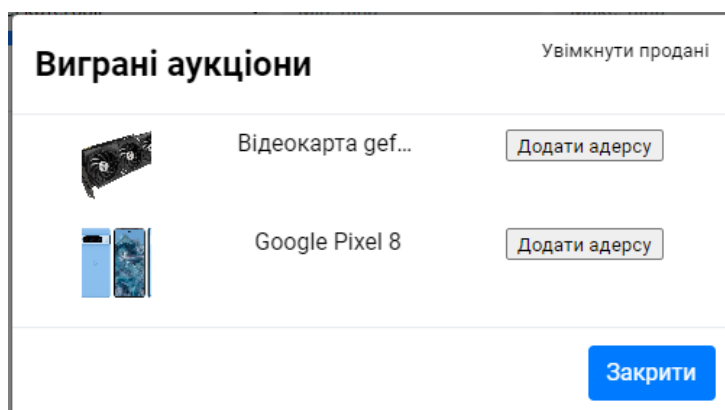


Рис. 3.11 – Вікно перегляду куплених лотів

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

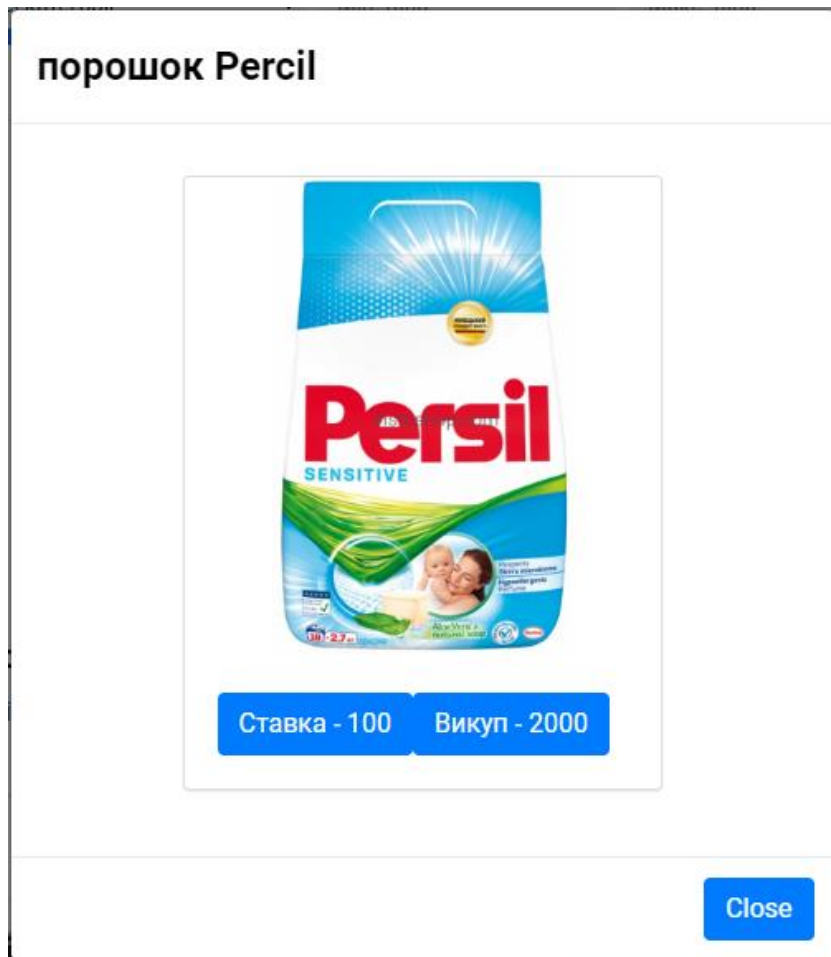


Рис. 3.12 – Модальне вікно товару

Натиснувши на кнопку «Додати адресу» в вікні, користувач потрапляю до вікна (рис. 3.13), де він повинен заповнити форму та надіслати дані продавцю, який повинен надіслати товар за вказаними даними.

Виграні аукціони

Увімкнуті продані

Місто

Житомир

Адреса

Київська, 77

Номер телефону

0966779087

Відправити

Назад

Закрити

Рис. 3.13 – Форма вводу персональних даних

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Конюсдайло В.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Натиснувши кнопку «Увімкнути продані», користувач потрапить до вікна (рис. 3.14) і відображає лоти, які придбали, але ще не були надіслані. У разі, якщо покупець уже додав адресу, з'явиться кнопка «Переглянути адресу», інакше – напис «Адреса не додана».

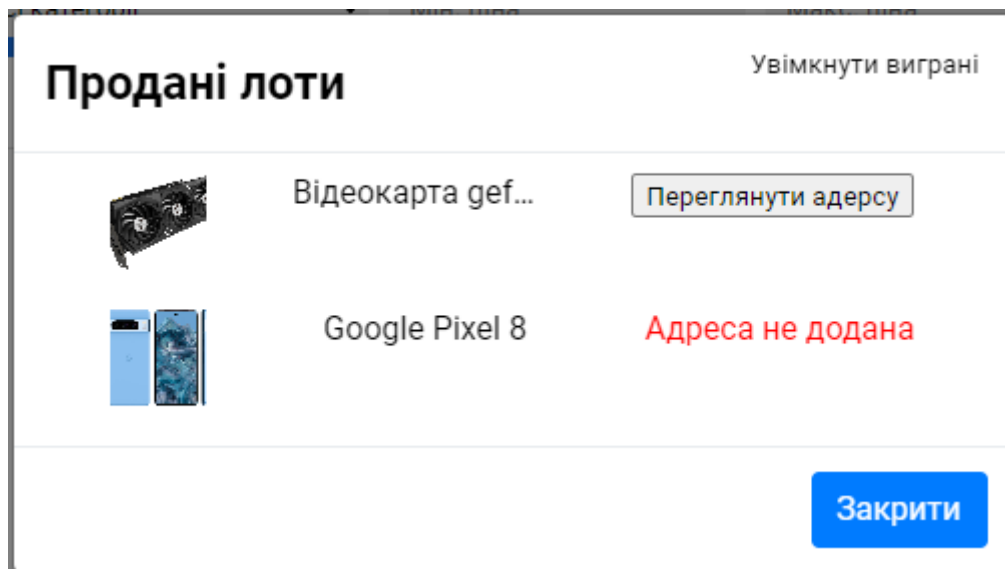


Рис. 3.14 – Вікно перегляду проданих товарів

Після натискання кнопки «Переглянути адресу» користувач потрапляє до вікна (рис. 3.15), де має можливість переглянути адресу та контактний номер покупця. Натиснувши кнопку «Позначити надісланим» продавець підтверджує, що надіслав посилку за заданою адресою.

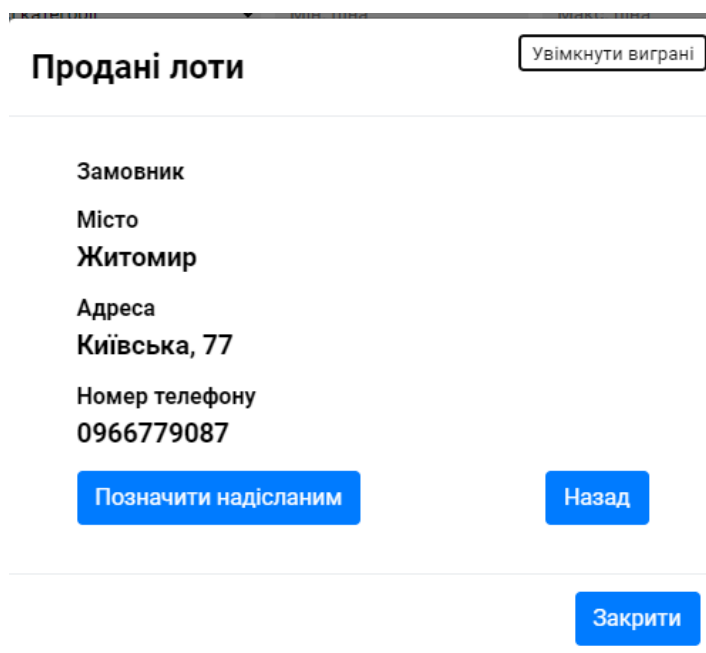


Рис. 3.15 – Вікно з адресою покупця

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

В профілі користувач може побачити усі лоти, в яких він переміг, усі лоти, в яких зробив ставку, в яких його ставку перебили та лоти, в яких він програв. Усі ці поля відображаються, лише у разі якщо вони можуть існувати. На скріншоті (рис. 3.16) користувача немає жодного лоту, в якому він програв, або в якому його ставку перебили, тому ці поля не відображаються.

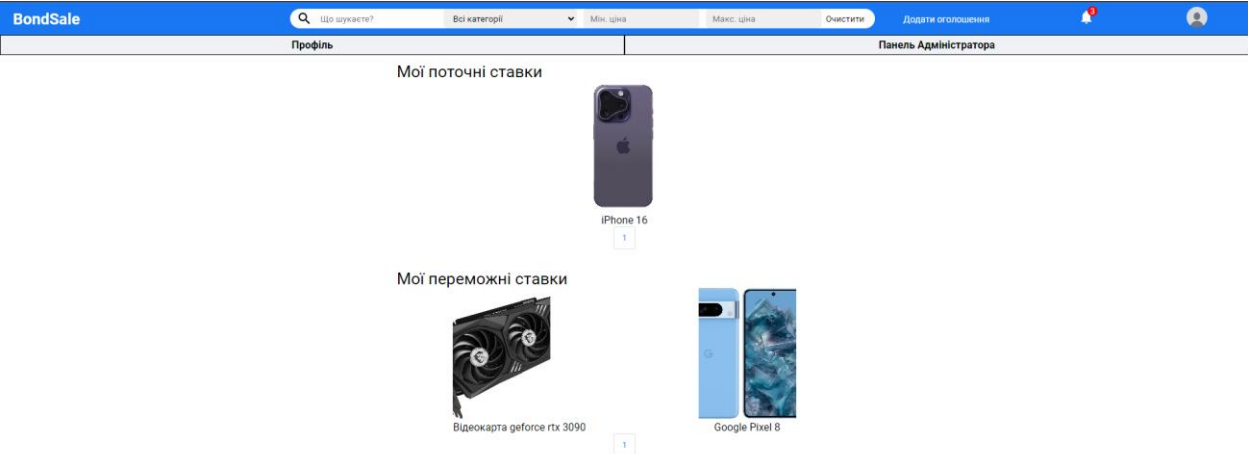


Рис. 3.16 – Екран профілю

3.3 Тестування платформи

Розробка будь-якого продукту завжди супроводжується помилками, що можуть з’явитися на етапі розробки, Для того, щоб додаток працював стабільні і без помилок варто виконувати його тестування. В процесі тестування перевіряються як вхідні, так і вихідні дані. До таких даних можна відносити логіни та паролі користувачів, або інформацію що повертає сервер на запити. Метою тестування є виявлення можливих помилок і дефектів, що виникли на етапі розробки, але не були помічені розробником.

По-перше, буде розроблено декілька тестових випадків (табл. 3.1). Тестовий випадок це перелік умов та дій, що мають бути виконані для отримання певного, відомого результату. Отриманий результат має бути порівняний з тим, який очікувалося і в залежності від результату буде виявлено, чи відбулася помилка в роботі програми [12].

По-друге, буде проведена робота над помилками, що будуть виявлені в процесі тестування, за-для покращення користувацького досвіду при роботі з сайтом, а також для усунення можливий помилок, що зроблять платформу недоступною.

Серед тестових випадків будуть розглянуті основні можливості платформи, такі як реєстрація користувачів, створення оголошень та участь в аукціонах. Усі тестові випадку будуть перевірені як на роботу з помилками, так і на роботу з даними, що відповідають вимогам поставленим при проведенні тесту.

Таблиця 3.1

Тестові випадки для платформи онлайн-оголошень з елементами аукціону

Test Case Description	Test Steps	Test Data	Expected Result	Actual Results	Pass/Fail
Перевірка авторизації з дійсними даними	1. Перейти на сторінку авторизації 2. Ввести логін 3. Ввести пароль 4. Натиснути «Увійти»	Email: sasha@gmail.com Password: 11111111	Має відбутися успішний вхід до системи	Як і очікувалося	Пройдено
Перевірка авторизації з недійсними параметрами	1. Перейти на сторінку авторизації 2. Ввести логін 3. Ввести пароль 5. Натиснути «Увійти»	Email: sasha.admin@gmail.com Password: 11111111	Має з'явитися повідомлення, що користувача не існує	Як і очікувалося	Пройдено

Продовження таблиці 3.1

Перевірка авторизації з пустими полями	1. Перейти на сторінку авторизації 2. Не вводити дані користувача 3. Натиснути «Увійти»	Email: Password:	Має з'явитися повідомлення про потребу в заповненні полів	Як і очікувалося	Пройдено
Перевірка створення оголошення	1. Відкрити вікно створення замовлення 2. Ввести всі дані про оголошення 3. Додати картинку 4. Натиснути «Розмістити оголошення»	Назва: мобільний телефон Ціна викупу: 15000 Мінімальна ставка: 1000 Дата завершення: 17.06.2024 Категорія: техніка	Має з'явитися повідомлення, що оголошення відправлено на верифікацію	Як і очікувалося	Пройдено
Перевірка створення ставки	1. Відкрити вікно оголошення 2. Натиснути кнопку «Зробити ставку»		Має з'явитися повідомлення, що ваша ставка принята	Як і очікувалося	Пройдено

Висновки до третього розділу

У цьому розділі було проведено аналіз і проектування фізичної структури платформи онлайн-оголошень з елементами аукціону за допомогою діаграм компонентів і розгортання. Після цього було проведено процес тестування створеного додатку і було описано користувацький інтерфейс додатку та надана інструкція роботи з інтерфейсом.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В першому розділі кваліфікаційної роботи було проведено дослідження технічного завдання. Далі було проведено порівняльний аналіз розроблюваного платформи з його аналогами та майбутніми конкурентами. Було створено таблицю з порівнянням усіх платформ. В кінці розділу були розроблені діаграми варіантів використання, використовуючи технології для розробки діаграм.

В другому розділі було детально описано акторів, які взаємодіють з платформою, та були розглянуті алгоритми роботи даних акторів з платформою. Алгоритми роботи були представлені у вигляді відповідних діаграм активностей. Також була розроблена база даних та проілюстрована у вигляді діаграми класів. Наприкінці розділу були досліджені комунікації і послідовності взаємодії між об'єктами платформи і проведена розробка платформи.

Третій розділ почався з проектування діаграм компонентів та класів та було описано порядок встановлення і розгортання платформи. Далі було розроблено та проведено декілька тестових випадків, для виявлення і усунення помилок в роботі платформи. Розділ було закінчено детальним описом інтерфейсу користувача та була надана інструкції до роботи ним.

В результаті роботи над кваліфікаційною роботою було спроектовано та розроблено програмний продукт, що відповідає усім зазначеним вимогам. Протягом виконання завдань я отримала низку нової інформації та засвоїла знання, що отримала протягом усього навчання

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фрімен Е. Head First. Програмування на JavaScript / Е. Фрімен, Р. Елізабет. – 2022. – 672 с.
2. Платформа оголошень OLX [Електронний ресурс] – Режим доступу до ресурсу: <https://www.olx.ua/uk/>.
3. Платформа оголошень [Електронний ресурс] – Режим доступу до ресурсу: <https://prom.ua/>.
4. Платформа оголошень Ebay [Електронний ресурс] – Режим доступу до ресурсу: <https://by.ebay.com/>.
5. What is UML diagram? [Електронний ресурс] – Режим доступу до ресурсу: <https://miro.com/diagramming/what-is-a-uml-diagram/>.
6. NPM JSONWEBTOKEN [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/jsonwebtoken>.
7. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/getting-started.html>.
8. NPM React Router DOM [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/react-router-dom>.
9. NPM yup [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/yup>.
10. NPM axios [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/axios>.
11. Документація MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>.
12. Що таке Test Case [Електронний ресурс] – Режим доступу до ресурсу: <https://qalight.ua/ru/baza-znaniy/test-case/>.
13. Jürgen W. HTML and CSS: The Comprehensive Guide Paperback / Wolf Jürgen., 2023. – 814 с.
14. David G. React Cookbook: Recipes for Mastering the React Framework. 1st Ed. / G. David, G. Dawn., 2021. – 500 с.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Концедайло В.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

15. Node.js in Action / M.Cantelon, M. Harter, T. Holowaychuk, N. Rajlich., 2013. – 416 с.
16. Що таке діаграма компонентів UML: символи та посібник [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mindonmap.com/uk/blog/uml-component-diagram/>.
17. Діаграма розгортання: Підручник з UML із ПРИКЛАДОМ [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/deployment-diagram-uml-example.html>.
18. Діаграма послідовності (Sequence Diagrams) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.maxzosim.com/sequence-diagrams/>.
19. Інструкція, як будувати UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/40575/>.
20. Дізнайтеся все про діаграму активності UML [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mindonmap.com/uk/blog/uml-activity-diagram/>.
21. Shelley S. Learning Node: Moving to the Server-Side 2nd Edition / Shelley Shelley., 2016. – 288 с.
22. MacDonald M. HTML5: The Missing Manual, 2nd Edition The Book That Should Have Been in the Box / Matthew MacDonald., 2014. – 518 с.
23. Wolf J. HTML and CSS: The Comprehensive Guide / Jurgen Wolf., 2023. – 814 с.
24. Node.js v22.2.0 documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
25. NPM multer [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/multer>.
26. Express documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>.
27. JavaScript Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.w3schools.com/js/>.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				49
Змн.	Арк.	№ докум.	Підпис	Дата		

28. Що таке CRUD простими словами: функції, переваги та приклади [Електронний ресурс] – Режим доступу до ресурсу: <https://highload.today/uk/shho-take-crud-prostimi-slovami-funktsiyi-perevagi-ta-prikladi/>.

29. Міннік К. JavaScript All-in-One For Dummies / Кріс Міннік., 2023. – 816 с.

30. MERN Stack Explained [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/resources/languages/mern-stack>.

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				50
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу «user.js»

```

const User = require("../models/User")
const Lot = require("../models/Lot")
const router = require('express').Router()
const auth = require("../middleware/auth.middleware")
const fs = require('fs').promises
const path = require('path');

//ОТРИМАННЯ ДАНИХ ПРО КОРИСТУВАЧА
router.get("/", auth, async (req, res)=>{
  try{
    const user = await User.findById(req.user.userId);
    res.status(200).json(user)
  }catch (err){
    res.status(500).json(err)
  }
})

router.get('/mywon', auth, async (req, res) => {
  try{
    const lots = await Lot.find({saled: true, bidderId: req.user.userId})
    console.log(lots)
    res.status(200).json(lots)
  }catch(err){
    res.status(500).json(err)
  }
})

router.get('/mylostactive', auth, async (req, res) => {
  try{
    const lots = await Lot.find({
      saled: false,
      bidders: {$in: [req.user.userId]},
      bidderId: {$ne: req.user.userId}
    })
    res.status(200).json(lots)
  }catch(err){
    res.status(500).json(err)
  }
})

router.get('/mylostsaled', auth, async (req, res) => {
  try{
    const lots = await Lot.find({
      saled: true,
      bidders: {$in: [req.user.userId]},

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Кониедайло В.В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        bidderId: {$ne: req.user.userId}))
    res.status(200).json(lots)
  }catch(err){
    res.status(500).json(err)
  }
})

router.get('/mycurrent', auth, async (req, res) => {
  try{
    const lots = await Lot.find({saled: false, bidderId: req.user.userId})
    res.status(200).json(lots)
  }catch(err){
    res.status(500).json(err)
  }
})

router.get('/admin', auth, async (req, res) => {
  try{
    const user = await User.findById(req.user.userId)
    if(!user.isAdmin) return res.status(400).json({message: "У вас немає прав адміністратора для доступу на сторінку"})
    const lots = await Lot.find({accepted: false})
    res.status(200).json(lots)
  } catch(err){
    res.status(500).send(err)
  }
})

router.post('/admin/accept', auth, async (req, res) => {
  try{
    const user = await User.findById(req.user.userId)
    if(!user.isAdmin) return res.status(400).json({message: "У вас немає прав адміністратора для доступу на сторінку"})
    const lot = await Lot.findById(req.body.lotId)
    lot.accepted = true
    await lot.save()
    res.status(200).json({message: "Товар затверджено"})
  }catch(err){
    res.status(500).send(err)
  }
})

router.post('/admin/delete', auth, async (req, res) => {
  try{
    const user = await User.findById(req.user.userId)
    const lot = await Lot.findById(req.body.lotId)
    const filePath = path.join(__dirname,
    `../../client/public/imgs/`+lot.imgs)
    console.log(filePath)
  }
})

```

		Бондарчук О.А.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Кониедайло В.В				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(!user.isAdmin && user._id !== lot.ownerId) return
        res.status(400).json({message: "У вас немає прав для видалення товару"})
        await Lot.deleteOne({_id:req.body.lotId})
        await fs.unlink(filePath)
        res.status(200).json({message: "Лот видалено"})
    } catch(err){
        res.status(500).send(err)
    }
})

router.get('/bought', auth, async (req, res) => {
    try{
        const lots = await Lot.find({bidderId: req.user.userId,
                                     saled: true,
                                     withAddress: false

        })
        console.log(lots)
        res.status(200).json(lots)
    }catch(err){
        res.status(500).json(err)
    }
})

router.get('/saled', auth, async(req, res) => {
    try{
        const lots = await Lot.find({
            ownerId: req.user.userId,
            saled: true,
            mailed: false
        })
        console.log(lots)
        res.status(200).json(lots)
    }catch(err){
        res.status(500).json(err)
    }
})

module.exports = router

```

Лістинг файлу «address.js»

```

const router = require('express').Router()
const Address = require('../models/Address')
const Lot = require('../models/Lot')
const auth = require("../middleware/auth.middleware")

router.post("/", auth, async (req, res) => {
    try{
        console.log(req.body.lotId)
    }

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const newAddres = new Address({
      name: req.user.username,
      lotId: req.body.lotId,
      city: req.body.city,
      address: req.body.address,
      PhoneNumber: req.body.PhoneNumber
    })
    console.log(newAddres)
    const lot = await Lot.findById(req.body.lotId)
    console.log(lot)
    lot.withAddress = true
    console.log(lot)

    await lot.save()
    await newAddres.save()
    res.status(200).json({message: 'Адресу додано, очікуйте відправлення'})

  } catch(err){
    res.status(500).json(err)
  }
})

router.get('/', auth, async (req, res) => {
  try{
    const address = await Address.find()
    res.status(200).json(address)
  }catch(err){
    res.status(500).json(err)
  }
})

module.exports = router

```

Лістинг файлу «index.js»

```

const express = require("express")
const mongoose = require("mongoose")
const dotenv = require("dotenv")
const helmet = require("helmet")
const morgan = require("morgan")
const cors = require('cors')

const Lot = require('./models/Lot')

const authRoute = require("./routes/auth")
const userRoute = require("./routes/user")
const lotRoute = require("./routes/lot")
const categoryRoute = require("./routes/category")
const addressRoute = require('./routes/address')

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const app = express()

dotenv.config()

mongoose.connect(process.env.MONGO_URL).then(()=>
{console.log("DB connected")},
err => {console.log(err)}
)

const interval = 10 * 1000

const func = async () => {
  const lot = await Lot.find({accepted:true, saled:false})
  const currentDate = new Date().toISOString()
  await Lot.updateMany(
    {endOfLot: {$lt: currentDate}},
    {$set: {saled: true}}
  ).then(() => {console.log("OK")})
}

setInterval(func, interval)

app.use(express.json())
app.use(helmet())
app.use(morgan("dev"))
app.use(cors())

app.use("/api/auth", authRoute)
app.use("/api/user", userRoute)
app.use("/api/lot", lotRoute)
app.use("/api/category", categoryRoute)
app.use('/api/address', addressRoute)

app.listen(4000, ()=>{
  console.log("Server started")
})

```

Лістинг файлу «App.js»

```

import { useAuth } from "./hooks/auth.hook";
import {useRoutes} from "./routes"
import { BrowserRouter as Router } from "react-router-dom";
import { AuthContext } from "./context/AuthContext";
import 'materialize-css'
import { useState } from "react";

function App() {
  const {token, login, logout, userId} = useAuth()
  const [name, setName] = useState('')

```

		Бондарчук О.А.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Кониєдайло В.В				56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const [category, setCategory] = useState("")
const [minPrice, setMinPrice] = useState('')
const [maxPrice, setMaxPrice] = useState('')
const isAuth = !!token
const routes = useRoutes(isAuth)

return (
  <AuthContext.Provider value={{token, login, logout, userId, isAuth,
    name, setName, category, setCategory, minPrice, setMinPrice, maxPrice,
    setMaxPrice}}>
    <Router>
      <div>
        {routes}
      </div>
    </Router>
  </AuthContext.Provider>
)
}

export default App;

```

Лістинг файлу «register.jsx»

```

import React, { useState, useEffect } from "react";
import axios from "axios";
import * as yup from "yup";
import "./register.css";
import { Link, Navigate } from "react-router-dom";
import Popup from "../../components/popup/popup";
import Notification from "../../components/notification/notification";

export default function Register() {
  const [login, setLogin] = useState("");
  const [mail, setMail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [errors, setErrors] = useState([]);
  const [redirect, setRedirect] = useState(false);
  const [avatar, setAvatar] = useState(null);
  const [img, setImg] = useState("/imgs/profile/default.jpg");
  const [message, setMessage] = useState(null)

  const schema = yup.object().shape({
    login: yup.string().required('Поле "Логін" має бути заповнене'),
    mail: yup.string().email("Невірний формат пошти").required("Поле 'Пошта' має бути заповнене"),
    password: yup.string().min(8, "Пароль має містити більше 8 символів").required("Поле 'Пароль' має бути заповнене"),
    confirmPassword: yup.string().oneOf([password], "Паролі не співпадають").required("Поле 'Повторіть пароль' має бути заповнене"),
  });
}

```

		Бондарчук О.А.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Кониедайло В.В.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const handleRegister = async () => {
  try {
    await schema.validate({ login, mail, password, confirmPassword }, {
      abortEarly: false });
    setErrors([]);
    const formData = new FormData();
    formData.append("avatar", avatar)
    formData.append("username", login)
    formData.append("email", mail)
    formData.append("password", password)
    for (const entry of formData.entries()) {
      console.log(entry[0], entry[1]);
    }
    await axios.post("http://localhost:4000/api/auth/register", formData, {
      headers: {
        "Content-Type": "multipart/form-data",
      },
    }).then(res => {
      setRedirect(true)
      const data = res.response.data.message
      setMessage(data)
    }).catch(error => {
      const data = error.response.data.message
      setMessage(data)
    });
  } catch (error) {
    setErrors(error.errors);
  }
};

useEffect(() => {
  window.M.updateTextFields()
}, [])

const handleFileChange = (event) => {
  setAvatar(event.target.files[0]);
};

return (
  <div className="login">
    <div className="loginWrapper">
      <div className="loginLeft">
        <h3 className="loginLogo">BondSale</h3>
        <span className="loginDesc">
          Виставляйте оголошення. Купуйте, продавайте, торгуйтесь.
        </span>
      </div>
      <div className="loginRight">
        <div className="loginBox">
          <div className="imgContainer">

```

		Бондарчук О.А.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Кониедайло В.В.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <label className="" htmlFor="fileInput">
        {avatar ? (
            <img className="registerImg" src={URL.createObjectURL(avatar)}
alt="Avatar Preview" />
        ) : (
            <img className="registerImg" src={img} alt="" />
        )}
        </label>
        <input
        id="fileInput"
        type="file"
        name="avatar"
        className="input-file"
        onChange={handleFileChange}
        />
    </div>
    <input value={login} onChange={(event) =>
setLogin(event.target.value)} placeholder="Логін" className="loginInput" />
    <input value={mail} onChange={(event) => setMail(event.target.value)}
placeholder="Пошта" className="loginInput" />
    <input type="password" value={password} onChange={(event) =>
setPassword(event.target.value)} placeholder="Пароль" className="loginInput" />
    <input type="password" value={confirmPassword} onChange={(event) =>
setConfirmPassword(event.target.value)} placeholder="Повторити пароль"
className="loginInput" />
    <button onClick={handleRegister}
className="loginButton">Зареєструватися</button>
    {errors && errors.length > 0 && <Popup errors={errors} />}
    <Link to="/login" className="loginRegisterButton">Увійти в свій
аккаунт</Link>
    {redirect && <Navigate replace to="/login"/>}
    {message && (<Notification notification={message}
setNotification={setMessage}/>)}
    </div>
    </div>
    </div>
    </div>
    );
}

```

Лістинг файлу «register.css»

```

.login{
    width: 100vw;
    height: 100vh;
    background-color: #f0f2f5;
    display: flex;
    align-items: center;
    justify-content: center;
}

.loginWrapper{

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        width: 70%;
        height: 70%;
        display: flex;
    }

    .loginLeft, .loginRight{
        flex:1;
        display: flex;
        flex-direction: column;
        justify-content: center;
    }
    .loginLogo{
        font-size: 50px;
        font-weight: 800;
        color: #1775ee;
        margin-bottom: 10px;
    }
    .loginDesc{
        font-size: 24px;
    }
    }

    .loginBox{
        height: 500px;
        padding: 20px;
        background-color: white;
        border-radius: 10px;
        display: flex;
        flex-direction: column;
        justify-content: space-between;
    }

    .loginInput{
        height: 50px;
        border-radius: 10px;
        border: 1px solid gray;
        padding-left: 20px;
    }
    }

    .loginInput:focus{
        outline: none;
    }
    }

    .loginButton{
        height: 50px;
        border-radius: 10px;
        border: none;
        background-color: #1775ee;
        color: white;
        font-size: 20px;
        font-weight: 500;
    }

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Кониедайло В.В				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        cursor: pointer;
    }
    .loginForgot{
        text-align: center;
        color: #1775ee
    }
    .loginRegisterButton{
        width: 50%;
        align-self: center;
        height: 50px;
        border-radius: 10px;
        border: none;
        background-color: #42b72a;
        color: white;
        font-size: 20px;
        font-weight: 500;
        cursor: pointer;
        text-decoration: none; /* Для удаления стандартного подчеркивания ссылки */
        display: flex; /* Чтобы кнопка оставалась inline элементом */
        justify-content: center; /* Чтобы текст кнопки оставался по центру */
        align-items: center; /* Чтобы текст кнопки оставался по центру */
    }

    .registerImg{
        width: 64px;
        height: 64px;
        cursor: pointer;
        border-radius: 25px;
    }

    .imgContainer{
        display: flex;
        justify-content: center;
        align-items: center;
    }

```

Лістинг файлу «main.jsx»

```

import Topbar from "../../components/topbar/topbar"
import { useEffect, useState, useContext } from "react"
import axios from "axios"
import { AuthContext } from "../../context/AuthContext";
import './main.css'
import Lot from '../../components/card/Card'
import Pagination from "../../components/pagination/pagination";

export default function Main(){

    const [lots, setLots] = useState([])
    const [loading, setLoading] = useState(false)
    const [currentPage, setCurrentPage] = useState(1)
    const [filteredLots, setFilteredLots] = useState([])

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Кониедайло В.В				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const [lotsPage] = useState(8)

const { token, name, category, minPrice, maxPrice } =
useContext(AuthContext);

async function fetchLots(){
  try{
    const res = await axios.get("http://localhost:4000/api/lot/", {
      headers: {
        Authorization: `Bearer ${token}`
      }
    });
    setLots(res.data)
    filterLots(name, category, minPrice, maxPrice)
  } catch (err){
    console.error("Помилка при завантаженні: ", err)
  }
}

const lastLotIndex = currentPage * lotsPage
const firstLotIndex = lastLotIndex - lotsPage
const currentLot = filteredLots.slice(firstLotIndex, lastLotIndex)

const paginate = pageNumber => setCurrentPage(pageNumber)

useEffect(()=>{
  if (!name && !category && !minPrice && !maxPrice){
    setLoading(true)
    fetchLots()
    setLoading(false)

    const intervalId = setInterval(() => {
      fetchLots()
    }, 5000)

    return () => {
      clearInterval(intervalId)
    }
  }
}, [])

const filterLots = (title, category, minPrice, maxPrice) => {
  const filtered = lots.filter(lot => {
    const titleMatch = title ?
lot.title.toLowerCase().includes(title.toLowerCase()) : true;
    const categoryMatch = category ? lot.category === category : true;
    let priceMatch = true;

    if (minPrice || maxPrice) {

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        const currentBid = lot.currentBid || lot.minBid;
        if (currentBid) {
            priceMatch = (!minPrice || currentBid >= minPrice) &&
(!maxPrice || currentBid <= maxPrice);
        }
    }

    return titleMatch && categoryMatch && priceMatch;
});

setFilteredLots(filtered);
};

useEffect(() => {
    if(name || category || minPrice || maxPrice){
        filterLots(name, category, minPrice, maxPrice)
    } else {
        setFilteredLots(lots)
    }
}, [name, category, maxPrice, minPrice, fetchLots])

return(
    <div>
        <Topbar/>
        <div className="content__container">
            {loading ? (
                <div>Завантаження...</div>
            ) : (
                <div className="card__containe">
                    {
                        currentLot.map((lot) => (
                            <Lot key={lot._id} lot={lot}
fetchLots={fetchLots} />
                        ))
                    }
                </div>
            )}
        </div>
        <Pagination
            lotsPage={lotsPage}
            totalLots={lots.length}
            paginate={paginate}
        />
    </div>
)
}

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				63
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу «topbar.jsx»

```
import './topbar.css';
import Search from '../icons/search.png';
import { Link } from "react-router-dom";
import { useState, useContext, useEffect } from "react";
import ModalLot from "../modal/ModalLot";
import { AuthContext } from "../context/AuthContext";
import axios from "axios";
import { ModalBought } from "../modal/ModalBought";

export default function Topbar() {
  const auth = useContext(AuthContext);
  const [modalActive, setModalActive] = useState(false);
  const [profilePic, setProfilePic] = useState("/imgs/profile/default.jpg");
  const [categories, setCategories] = useState([]);

  async function fetchUser() {
    try {
      const res = await axios.get("http://localhost:4000/api/user/", {
        headers: {
          Authorization: `Bearer ${auth.token}`
        }
      });
      if (res.data.profilePicture) {
        setProfilePic("/imgs/profile/" + res.data.profilePicture);
      }
    } catch (err) {
      console.error("Помилка при завантаженні: ", err);
    }
  }

  async function fetchCategories() {
    try {
      const res = await axios.get("http://localhost:4000/api/category/", {
        headers: {
          Authorization: `Bearer ${auth.token}`
        }
      });
      setCategories(res.data);
    } catch (error) {
      console.error("Error fetching categories:", error);
    }
  }

  const handleDelete = () => {
    auth.setName('')
    auth.setCategory('')
    auth.setMaxPrice('')
    auth.setMinPrice('')
  }
}
```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Кониедайло В.В				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

useEffect(() => {
    fetchUser()
    fetchCategories()
}, []);

useEffect(() => {
    console.log(auth.name);
    console.log(auth.category)
}, [auth.name, auth.category]);

return (
    <div className="topbarContainer">
        <div className="topbarLeft">
            <Link className="logoLink" to="/">
                <span className="logo">BondSale</span>
            </Link>
        </div>
        <div className="topbarCenter">
            <div className="searchbar">
                <img src={Search} alt="" className="searchIcon"/>
                <input
                    placeholder="Що шукаєте?"
                    className="searchInput"
                    value={auth.name}
                    onChange={(e) => auth.setName(e.target.value)}
                />
                <select className="categorySelect" value={auth.category}
                    onChange={(e) => auth.setCategory(e.target.value)}>
                    <option value=''>Всі категорії</option>
                    {
                        categories.map((category) => (
                            <option
                                value={category._id}>{category.title}</option>
                        ))
                    }
                </select>
                <input
                    type="number"
                    placeholder="Мін. ціна"
                    className="priceInput"
                    value={auth.minPrice}
                    onChange={(e) => auth.setMinPrice(e.target.value)}
                />
                <input
                    type="number"
                    placeholder="Макс. ціна"
                    className="priceInput"
                    value={auth.maxPrice}
                    onChange={(e) => auth.setMaxPrice(e.target.value)}
                />
            </div>
        </div>
    </div>
)

```

		Бондарчук О.А.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Кониедайло В.В.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <button className="clearButton"
        style={{border:'none',
            backgroundColor: 'white',
            cursor: 'pointer'
        }}
        onClick={handleDelete}>
            ОЧИСТИТИ
        </button>
    </div>
</div>
<div className="topbarRight">
    <div className="topbarLinks">
        <button onClick={() => setModalActive(true)}
className="topbarButton">Додати оголошення</button>
    </div>
    <div className="topbarIcons">
        <div className="topbarIconItem">
            <ModalBought/>
        </div>
    </div>
    <Link to="/profile">
        <img src={profilePic} alt="" className="topbarImg"/>
    </Link>
</div>
    <ModalLot active={modalActive} setActive={setModalActive} />
</div>
    );
}

```

		Бондарчук О.А.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Кониедайло В.В				66
Змн.	Арк.	№ докум.	Підпис	Дата		