

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОДАЖУ РЕЧЕЙ ЗА
КОНКУРЕНТНОЮ ЦІНОЮ
SOFTWARE FOR SELLING GOODS AT A COMPETITIVE PRICE

Виконав: студент(ка) 4 курсу, групи КНТ-217
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Буренко В.М.

(прізвище та ініціали)

Керівник Колпакова Т.О.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Буренка Владислава Миколайовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення продажу речей за конкурентною ціною

Software for Selling Goods at a Competitive Price

керівник проєкту (роботи) Колпакова Тетяна Олексіївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 17.05.2021

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми.

3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Колпакова Т.О., доцент		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “17” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Буренко В.М.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Колпакова Т.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 77 с., 27 рис., 1 табл., 3 дод., 14 джерел.

КОНКУРЕНТНА ЦІНА, ПРОДАЖ РЕЧЕЙ, АУКЦІОН, ВЕБЗАСТОСУНОК, БАЗА ДАНИХ, ASP.NET.

Об'єкт дослідження – процес продажу речей за конкурентною ціною.
Предмет дослідження – програмне забезпечення продажу речей за конкурентною ціною.

Мета роботи – розроблення програмного забезпечення продажу речей для встановлення конкурентної ціни на основі попиту покупців.

Матеріали, методи та технічні засоби: мова програмування C#, система управління базами даних Microsoft SQL Server.

Результати. Виконано аналіз проблеми продажу речей за конкурентною ціною. Представлено архітектуру програмного забезпечення, у процесі визначено функціональні вимоги до програми, обрано технології проєктування і розроблення програми, виконано проєктування структури даних. Реалізовано програмне забезпечення, представлене вебзастосунком, створеним за допомогою технології ASP.NET, описано структурні елементи, створені під час реалізації програми.

Висновки. Створений вебзастосунок надає можливість визначення речей, які продає користувач, створення оголошень на продаж, створення пропозицій на придбання, автоматичного визначення пропозиції, яка перемогла, або набору таких пропозицій, відстеження власних пропозицій, придбань та пропозицій на власні оголошення.

Галузь використання. Організація індивідуальних продажів, створення платформи для взаємодії продавців та покупців.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 77 pages, 27 figures, 1 table, 3 appendixes, 14 sources.

COMPETITIVE PRICE, SELLING GOODS, AUCTION, WEB APPLICATION, DATABASE, ASP.NET.

The object of research is the process of selling goods at a competitive price. The subject of research is software for selling goods at a competitive price.

The purpose of the work is to develop software for selling goods to set a competitive price based on customer demand.

Materials, methods and technical means: C# programming language, Microsoft SQL Server database management system.

Results. The analysis of the problem of selling goods at a competitive price is performed. The software architecture is presented. The functional requirements to the software are defined, technologies of designing and development of the program are chosen, designing of data structure is executed. The implemented software is represented by a web application created using ASP.NET technology. The structural elements created during the implementation of the software are presented.

Conclusions. The web application allows users to identify goods that the user is selling, create sales ads, create purchase offers, automatically determine the winning offer or set of such offers, track own offers, purchases and offers for own ads.

Field of usage. Organization of individual sales, creating a platform for interaction between sellers and buyers.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Організація продажу речей за конкурентною ціною	10
1.2 Програмні засоби продажу речей за конкурентною ціною	11
1.3 Висновки за розділом 1	15
2 Розробка архітектури програми.....	16
2.1 Функціональні вимоги до програмного забезпечення	16
2.2 Вибір засобів розробки	19
2.3 Проектування бази даних	22
2.4 Висновки за розділом 2	28
3 Розробка програми	29
3.1 Алгоритм функціонування програмного забезпечення	29
3.2 Структура реалізованого програмного забезпечення	32
3.3 Опис прийнятих під час реалізації рішень	34
3.4 Висновки за розділом 3	41
4 Експлуатація, тестування та експериментальне дослідження програми	42
4.1 Призначення програми	42
4.2 Умови виконання програми	42
4.3 Виконання програми.....	42
Висновки	48
Перелік джерел посилання	49
Додаток А Технічне завдання	51
Додаток Б Текст програми	55
Додаток В Слайди презентації.....	71

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

СУБД – система управління базами даних.

ВСТУП

Не завжди масові продажі товарів широкого вжитку є вирішенням проблеми пошуку необхідних речей для кожної людини. При стандартній моделі продавець самостійно встановлює вартість, а покупець сам обирає, в якого продавця він бажає придбати необхідні йому речі. Це за рахунок пропозицій різних продавців та здатності покупців сплатити певну суму визначає пропозицію, яку придбає кожен конкретний покупець. З іншого боку на вартість можуть впливати і покупці. Це відбувається в іншій моделі взаємодії. Це аукціонна модель. Аукціонна модель передбачає конкуренцію між покупцями за право придбати товар. Речі, які виставляються на аукціоні, можуть бути унікальними, можуть мати знижену вартість, все це призводить до того, що не зважаючи на велику конкуренцію звичайних інтернет-магазинів, велику популярність мають також і електронні аукціони.

Електронні аукціони, що пропонують нові товари широкого вжитку, зафіксували сплеск популярності, коли нове покоління молодих учасників торгів почало шукати вигідні пропозиції в мережі Інтернеті: серед тих, хто бере участь в аукціонах, є покупці та орендодавці, які вперше знаходяться в пошуку заощаджень на обладнанні своїх будинків новими меблями, технікою та гаджетами, які іноді можна придбати за зниженими цінами, загалом у світі такі аукціони можуть призвести до економії до 75 відсотків на нових або забруднених магазинами меблях, техніці та гаджетах, але головною причиною великого зростання протягом останнього періоду є не розмір вигідних пропозицій, а перехід від традиційного аукціону до аукціонів, що проводяться в Інтернеті, зі вступом аукціонних будинків у цифрову еру покупці, які вперше купують товари, все частіше цікавляться ними [1].

Електронні аукціони можуть реалізовуватися для обмеженого кола користувачів або за рахунок обмеженого кола користувачів, коли продаж речей, наприклад, реалізується на конкретному підприємстві, в конкретній спільноті людей. Тоді така спільнота потребує власного програмного

забезпечення для підтримки даного процесу. Дана проблема має бути розв'язана в дипломній кваліфікаційній роботі за рахунок розроблення відповідного програмного рішення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Організація продажу речей за конкурентною ціною

Продаж речей за конкурентною ціною реалізується у формі аукціонів. На даний момент найактивнішою формою проведення аукціонів для звичайних покупців є електронні аукціони. Вони ж дозволяють галузі розвиватися в бік привертання нових покупців.

Аукціон – це система купівлі-продажу товарів чи послуг у формі виставлення їх для торгів. дозволяючи людям робити ставки та продавати учасникам, які пропонують найкращі пропозиції, відносно даної форми продажу можна відзначити наступне:

- учасники торгів конкурують між собою, при цьому кожна наступна заявка перевищує попередню ставку: після того, як товар буде виставлений на продаж, аукціон розпочне роботу за відносно низькою ціною, щоб залучити велику кількість учасників торгів, ціна зростає кожного разу, коли хтось робить нову, більш високу ставку, поки нарешті жоден учасник не бажає запропонувати більше, ніж остання заявка, і учасник, який встановив найбільшу ціну, купує товар, аукціон вважається закінченим, коли продавець приймає найвищу запропоновану пропозицію, а покупець оплачує товари чи послуги та заволодіває ними;

- аукціони часто вважаються синонімами продажу антикваріату, рідкісних предметів колекціонування та картин, вони також використовуються в інвестиційному банкінгу: інвестиційні банкіри використовують аукціони, щоб залучити якнайвищу ціну при продажу компанії, процес починається із запрошення декількох покупців на аукціон, конкурентні торги призводять до того, що учасники піднімають ціну вище, що дозволяє банку максимізувати свій прибуток від продажу [2].

У даній роботі передбачається використання форми електронного аукціону, коли продавець не зобов'язаний занижувати початкову вартість, а потенційно при цьому може отримати кращу пропозицію. Аукціон

реалізується автоматично, можна виставляти не тільки одну річ, але і декілька екземплярів речей. Усе це призводить до того, що продавець може під себе сформулювати необхідні йому початкові умови, а потім на основі торгів виносяться остаточні рішення. Розігрівання торгів реалізується за рахунок встановлення остаточної дати.

1.2 Програмні засоби продажу речей за конкурентною ціною

Існують різні вебсайти, які підтримують організації продажу речей за конкурентною ціною на даний момент.

ShopGoodwill [3] (рис. 1.1) – це неприбутковий аукціонний сайт, на якому перелічуються та продаються товари для збору коштів для людей, які живуть з обмеженими можливостями, аукціон проводиться у співпраці з магазинами Goodwill по всій території Сполучених Штатів Америки, що дозволяє продавцям Goodwill продавати товари в Інтернеті та отримувати гроші для благодійної організації Goodwill [4]. Деякі з категорій предметів, що виставляються на аукціон включають товари для дому, елементи мистецтва, інструменти, речі, потрібні для організації весіль, одяг, музичні інструменти, товари для домашніх тварин, комп'ютери та електроніку, спортивні товари, іграшки [5].

На сайті ShopGoodwill можна визначити власну пропозицію щодо придбання речі, яка є лотом аукціону, для цього потрібно визначити вартість та підтвердити власну пропозицію. Для кожної зміни вартості встановлено мінімальний поріг. Щодо пропозицій інших користувачів відображається тільки поточна вартість, можлива мінімальна зміна вартості та кількість пропозицій інших користувачів. Тобто отримати інформацію про конкретні зміни вартості неможливо. Нemoжливо також і отримати інформацію про початкову вартість речі. На кожну річ може демонструватися цілий набір фотографій, що її характеризують.

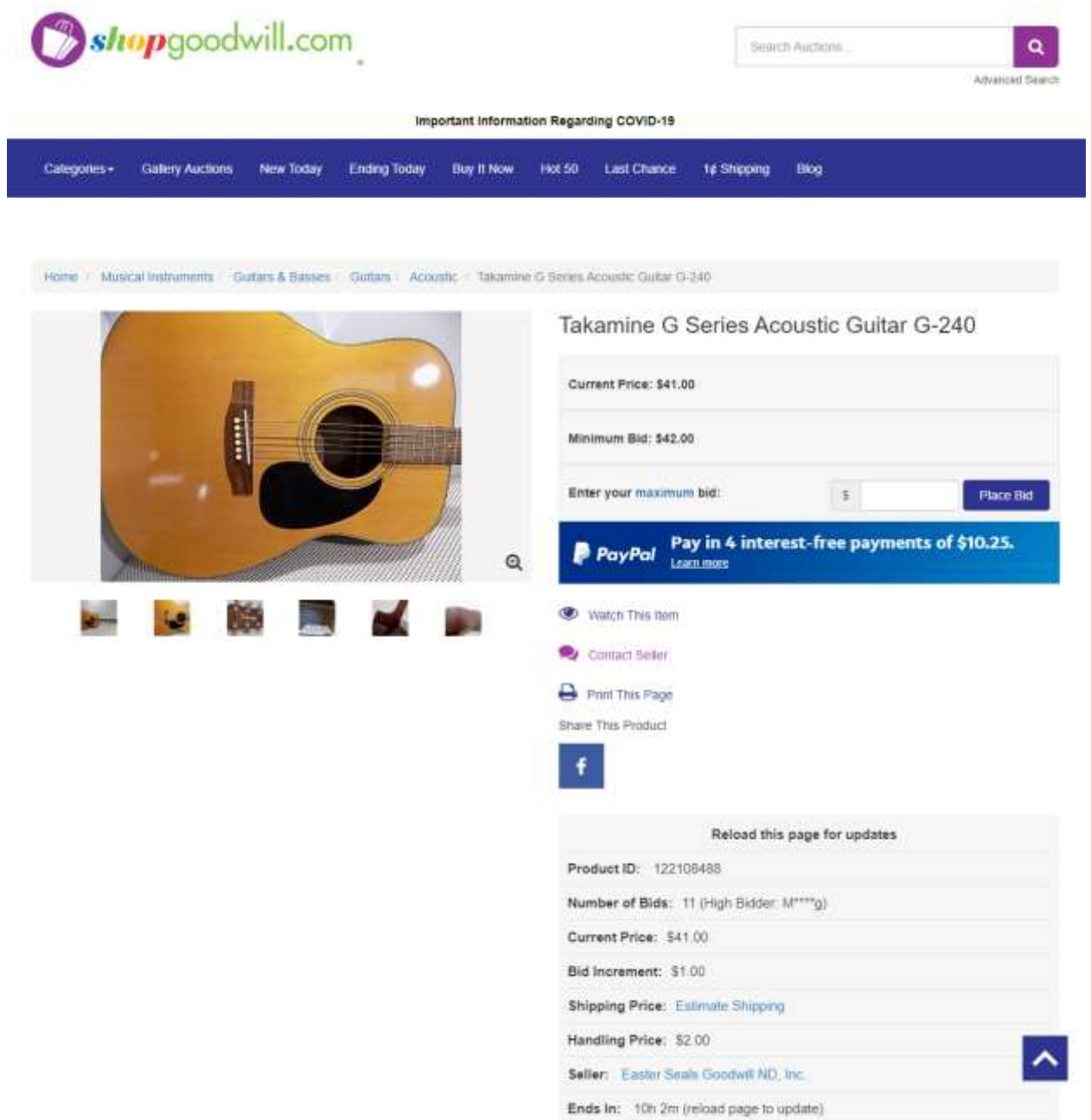
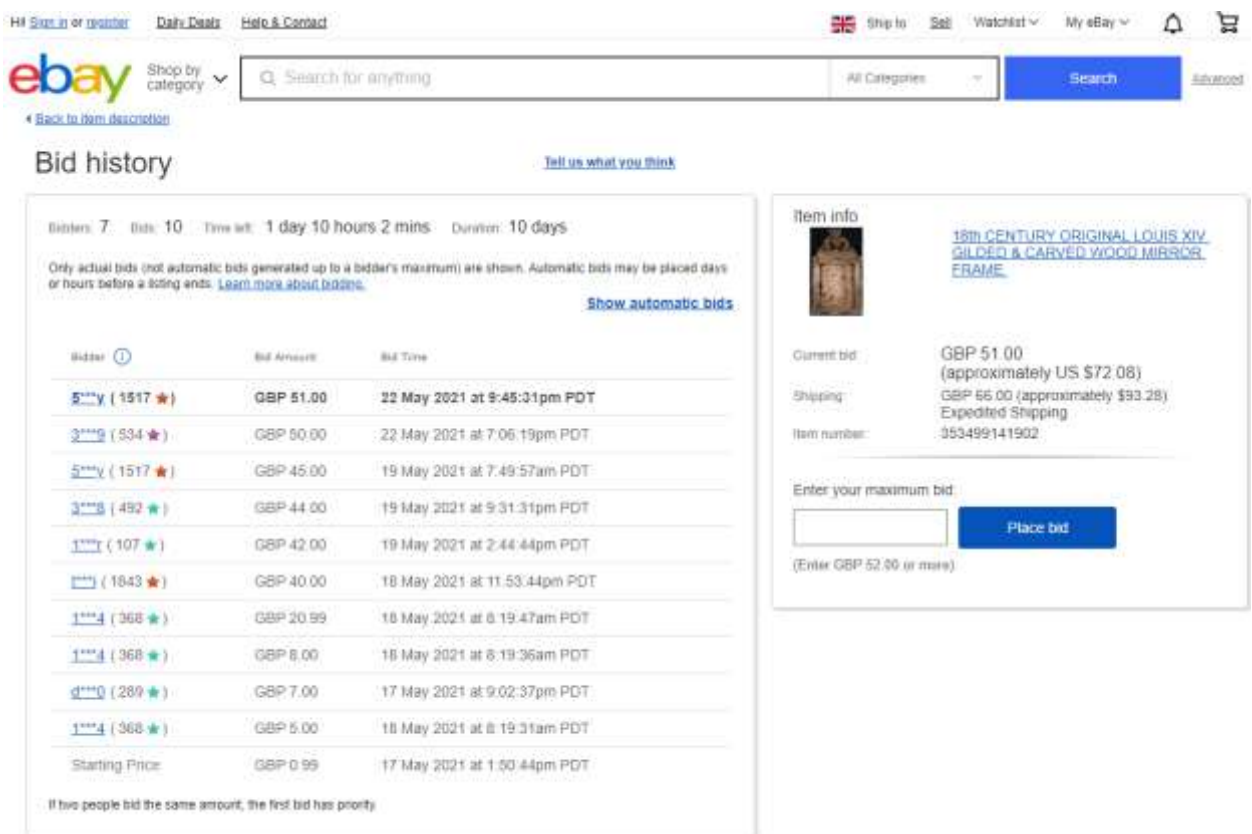


Рисунок 1.1 – Електронний аукціон Shopgoodwill

eBay [6] є найбільшим аукціонним сайтом в Інтернеті і позиціонує себе як найбільший торговий центр з понад 100 мільйонами зареєстрованих користувачів, основою організації його є те, що він пропонує безпечну платформу як для покупців, так і для продавців, при цьому велика кількість функцій доступні як для покупців, так і для продавців для організації їх роботи, включаючи розширені функції пошуку, опції зворотного зв'язку, недоліком системи є те, що збори сервісу при цьому вищі за аналоги, а

система дещо заплутана для початківців, товари, наявні на сайті, включають одяг і аксесуари, електроніку, предмети колекціонування і мистецтво, дім і сад, спортивні товари тощо [7].

На відміну від попереднього аукціону, eBay надає можливість переглянути історію зміни вартості, які включаючи початкову вартість, виставлену продавцем, так і всі пропозиції, які змінювали цю вартість в подальшому (рис. 1.2).



Bid history

Bidders: 7 Bids: 10 Time left: 1 day 10 hours 2 mins Duration: 10 days

Only actual bids (not automatic bids generated up to a bidder's maximum) are shown. Automatic bids may be placed days or hours before a listing ends. [Learn more about bidders.](#)

[Show automatic bids](#)

Bidder	Bid Amount	Bid Time
5***y (1517 ★)	GBP 51.00	22 May 2021 at 9:45:31pm PDT
3***9 (534 ★)	GBP 50.00	22 May 2021 at 7:06:19pm PDT
5***y (1517 ★)	GBP 45.00	19 May 2021 at 7:49:57am PDT
3***9 (492 ★)	GBP 44.00	19 May 2021 at 9:31:31pm PDT
1***t (107 ★)	GBP 42.00	19 May 2021 at 2:44:44pm PDT
1***t (1843 ★)	GBP 40.00	18 May 2021 at 11:53:44pm PDT
1***4 (368 ★)	GBP 20.99	18 May 2021 at 8:19:47am PDT
1***4 (368 ★)	GBP 8.00	18 May 2021 at 8:19:36am PDT
4***0 (289 ★)	GBP 7.00	17 May 2021 at 9:02:37pm PDT
1***4 (368 ★)	GBP 5.00	18 May 2021 at 8:19:31am PDT
Starting Price	GBP 0.99	17 May 2021 at 1:50:44pm PDT

If two people bid the same amount, the first bid has priority.

Item info

[18th CENTURY ORIGINAL LOUIS XIV GILDED & CARVED WOOD MIRROR FRAME](#)

Current bid: GBP 51.00 (approximately US \$72.08)

Shipping: GBP 65.00 (approximately \$93.28) Expedited Shipping

Item number: 353499141902

Enter your maximum bid: [Place bid](#)

(Enter GBP 52.00 or more)

Рисунок 1.2 – Електронний аукціон eBay

У результаті виконання огляду програмних аналогів було виявлено різного роду вебсайти, які дозволяють обмежувати коло учасників. Це говорить про можливість організації аукціонів для людей, які входять в певну обособлену спочатку групу. До цієї групи програмних аналогів можна віднести GSA Auctions (рис. 1.3).

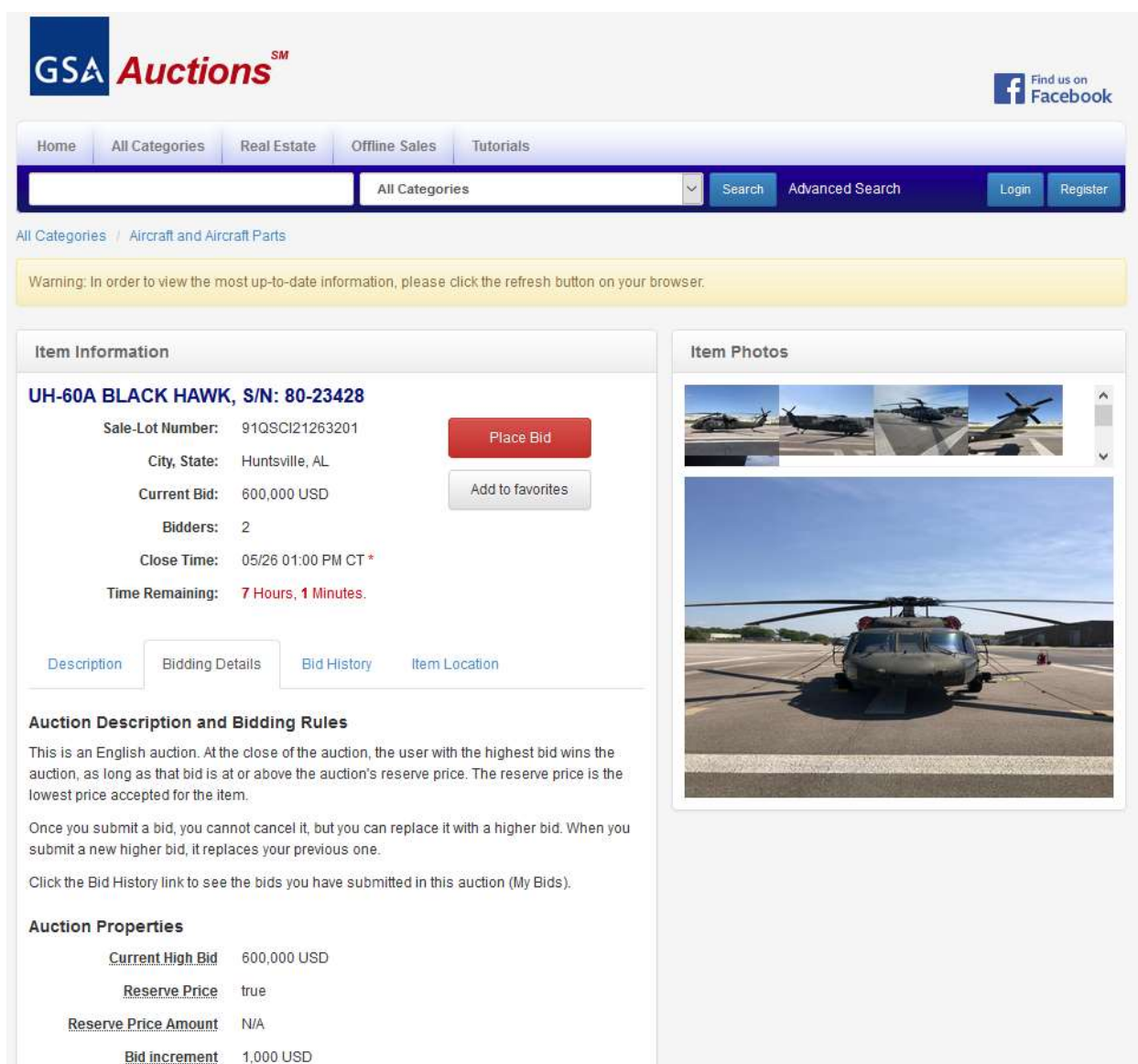


Рисунок 1.3 – Електронний аукціон GSA Auctions

GSA Auctions [8] – це урядовий вебсайт, де можна придбати федеральну власність, наприклад, човни, вертольоти, іншу техніку, медичне обладнання та предмети аерокосмічної програми уряду Сполучених Штатів Америки.

На вебсайті здійснюється продаж речей напряму, відповідно не потрібно платити націнку вебсайту, який виконує роль посередника, кожен лістинг має мінімальну резервну ціну, що означає, що навіть за пропозиції найбільшої ціни можна не виграти аукціон, при цьому безпосередньо на вебсайті така ціна не вказується, що зроблено з метою максимізації зароблених коштів, у свою чергу отримані кошти, як правило,

спрямовуються на виплати жертвам злочинів, підтримку програм запобігання злочинам та придбання нового обладнання [9].

1.3 Висновки за розділом 1

Завдання дипломної кваліфікаційної роботи полягають в наступному:

- аналіз предметної області;
- проєктування програмного забезпечення;
- реалізація програмного забезпечення.

У даному розділі виконано аналіз предметної області, для чого розглянуто принципи організації продажу речей за конкретною ціною, включаючи поняття аукціону, принципів його функціонування, сформульовано принципи організації аукціону в програмі, що розробляється, виконано огляд програмних аналогів.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Функціональні вимоги до програмного забезпечення

Програма повинна забезпечувати наступні функціональні вимоги:

- визначення інформації про річ, яка буде продаватися;
- редагування інформації про річ;
- відображення всіх речей користувача;
- створення оголошення про продаж речі;
- редагування оголошення;
- відображення всіх оголошень продавця;
- відображення всіх власних оголошень;
- відображення активних власних оголошень;
- відображення виконаних власних оголошень;
- створення пропозиції про купівлю речі;
- перегляд пропозицій про купівлю речі;
- внесення грошей на рахунок;
- відображення повних даних оголошення;
- відображення речей з найбільшою різницею між конкурентною і базовою вартостями;
- деактивація оголошення;
- пошук речей;
- відображення оголошень за категоріями;
- відображення власних пропозицій про купівлю речей;
- скасування власної пропозиції про купівлю речі;
- відображення власних придбаних речей;
- створення відгуків про продавця;
- перегляд відгуків на продавця;
- перегляд власних відгуків;
- авторизація облікового запису;
- реєстрація облікового запису.

Усі користувачі можуть бути розділені на 2 ролі: продавці та покупці.

Сценарії використання покупцем програми представлені на рис. 2.1. Сценарії реєстрації облікового запису та скасування власної пропозиції про купівлю речі є альтернативними.



Рисунок 2.1 – Діаграма прецедентів покупця

Сценарії використання продавцем програми представлені на рис. 2.2.

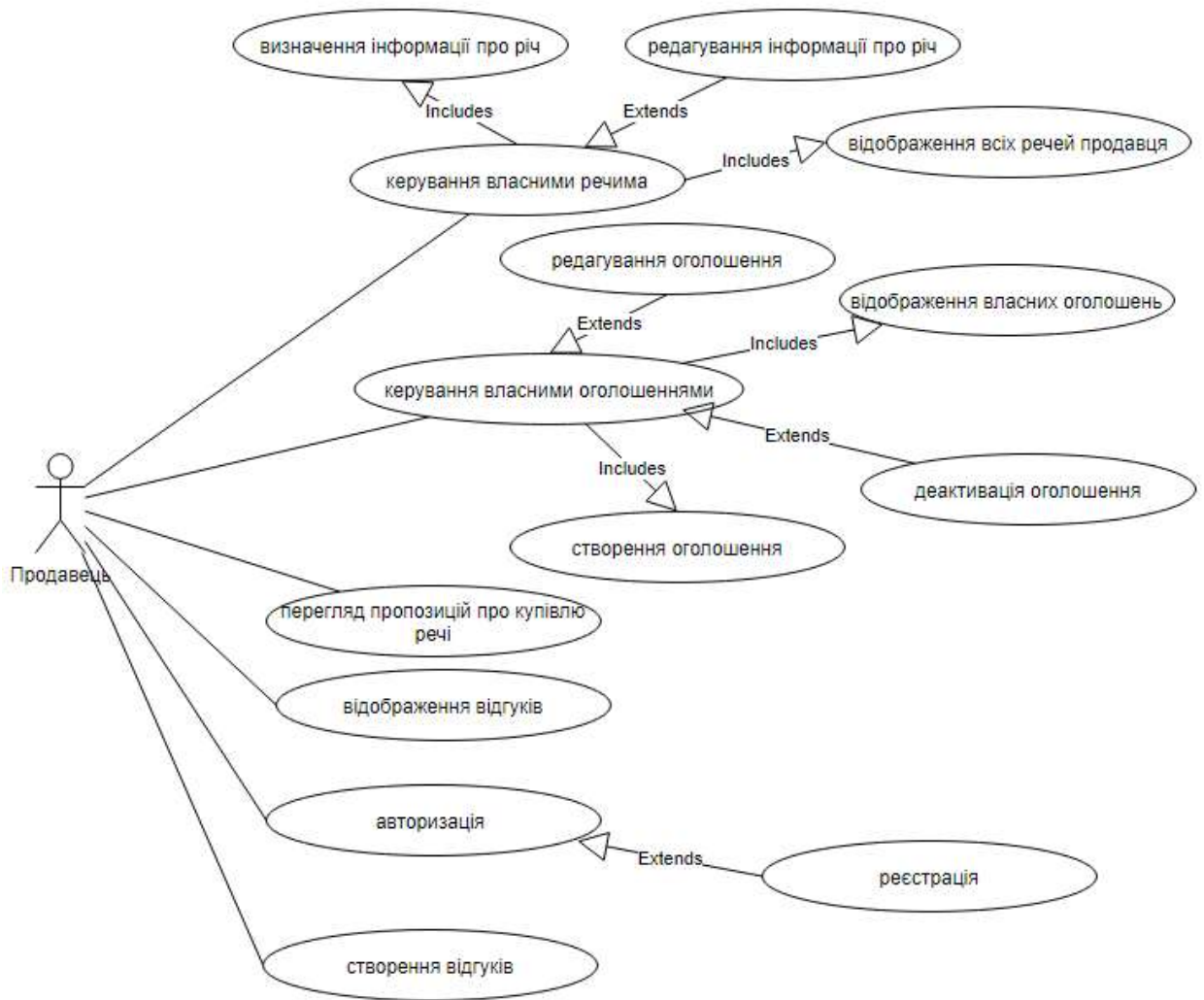


Рисунок 2.2 – Діаграма прецедентів продавця

Сценарії редагування інформації про річ, редагування оголошення, деактивації оголошення, реєстрації є альтернативними для продавця.

Неавторизований користувач (рис. 2.3) може взаємодіяти з програмою, але при цьому має обмежений набір сценаріїв. Цей набір сценаріїв охоплює частину сценаріїв покупця. Це сценарії відображення даних стосовно оголошень, відгуків, а також сценарії пошуку речей, авторизації та альтернативний сценарій реєстрації користувача.

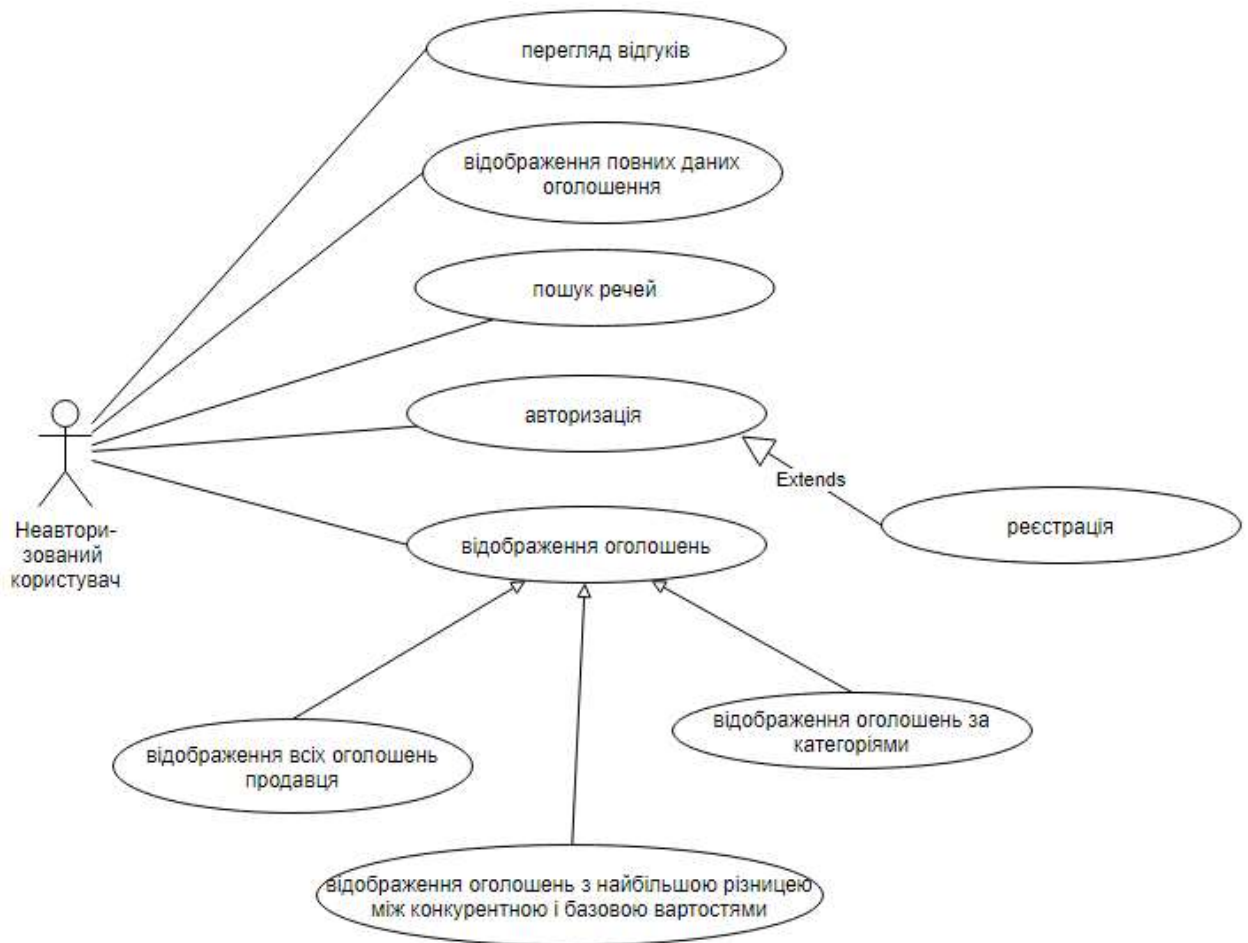


Рисунок 2.3 – Діаграма прецедентів неавторизованого користувача

2.2 Вибір засобів розробки

У даному підрозділі потрібно виконати вибір мови програмування, що буде використано для розробки програмного забезпечення, та вибір засобів роботи з базою даних (БД), оскільки саме роботи з даними лежить в основі роботи програми.

Розроблюване вебрішення можна створити на основі мов програмування C# та Java. Результати порівняння зведено до табл. 2.1.

Мову програмування C# обрано для реалізації програмного забезпечення, оскільки разом з технологією ASP.NET [10]-[11] це дозволить створити вебзастосунок, який буде мати більшу швидкість виконання, окрім того сам процес розробки буде достатньо зручним, хоча в даному компоненті мови програмування співставні.

Таблиця 2.1 – Вибір мови програмування

	C#	Java
Веброзробка	Так	Так
Вебфреймворк	ASP.NET	Spring
Автоматичне керування пам'яттю	Так	Так
Кросплатформність	Так	Так
Швидкість виконання програм	Більша	Менша

Безпосередньо веброзробка буде виконуватися на основі використання технології ASP.NET – це набір інструментів веброзробки, пропонованих корпорацією Майкрософт: такі програми, як Visual Studio .NET, дозволяють веброзробникам створювати динамічні вебсайти за допомогою візуального інтерфейсу, програмісти можуть писати свій власний код та сценарії та включати його у вебсайти ASP.NET, загалом ASP.NET окрім C# також підтримує Visual Basic.NET, JScript .NET та мови з відкритим кодом, такі як Python та Perl, ASP.NET побудований на платформі .NET, що забезпечує інтерфейс прикладних програм для програмістів, засоби розробки .NET можна використовувати для створення застосунків як для операційної системи Windows, так і для Інтернету, такі програми, як Visual Studio .NET, надають розробникам візуальний інтерфейс для створення своїх застосунків, що робить .NET зручним вибором для розробки вебінтерфейсів, щоб вебсайт ASP.NET функціонував правильно, його потрібно опублікувати на вебсервері, що підтримує програми ASP.NET, вебсервер Інтернет-служб Microsoft на сьогоднішній день є найпоширенішою платформою для вебсайтів ASP.NET, хоча для систем на базі Linux доступні деякі варіанти з відкритим кодом, ці альтернативи часто забезпечують менше засобів [12].

За результатами порівняння систем управління базами даних (СУБД) Microsoft SQL Server та MySQL було виявлено наступні відмінності:

– SQL Server спочатку була розроблена виключно для операційної системи Windows компанією Microsoft, але Microsoft нещодавно зробила СУБД доступною як на Mac OS X, так і на Linux, це дає підприємствам можливість запускати СУБД на трьох окремих платформах, з іншого боку, MySQL може безперебійно працювати в багатьох популярних операційних системах, включаючи Linux, Mac OS X та Windows;

– як MySQL, так і MSSQL підтримують декілька мов програмування, включаючи Java, C++, PHP, Ruby, Python, Delphi, Visual Basic, Go та R, MySQL підтримує додаткові мови, такі як Tcl, Scheme, Perl, Eiffel та Haskell, хоча можна використовувати обидва типи баз даних для проєктів Windows і Linux, MySQL працює з PHP, а Microsoft SQL Server в основному використовується з .NET, інтеграція простіша, якщо використовується MySQL для PHP та Microsoft SQL Server для проєктів Windows;

– Microsoft SQL Server дозволяє налаштовувати свої класи сутностей в .NET, можна розпочати роботу з запитамі LINQ, за допомогою MySQL потрібно буде завантажити сторонні інструменти, якщо необхідно використовувати .NET;

– як MySQL, так і Microsoft SQL Server мають засоби IDE, але потрібно підібрати потрібний інструмент до відповідного сервера: MySQL має Enterprise Manager, а Microsoft SQL Server використовує Management Studio, ці інструменти дозволяють підключатися до сервера та керувати конфігураціями архітектури, безпеки та дизайну таблиць;

– Microsoft SQL Server та MySQL розроблені як бінарні колекції: MySQL дозволяє розробникам використовувати бінарні файли для обробки файлів БД навіть під час запуску, до файлів БД також можна отримати доступ та керувати ними за допомогою альтернативних процесів під час виконання, навпаки, MSSQL не дозволяє жодному процесу маніпулювати бінарними файлами або файлами БД або отримувати до них доступ, це позбавляє можливості отримати доступ до даних або безпосередньо

маніпулювати ними, таким чином, Microsoft SQL Server є більш безпечною СУБД ніж MySQL;

- при використанні MySQL потрібно створити резервну копію даних, витягнувши їх за допомогою операторів SQL, СУБД надає інструмент для блокування БД під час резервного копіювання даних, ця функція мінімізує ймовірність пошкодження даних під час перемикання між версіями або версіями MySQL, недоліком є те, що це робить відновлення даних трудомістким процесом, оскільки вимагає виконання декількох операторів SQL, Microsoft SQL Server не блокує БД під час резервного копіювання даних, дозволяючи користувачам робити копії та відновлювати масові обсяги даних з мінімальними зусиллями;

- MySQL не дозволяє користувачам скасовувати запит після його запуску, так щоб зупинити виконання запиту SQL, користувачі повинні зупинити повний процес, користувачі Microsoft SQL Server можуть скасувати запит до БД під час його виконання, не вбиваючи повний процес, більше того, Microsoft SQL Server використовує механізм транзакцій для підтримки стабільного стану, це дає Microsoft SQL Server достатньо помітну перевагу над MySQL [13].

За сукупністю перелічених відмінностей, враховуючи, що за попереднім вибором проєкт буде виконуватися у вигляді вебзастосунку для платформи .NET, саме Microsoft SQL Server було обрано в якості СУБД для реалізації проєкту.

2.3 Проєктування бази даних

Сутність Thing представляє річ, яка виставляється на продаж.

Сутність Ad представляє оголошення про продаж речі. Оголошення відділено від речі для того, щоб не потрібно було з кожним оголошенням вводити заново інформацію про річ, а можна було скористатися описами, які було виконано раніше. Тобто якщо користувач буде продавати постійно або

через деякий час ту саму річ, то вводити дані повторно йому не доведеться, а потрібно буде тільки визначити параметри продажу цієї речі цього разу.

Сутність Proposition представляє пропозицію користувача про купівлю речі.

Сутність Image представляє зображення речі. Виділення окремої сутності знадобилося для того, щоб для кожної речі можна було встановити різні зображення.

Сутність User представляє дані користувача, який може виступати і як продавець, і як покупець.

Сутність Review представляє відгук про користувача.

Сутність Payment представляє платіж, виконаний для придбання речі.

Сутність Category представляє категорію речей, що необхідно для їх

Таблиця Thing має наступні поля:

- id – номер речі;
- name – назва речі;
- information – повна інформація про річ;
- userId – користувач, який є власником речі (представляється номером, який є зовнішнім ключем з таблиці User).

Таблиця Ad має наступні поля:

- id – номер оголошення;
- thingId – річ, щодо продажу якої створено оголошення;
- created – дата створення оголошення;
- deadline – остаточна дата, коли продаж буде завершено;
- price – мінімальна вартість, за яку згоден продати речі продавець (покупець не може встановити вартість, меншу за задану);
- count – кількість речей, яку здатен продати продавець;
- status – стан оголошення;
- city – місто, де знаходяться речі;
- delivery – інформація про доставку речі;

– limit – мінімальна оцінка користувача, за якої покупцю може бути продано річ.

Таблиця Proposition має наступні поля:

- id – номер пропозиції;
- userId – користувач, який висунув дану пропозицію про купівлю речі (представляється номером, який є зовнішнім ключем з таблиці User);
- adId – оголошення, на яке оформлено дану пропозицію про купівлю речі (представляється номером, який є зовнішнім ключем з таблиці Ad);
- price – вартість, за яку згоден придбати річ покупець;
- count – кількість речей, яку згоден придбати покупець;
- created – дата і час створення пропозиції;
- status – стан пропозиції.

Таблиця Image має наступні поля:

- id – номер зображення;
- name – назва зображення (або короткий опис, використовується як підказка);
- source – розташування файлу зображення;
- thingId – річ, яку представляє зображення (представляється номером, який є зовнішнім ключем з таблиці Thing).

Таблиця User має наступні поля:

- id – номер облікового запису користувача;
- login – логін користувача;
- password – пароль користувача;
- name – ім'я користувача;
- account – рахунок користувача;
- balsum – сума балансу користувача;
- score – оцінка користувача.

Таблиця Review має наступні поля:

- id – номер відгуку;

- `userId` – користувач, про якого сформульовано відгук (представляється номером, який є зовнішнім ключем з таблиці `User`);
- `fromuserId` – користувач, який написав відгук (представляється номером, який є зовнішнім ключем з таблиці `User`);
- `created` – дата створення відгуку;
- `information` – текст відгуку;
- `score` – оцінка відгуку.

Таблиця `Category` має наступні поля:

- `id` – номер категорії;
- `name` – назва категорії.

Кожна річ характеризується різними категоріями, тобто є можливість описати одну річ не однією, а різними категоріями. Звичайно, що кожна категорія має багато речей. Тобто таблиці `Thing` і `Category` зв'язані зв'язком багато-до-багатьох. Для реалізації даного зв'язку було створено таблицю `ThingCategory`.

Таблиця `ThingCategory` має наступні поля:

- `id` – номер;
- `thingId` – річ, яка характеризується категорією (представляється номером, який є зовнішнім ключем з таблиці `Thing`);
- `categoryId` – категорія, до якої належить дана річ (представляється номером, який є зовнішнім ключем з таблиці `Category`).

Таблиця `Payment` має наступні поля:

- `id` – номер платежу;
- `propositionId` – пропозиція про придбання речі, на основі якої виконано платіж (представляється номером, який є зовнішнім ключем з таблиці `Proposition`);
- `pay` – сума платежу;
- `created` – дата виконання платежу.

Схема БД представлена на рис. 2.4.

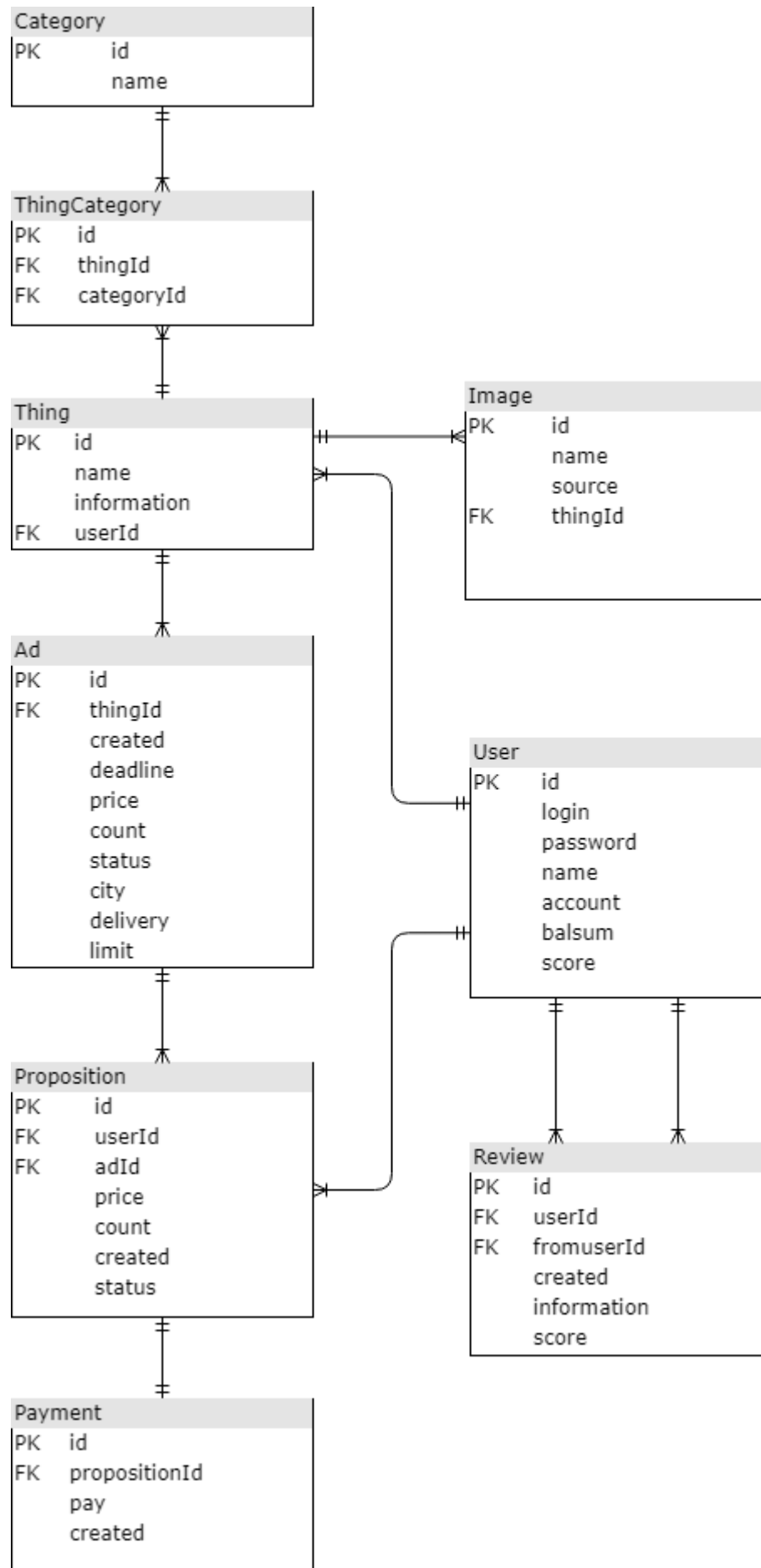


Рисунок 2.4 – Схема БД

Зв'язок між таблицями User і Thing один-до-багатьох, оскільки один користувач (продавець) може мати багато речей на продаж, але кожна річ може належати тільки одному продавцю.

Зв'язок між таблицями Thing і Ad один-до-багатьох, оскільки одна річ може мати багато пов'язаних з нею оголошень про продаж, але одне оголошення може мати відношення тільки до однієї речі.

Зв'язок між таблицями User і Proposition один-до-багатьох, оскільки один користувач може встановлювати багато пропозицій про придбання речей, але кожна пропозиція може бути встановлена тільки одним покупцем.

Зв'язок між таблицями Ad і Proposition один-до-багатьох, оскільки до одного оголошення може бути висунуто багато пропозицій про купівлю, але кожна пропозиція стосується тільки одного оголошення.

Зв'язок між таблицями Thing і Image один-до-багатьох, оскільки одна річ може мати багато зображень, при цьому кожне зображення може відноситися тільки до однієї речі.

Зв'язок між таблицями User і Review один-до-багатьох, один користувач може написати багато відгуків, але кожен відгук може бути створений тільки одним користувачем.

Зв'язок між таблицями Proposition і Payment один-до-одного, оскільки одна пропозиція може мати тільки одну фактичну сплату в результаті, але і кожна сплата може бути виконана тільки за однією раніше створеною пропозицією.

Зв'язок між таблицями Thing і ThingCategory один-до-багатьох, оскільки кожній речі може відповідати різна кількість записів про її категорії, але один запис може характеризуватися тільки одну річ.

Зв'язок між таблицями Category і ThingCategory один-до-багатьох, оскільки кожна категорія може мати багато записів про категорії різних речей, але кожен запис може містити тільки одну категорію.

2.4 Висновки за розділом 2

За результатами виконання другого розділу:

- визначено функціональні вимоги до програмного забезпечення;
- обрано мову програмування C#, технологію ASP.NET і СУБД Microsoft SQL Server для розроблення програмного забезпечення;
- визначено структуру БД для організації програми.

З РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програмного забезпечення

Функціонування програмного забезпечення побудовано на основі взаємодії з користувачем, яка визначає порядок виконання програми.

Спочатку користувачу демонструється сторінка пошуку речей, звідки він може перейти на сторінки деталізації оголошень. Окрім того він може за допомогою меню обрати один з режимів роботи з програмою. Режими роботи програми включають:

- відображення історії всіх придбань, що дозволяє переглянути придбання речей з найбільшою різницею між початковою та остаточною цінами;
- відображення оголошень за обраною категорією;
- відображення відгуків про покупців.

Окрім того користувачу доступна можливість повторно виконати пошук, а також авторизуватися, після чого надаються додатково пункти меню:

- відображення власних оголошень, що дозволяє в різних варіантах відобразити перелік оголошень, які розмістив або розміщав раніше авторизований користувач, відображення також додає можливість керування власними оголошеннями;
- відображення власних речей, що дозволяє переглянути речі, на основі яких авторизованим користувачем було створено оголошення, відображення також вміщує процес керування речами;
- відображення власних пропозицій про придбання речей;
- відображення власних придбаних речей;
- внесення грошей на рахунок.

Результуюча схема функціонування програми приведена на рис. 3.1.

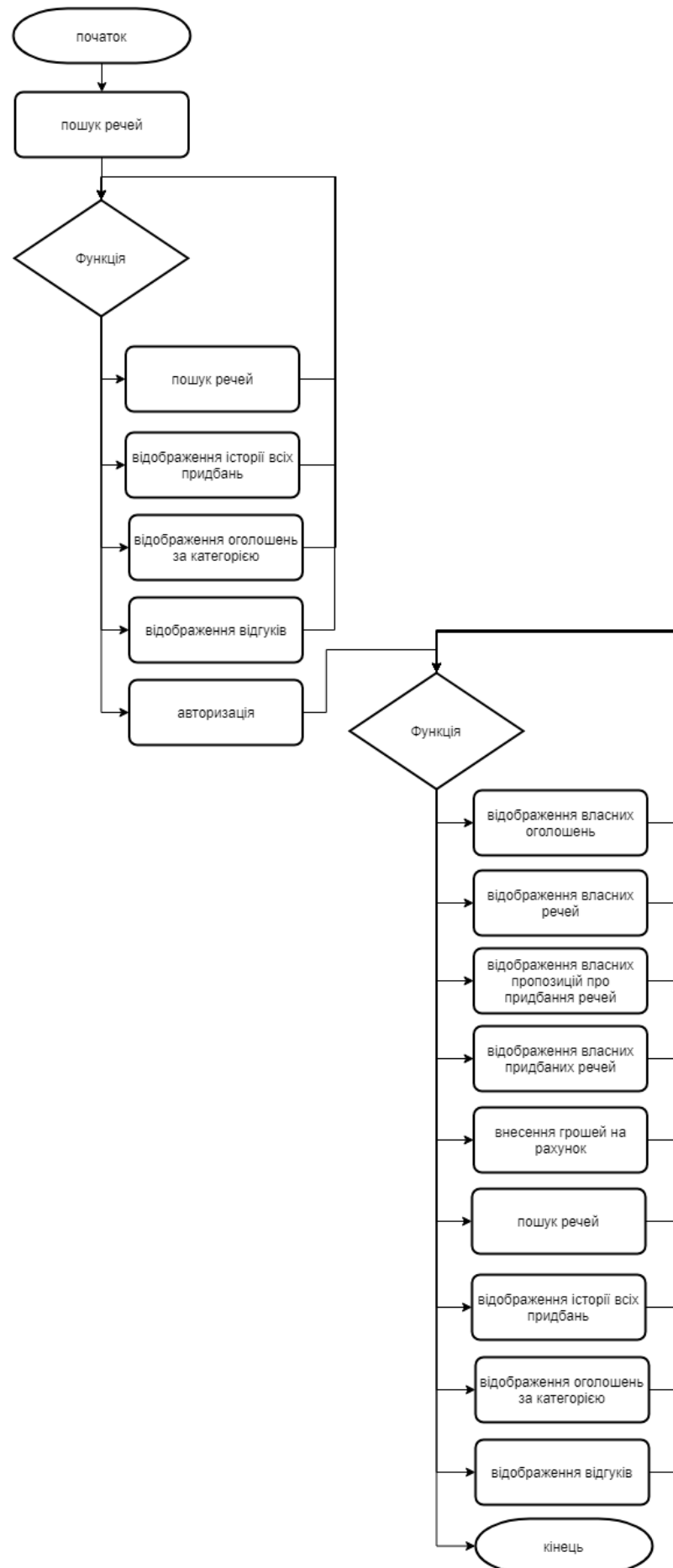


Рисунок 3.1 – Схема функціонування програми

Окремо необхідно розглянути об'єкт оголошення, оскільки він є одним з основних у предметній області. Саме оголошення є основою взаємодії між продавцем і покупцем. Речі як такі не доступні покупцю, а є тільки частиною оголошення. У той же час на пропозиції покупця продавець не може впливати, вони відфільтровуються автоматично.

На рис. 3.2 представлено діаграму станів оголошення.

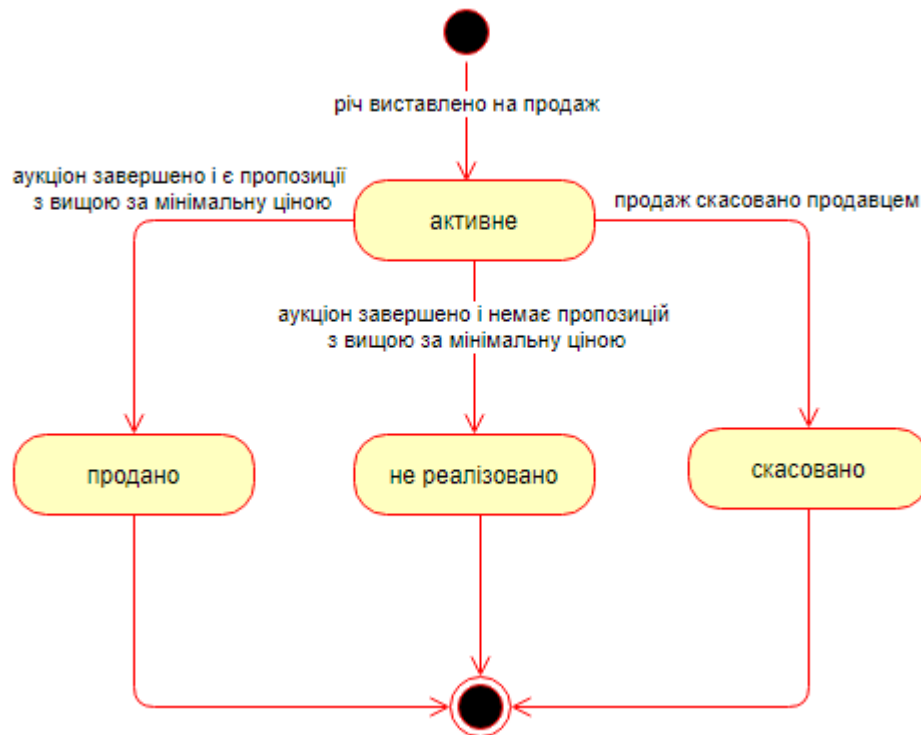


Рисунок 3.2 – Діаграма станів оголошення

Після того як речі виставлено на продаж, тобто створено оголошення, оголошення перебуває в активному стані. Кожен покупець може подавати пропозиції на придбання речі або речей з оголошення.

Далі оголошення може бути в будь-який момент скасовано за наявності потреби у продавця.

Якщо аукціон завершено, то відбувається перевірка його результатів. Якщо було отримано хоча би одну пропозицію про купівлю речі з ціною вищою за мінімальний встановлений продавцем, то оголошення переміщується в стан «продано».

Якщо аукціон завершено, але жодної підходящої пропозиції отримано не було, то оголошення переміщується до стану «не реалізовано».

3.2 Структура реалізованого програмного забезпечення

До структури реалізованого програмного забезпечення входять:

- моделі – класи, які містять основні сутності предметної області і знаходяться в папці Models;
- представлення – динамічні вебсторінки, які визначають спосіб представлення даних користувачу і знаходяться в папці Views;
- контролери – класи, які визначають дані, необхідні для роботи відповідних представлень.

Серед представлень було розроблено наступні вебсторінки в папці thing для відповідного контролера:

- index.cshtml – вебсторінка роботи з власними речами;
- details.cshtml – вебсторінка повної інформації про річ;
- create.cshtml – вебсторінка створення речі;
- edit.cshtml – вебсторінка редагування речі.

Серед представлень було розроблено наступні вебсторінки в папці ad для відповідного контролера:

- index.cshtml – вебсторінка роботи з власними оголошеннями;
- details.cshtml – вебсторінка повної інформації про оголошення;
- create.cshtml – вебсторінка створення оголошення;
- edit.cshtml – вебсторінка редагування оголошення;
- searchads.cshtml – вебсторінка пошуку оголошень;
- categoryads.cshtml – вебсторінка перегляду оголошень за категорією;
- doneads.cshtml – вебсторінка перегляду виконаних оголошень;
- activeads.cshtml – вебсторінка перегляду активних оголошень;
- userads.cshtml – вебсторінка перегляду оголошень, що належать заданому користувачу.

Схематичне представлення даної структури представлено на рис. 3.3.

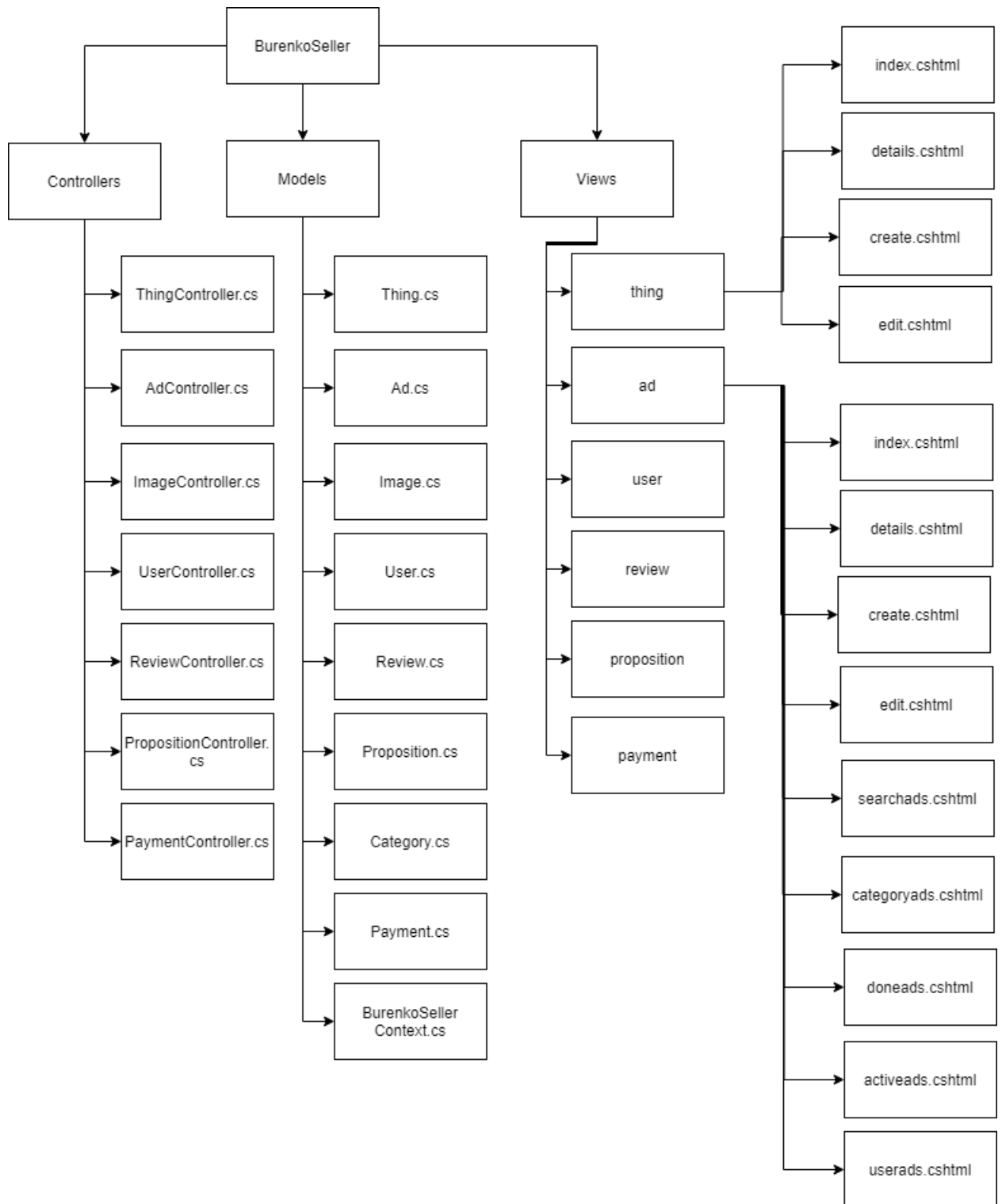


Рисунок 3.3 – Структурна схема програмного забезпечення

Серед представлень було розроблено наступні вебсторінки в папці *proposition* для відповідного контролера:

- index.cshtml – вебсторінка роботи з власними пропозиціями;
- done.cshtml – вебсторінка перегляду власних покупок;
- history.cshtml – вебсторінка перегляду покупок з найбільшою різницею між початковою вартістю речі та ціною придбання;
- details.cshtml – вебсторінка повної інформації про пропозицію;
- create.cshtml – вебсторінка створення пропозиції;
- edit.cshtml – вебсторінка редагування пропозиції;
- cancel.cshtml – вебсторінка скасування пропозиції.

Серед представлень було розроблено наступні вебсторінки в папці review для відповідного контролера:

- index.cshtml – вебсторінка роботи з власними відгуками;
- create.cshtml – вебсторінка створення відгуку;
- user.cshtml – вебсторінка перегляду відгуків за користувачем.

Серед представлень було розроблено наступні вебсторінки в папці user для відповідного контролера:

- create.cshtml – вебсторінка реєстрації користувача;
- auth.cshtml – вебсторінка авторизації користувача;
- index.cshtml – вебсторінка профілю користувача.

Реалізацію представлень було виконано за допомогою шаблону [14].

Також було створено сторінку payment.cshtml для підтвердження внесення суми коштів на рахунок.

3.3 Опис прийнятих під час реалізації рішень

У відповідність з кожною таблицею БД було реалізовано класи моделей. Вони повторюють всі таблиці, окрім таблиці ThingCategory, необхідної для реалізації зв'язка багато-до-багатьох. Даний зв'язок в коді створений наступним чином. Для цього в класі Thing було створено віртуальне поле колекції категорій, а в конструкторі класу реалізовано створення відповідної колекції:

```

public virtual ICollection<Category> Categories{ get; set; }
public Thing()
{
    Categories = new List<Category>();
    Ads = new List<Ad>();
    Images = new List<Image>();
}

```

У класі Category реалізовано визначення колекції речей, що відповідають категорії:

```

public virtual ICollection<Thing> Things{ get; set; }
public Category()
{
    Things = new List<Thing>();
}

```

Далі в класі контексту даних реалізовано метод OnModelCreating(DbModelBuilder mb), який відповідає за створення зв'язка між таблицями і визначення імені проміжної таблиці:

```

mb.Entity<Thing>().HasMany(c => c.Category)
    .WithMany(s => s.Things)
    .Map(t => t.MapLeftKey("ThingId"))
    .MapRightKey("CategoryId")
    .ToTable("ThingCategory"));

```

Реалізацію зв'язка один-до-багатьох виконано наступним чином. Наприклад, в класі Proposition міститься поле, яке вказує на номер

оголошення, на яке створено пропозицію, та віртуальне поле, яке містить повний об'єкт оголошення:

```
public int? AdId{ get; set; }
public User Ad{ get; set; }
```

Відповідно з іншого боку, в класі Ad, забезпечено визначення колекції пропозицій в якості відповідного поля та створено відповідний об'єкт в конструкторі класу:

```
public virtual ICollection<Proposition> Propositions{ get; set; }
public Ad()
{
    Propositions = new List<Proposition>();
}
```

Клас BurenkoSellerContext реалізує контекст даних і містить відповідні поля:

- DbSet<Category> Categories – множина категорій речей;
- DbSet<Thing> Things – множина речей;
- DbSet<Image> Images – множина зображень речі;
- DbSet<Ad> Ads – множина оголошень речей;
- DbSet<User> Users – множина користувачів;
- DbSet<Proposition> Propositions – множина пропозицій;
- DbSet<Review> Reviews – множина відгуків;
- DbSet<Payment> Payments – множина платежів.

Клас ThingController реалізує контролери речей. Для цього усередині класу створено наступні методи:

- ThingController (BurenkoSellerContext cont) – конструктор, який визначає контекст даних;

- `async Task<IActionResult> Index()` – метод контролера, що підтримує звернення до базової сторінки роботи з речами;
- `async Task<IActionResult> Details(int? id)` – метод контролера для виведення повної інформації про річ з ідентифікаційним номером `id`;
- `IActionResult Create()` – метод контролера відображення початкового стану сторінки створення речі;
- `async Task<IActionResult> Create([Bind("Name, Information")] Thing thing)` – метод контролера збереження нової речі `thing` з назвою `Name` та описом `Information`;
- `async Task<IActionResult> Edit(int? id)` – метод контролера відображення початкового стану сторінки редагування речі;
- `async Task<IActionResult> Edit(int id, [Bind("Id, Name, Information")] Thing thing)` – метод контролера редагування речі `thing` з ідентифікаційним номером `id`, змінюючи її назву на `Name`, опис на `Information`;
- `bool ThingExists(int id)` – метод перевірки існування речі з номером `id`.

Клас `AdController` реалізує контролери оголошень про продаж речей. Для цього усередині класу створено наступні методи:

- `AdController (BurenkoSellerContext cont)` – конструктор, який визначає контекст даних;
- `async Task<IActionResult> Details(int? id)` – метод контролера для виведення повної інформації про оголошення про продаж речі з ідентифікаційним номером `id`;
- `IActionResult Create()` – метод контролера відображення початкового стану сторінки створення оголошення;
- `async Task<IActionResult> Create([Bind("Id, Deadline, Price, Count, City, Delivery, Limit, ThingId")] Ad ad)` – метод контролера збереження нового оголошення `ad` з датою завершення торгів `Deadline`, вартістю `Price`, кількістю речей на продаж `Count`, містом розміщення речей `City`, доставкою `Delivery`, мінімальною оцінкою покупця `Limit`, номером речі `ThingId`, продаж якої

відбувається, при цьому автоматично встановлюється дата створення оголошення та стан оголошення;

- `async Task<IActionResult> Edit(int? id)` – метод контролера відображення початкового стану сторінки редагування оголошення;

- `async Task<IActionResult> Edit(int id, [Bind("Id, Deadline, Price, Count, Status, City, Delivery, Limit, ThingId")] Ad ad)` – метод контролера редагування оголошення `ad` з ідентифікаційним номером `id`, змінюючи дату завершення торгів на `Deadline`, вартість на `Price`, кількість речей на продаж на `Count`, місто розміщення речей на `City`, доставку на `Delivery`, мінімальну оцінку покупця на `Limit`, номер речі на `ThingId`;

- `async Task<IActionResult> Deacticate(int id)` – метод, який реалізує деактивацію оголошення;

- `async Task<IActionResult> UserAds(int? id)` – метод, який обробляє виведення оголошень, створених користувачем з номером облікового запису `id`;

- `async Task<IActionResult> ActiveAds()` – метод контролера сторінки виведення активних оголошень;

- `async Task<IActionResult> DoneAds()` – метод контролера сторінки виведення оголошень, за якими було виконано продаж речей;

- `async Task<IActionResult> CategoryAds(int id)` – метод контролера сторінки виведення оголошень про продаж речей, які відповідають категорії з номером `id`;

- `async Task<IActionResult> SearchAds(string name, float price1, float price2)` – метод контролера пошуку оголошень, назва речей яких містить текст `name`, а вартість знаходиться в проміжку від `price1` до `price2`;

- `bool AdExists (int id)` – метод перевірки існування оголошення з номером `id`.

Метод `async Task<IActionResult> Index()` відповідає методу контролера, що підтримує звернення до базової сторінки роботи з оголошеннями про продаж речей, на якій виводяться всі оголошення,

створені авторизованим у даний момент користувачем. Для цього відбувається витягання даних про користувача, далі витягаються всі речі, які були внесені користувачем, заповнюються їх дані, що відповідають зв'язкам з іншими таблицями, а далі формується перелік оголошень на основі встановлення відповідності тим речам, які були внесені авторизованим користувачем:

```

IQueryable<User> users = bsc.Users;
var user = await users.FirstOrDefault(a => a.Login ==
User.Identity.Name);
var things = bsc.Things.Include(a => a.Ads).Include(a =>
a.Categories).Include(a => a.Images).Where(i => i.UserId == user.Id);
List<Ad> ads = new List<Ad>();
foreach (var thing in things)
{
    ads.Add(bsc.Ads.Include(a => a.Thing).Include(a =>
a.Propositions).Where(i => i.ThingId == thing.Id));
}
ViewData["Ads"] = ads;
ViewData["Things"] = things;
return View();

```

Клас `PropositionController` реалізує контролери пропозицій про купівлю речей. Для цього усередині класу створено наступні методи:

- `PropositionController (BurenkoSellerContext cont)` – конструктор, який визначає контекст даних;
- `async Task<IActionResult> Index()` – метод контролера, що підтримує виведення всіх пропозицій на придбання речей, які були створені авторизованим у даний момент користувачем;

- `async Task<IActionResult> Details(int? id)` – метод контролера для виведення повної інформації про пропозицію з номером `id`;
- `IActionResult Create()` – метод контролера відображення початкового стану сторінки створення пропозиції;
- `async Task<IActionResult> Create([Bind("AdId, Price, Count")] Proposition proposition)` – метод контролера внесення нової пропозиції `proposition` щодо купівлі речі за оголошенням з номером `AdId` у кількості `Count` з вартістю `Price`;
- `async Task<IActionResult> Edit(int? id)` – метод контролера відображення початкового стану сторінки редагування пропозиції;
- `async Task<IActionResult> Edit(int id, [Bind("Price, Count")] Proposition proposition)` – метод контролера редагування пропозиції `proposition` з ідентифікаційним номером `id`, змінюючи її вартість на `Price`, а кількість на `Count`;
- `async Task<IActionResult> Deacticate(int id)` – метод, який реалізує скасування пропозиції;
- `async Task<IActionResult> AdPropositions(int id)` – метод, який реалізує підготовку даних для відображення всіх пропозицій про купівлю за оголошенням з номером `id`;
- `async Task<IActionResult> AdBestPropositions(int id)` – метод, який реалізує підготовку даних для відображення пропозицій про купівлю за оголошенням з номером `id`, що перемогли;
- `async Task<IActionResult> MyBestPropositions()` – метод, який реалізує підготовку даних для відображення пропозицій авторизованого користувача, які перемогли в конкурсах;
- `async Task<IActionResult> BestPropositions()` – метод, який реалізує підготовку даних для відображення купівель товарів, фактична ціна переможної пропозиції щодо яких найбільше переважала у відсотковому відношенні початку вартість, встановлену продавцем;

– bool PropositionExists(int id) – метод перевірки існування пропозиції з номером id.

3.4 Висновки за розділом 3

За результатами виконання розділу реалізовано програмне забезпечення. Описано алгоритм функціонування програми, логіку зміни станів оголошення про продаж речі. Описано структуру програмного забезпечення, реалізовані класи програмного застосунку.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Розробка призначена для продажу речей за ціною, яка буде встановлена шляхом урівноваження попиту різних покупців. Окрім того програма дозволяє виконувати індивідуальним продавцям пошук покупців на ті речі, які вони бажають продати. У свою чергу покупці можуть виконувати пошук потрібних їм речей. Серед таких речей можуть бути індивідуальні вироби.

4.2 Умови виконання програми

Технічні вимоги до сервера включають:

- жорсткий диск з вільним простором – 1 Гб;
- оперативна пам'ять – 4 Гб;
- процесор: 4 ядра, 2,8 ГГц частота.

Клієнт представлений браузером користувача. Для виконання вимог до нього потрібний встановлений сучасний браузер з доступом до мережі інтернет.

4.3 Виконання програми

Після виконання першого звернення до вебсервера користувач отримує вебсторінку пошуку (рис. 4.1). На цій сторінці користувач може виконати пошук, задаючи назву речі (або частину цієї назви) та діапазон вартостей речі. Після завдання всіх параметрів потрібно натиснути на кнопку «Знайти» для виконання пошуку та відображення в таблиці всіх знайдених речей за встановленими критеріями.

ПРОГРАМНЕ ЗАБЕПЕЧЕННЯ ПРОДАЖУ РЕЧЕЙ ЗА КОНКУРЕНТНОЮ ЦІНОЮ

МОЯ РЕЧЬ - ОГОЛОШЕННЯ - АКТИВНОСТІ - **Вийти**

Пошук речей

модель автомобіля

МІНІМАЛЬНА ВАРТІСТЬ

МАКСИМАЛЬНА ВАРТІСТЬ **Знайти**

Річ	Початкова вартість	Поточна вартість	Кінцева дата		
 Іграшкова модель автомобіля Fiat	2350 грн.	2450 грн.	21.05.2021	Перейти	Пропозиції
 Іграшкова модель автомобіля volkswagen	100 грн.	150 грн.	25.05.2021	Перейти	Пропозиції

Рисунок 4.1 – Пошук речей

Повний набір пунктів меню включає наступні розділи:

- «Мої речі» (відсутній у неавторизованого користувача), що включає підпункти меню: «Мої речі», що дозволяє керувати власними речами, «Мої оголошення», що дозволяє керувати оголошеннями про продаж речей, «Відгуки», що дозволяє переглянути користувачу відгуки про себе;
- «Оголошення», що включає підпункти меню: «Пошук», що дозволяє виконати пошук серед всіх активних на даний момент оголошень, «Історія», що дозволяє переглянути пропозиції з найбільшою різницею між вартістю придбання та початковою вартістю, «Категорії», що дозволяє переглянути оголошення за обраною з переліку категорією, «Відгуки», що дозволяє переглянути створені користувачем оголошення (відсутній у неавторизованого користувача);
- «Активності» (відсутній у неавторизованого користувача), що включає підпункти меню: «Пропозиції», що дозволяє переглянути всі власні

пропозиції про придбання речей, «Придбання», що дозволяє переглянути фактично придбані речі, та «Рахунок», що дозволяє переглянути стан рахунку та поповнити його за необхідності;

– «Вийти», що дозволяє залишити авторизований обліковий запис, або «Увійти», що дозволяє перейти до авторизації облікового запису.

Під час виконання пошуку можна натиснути на кнопку «Перейти» і перейти до сторінки з повною інформацією про річ, що продається. На сторінках, де така кнопка відсутня, можна натиснути на зображення речі і відкриється дана сторінка (рис. 4.2).

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОДАЖУ РЕЧЕЙ ЗА КОНКУРЕНТНОЮ ЦІНОЮ
МІЗ РЕЧЕЙ -
ОГОЛОШЕННЯ -
АКТИВНОСТІ -
Вийти

Іграшкова модель автомобіля Fiat



Модель у природному середовищі

Іграшкова модель автомобіля італійського виробника Fiat виконана з повною деталізацією. Автомобіль має шкіряну обшивку всередині, має повну деталізацію моторного відсіку. Працює підсвічування. Для підсвічування використовуються 2 батарейки АЗ. Модель реального автомобіля, який випускався виробником у 1960-х роках. Розмір: 10 x 5 x 7 см. Стан: гарний.

Встановити пропозицію:

2474 грн

Підтвердити

Продавець: **carseller**

Створено: 12.05.2021
Дата завершення: 21.05.2021

Початкова вартість: 2350 грн.
Поточна вартість: 2450 грн.

Пропозиції

Кількість: 1

Місто: Київ
Доставка: Нова пошта
Категорії: іграшки, автомобілі

Рисунок 4.2 – Перегляд оголошення

На сторінці оголошення відображається назва речі, карусель з зображеннями, повний опис речі, надається можливість встановити власну пропозицію про придбання речі, при цьому вартість має бути не меншою за поточну (найбільшу пропозицію) вартість з додаванням 1 % від початкової вартості, встановленої продавцем (після натиснення на кнопку «Підтвердити» відкриється окрема сторінка, на якій потрібно підтвердити операцію, перед цим вказавши кількість речей, які потрібно придбати, після чого відбудеться перевірка достатності коштів, і операція буде виконана тільки, якщо коштів достатньо).

При відкритті сторінки власних оголошень (рис. 4.3) відображається перелік усіх наявних оголошень (можна відобразити тільки активні або виконані за допомогою відповідних кнопок, а поточна натиснута кнопка відображається жирним шрифтом).

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОДАЖУ РЕЧЕЙ ЗА КОНКУРЕНТНОЮ ЦІНОЮ

МОЯ РЕЧЬ - ОГОЛОШЕННЯ - АКТИВНОСТІ - ВІДПОВІДІ

Ваші оголошення

Всі Активні Виконані

Створити оголошення

Річ	Початкова вартість	Поточна вартість	Створено	Кінцева дата	Стан			
	300 грн.	300 грн.	10.05.2021	26.05.2021	Скасовано	Пропозиції		
	930 грн.	970 грн.	10.05.2021	18.05.2021	Продано	Пропозиції		
	2350 грн.	2450 грн.	12.05.2021	21.05.2021	Активне	Пропозиції	Скасувати	Редагувати
	400 грн.	500 грн.	15.05.2021	25.05.2021	Активне	Пропозиції	Скасувати	Редагувати
	750 грн.	900 грн.	17.05.2021	28.05.2021	Активне	Пропозиції	Скасувати	Редагувати

Рисунок 4.3 – Перегляд власних оголошень

За кожним оголошенням можна натиснути на кнопку «Пропозиції» для перегляду всіх пропозицій, якщо оголошення скасоване в поточний момент або річ продано. Якщо натиснути на кнопку «Редагувати» або «Скасувати», то можна відповідним чином вплинути на оголошення. Дані кнопки наявні тільки у активних оголошень.

При перегляді власних покупок (рис. 4.4), виконаних користувачем, відображаються всі придбані товари, для кожного з яких відображається можливість перейти до створення відгуку, перегляду ходу торгів та перегляду повного оголошення.

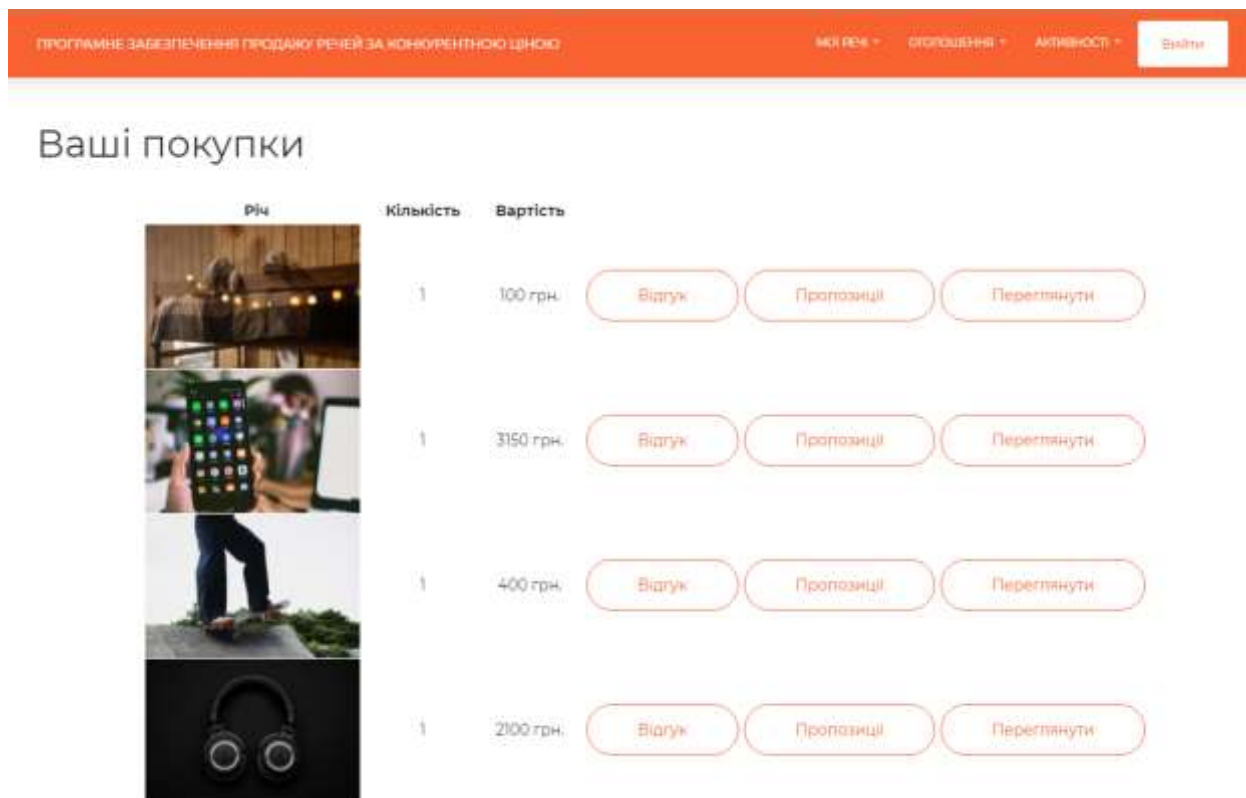


Рисунок 4.4 – Перегляд власних покупок

За кожним оголошенням, як було продемонстровано вище, можна перейти до перегляду пропозицій щодо придбання цієї речі (рис. 4.5). Дана сторінка містить логін користувача, вартість, кількість та дату виконання пропозиції. Якщо користувач не є власником оголошення, то йому ця інформація відображається тільки для ознайомлення, а за допомогою

натискання на назву речі можна перейти до оголошення і там визначити власну пропозицію.

Якщо користувач є власником оголошення, то йому надається можливість натиснути на логін користувача і перейти на його сторінку, щоб побачити його профіль (відгуки про нього і середню оцінку).

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОДАЖУ РЕЧЕЙ ЗА КОНКУРЕНТНОЮ ЦІНОЮ

МОЯ РЕЧЬ - ОГОЛОШЕННЯ - АКТИВНОСТІ - ВІДГУКИ

Пропозиції про придбання

Іграшкова модель автомобіля Fiat

Користувач	Вартість	Кількість	Дата
punto	2400 грн.	1	14.05.2021
winner	2450 грн.	1	17.05.2021

Рисунок 4.5 – Перегляд пропозицій

ВИСНОВКИ

Виконання дипломної кваліфікаційної роботи призвело до отримання наступних результатів:

- виконано аналіз проблеми продажу речей за конкурентною ціною, для чого розглянуто проблему встановлення конкурентної ціни, виконано огляд і порівняння програмних аналогів;
- виконано розробку архітектури програми, під час якої визначено функціональні вимоги до програмного забезпечення, промодельовано роботу з програмою на основі діаграм прецедентів, обрано мову програмування C#, технологію ASP.NET і СУБД Microsoft SQL Server для розроблення програмного забезпечення, визначено структуру БД для організації роботи з даними в програмі;
- реалізовано програмне забезпечення, в результаті чого описано алгоритм функціонування програми і схему функціонування програми, логіку зміни станів оголошення про продаж речі за допомогою діаграми станів, описано структуру програмного забезпечення, реалізовані класи програмного застосунку.

Створений вебзастосунок надає можливість визначення речей, які продає користувач, створення оголошень на продаж, створення пропозицій на придбання, автоматичного визначення пропозиції, яка перемогла, або набору таких пропозицій, відстеження власних пропозицій, придбань та пропозицій на власні оголошення. У результаті речі продаються продавцем покупцям за тією вартістю, яка була встановлена на основі попиту, і якщо продавець виставляє на продаж унікальні речі, то це підвищує попит на них, а значить і результуючу ціну.

Завдання дипломної кваліфікаційної роботи в результаті повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Auctions increase in popularity as bidding goes digital [Electronic resource]. – Access mode : <https://www.thisismoney.co.uk/money/smallbusiness/article-6886621/Auctions-increase-popularity-bidding-goes-digital.html>
2. Auction – Understanding How the Auction Process Works [Electronic resource]. – Access mode : <https://corporatefinanceinstitute.com/resources/knowledge/finance/auction/>
3. Shopgoodwill [Electronic resource]. – Access mode : <https://www.shopgoodwill.com/>
4. Pickard-Whitehead, G. 30 eBay Alternatives Beckon at These Online Auction Sites [Electronic resource] / G. Pickard-Whitehead. – Access mode : <https://smallbiztrends.com/2017/03/ebay-alternatives.html>
5. Fisher, S. 10 Best Online Auction Websites of 2021 [Electronic resource] / S. Fisher. – Access mode : <https://www.lifewire.com/online-auction-website-3482641>
6. Electronics, Cars, Fashion, Collectibles & More – eBay [Electronic resource]. – Access mode : <https://www.ebay.com/>
7. Top 60 Best Online Auction Sites – Complete Review [Electronic resource]. – Access mode : <https://www.charityauctionstoday.com/p/charity-auction-sites/>
8. GSA Auctions, General Service Administration, Government Site for Auctions [Electronic resource]. – Access mode : <https://gsaauctions.gov/gsaauctions/gsaauctions/>
9. Haan, K. The Best Online Auction Websites for 2021 [Electronic resource] / K. Haan. – Access mode : <https://www.investopedia.com/best-online-auction-websites-5114546>
10. Фримен, А. ASP.NET Core MVC 2 с примерами на C# для профессионалов [Текст] / А. Фримен. – М. : Вильямс, 2016. – 1008 с.
11. Чамберс, Дж. ASP.NET Core. Разработка приложений [Текст] /

Джеймс Чамберс, Дэвид Пэкетт, Саймон Тиммс. – СПб. : Питер, 2017. – 464 с.

12. ASP.NET Definition [Electronic resource]. – Access mode : <https://techterms.com/definition/aspnet>

13. MySQL vs. MSSQL – Performance and Main Differences Between Database and Servers [Electronic resource]. – Access mode : <https://www.dnsstuff.com/mysql-vs-mssql-performance>

14. Now UI Kit [Electronic resource] : Free Bootstrap 4 UI Kit. – Access mode : <https://www.creative-tim.com/product/now-ui-kit#>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення направлене не на масовий продаж товарів, а на його індивідуальні форми. Серед речей, продаж яких виконується, можуть бути індивідуальні вироби, тобто вироби ручної роботи, речі, які вже були у вжитку, але несуть певну цінність.

A.1 Підстави для розробки

Підставою для розробки було завдання на дипломну кваліфікаційну роботу за темою «Програмне забезпечення продажу речей за конкурентною ціною» на основі наказу № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для продажу речей за ціною, яка буде встановлена шляхом урівноваження попиту різних покупців. Окрім того програма дозволяє виконувати індивідуальним продавцям пошук покупців на ті речі, які вони бажають продати. У свою чергу покупці можуть виконувати пошук потрібних їм речей.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик складаються з наступних:

- визначення інформації про річ, яка буде продаватися;
- редагування інформації про річ;
- відображення всіх речей користувача;
- створення оголошення про продаж речі;

- редагування оголошення;
- відображення всіх оголошень продавця;
- відображення всіх власних оголошень;
- відображення активних власних оголошень;
- відображення виконаних власних оголошень;
- створення пропозиції про купівлю речі;
- перегляд пропозицій про купівлю речі;
- внесення грошей на рахунок;
- відображення повних даних оголошення;
- відображення речей з найбільшою різницею між конкурентною і базовою вартостями;
- деактивація оголошення;
- пошук речей;
- відображення оголошень за категоріями;
- відображення власних пропозицій про купівлю речей;
- скасування власної пропозиції про купівлю речі;
- відображення власних придбаних речей;
- створення відгуків про продавця;
- перегляд відгуків на продавця;
- перегляд власних відгуків;
- авторизація облікового запису;
- реєстрація облікового запису.

A.3.2 Вимоги до надійності

Надійність роботи системи має забезпечуватися надійністю організації технічної складової серверної сторони, надійністю системи управління базами даних та обробленням даних, які вводить користувач для уникнення проблем з перетворенням типів.

A.3.3 Умови експлуатації

Для експлуатації програми необхідно забезпечити виділений сервер або хостинг, який буде відповідати необхідній кількості запитів на оброблення, що передбачається відповідним масштабом системи.

A.3.4 Вимоги до складу та параметрів технічних засобів

Апаратні вимоги до сервера:

- жорсткий диск з вільним простором – 1 Гб;
- оперативна пам'ять – 4 Гб;
- процесор: 4 ядра, 2,8 ГГц частота.

Клієнт представлений браузером користувача. Для виконання вимог до нього потрібний встановлений сучасний браузер з доступом до мережі інтернет.

A.4 Порядок контролю та приймання

Контроль за виконанням дипломної кваліфікаційної роботи покладається на керівника. Остаточне рішення приймається екзаменаційною комісією під час захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Thing.cs

```

using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Thing
    {
        public int Id{ get; set; }
        public string Name{ get; set; }
        public string Information{ get; set; }
        public int? UserId{ get; set; }
        public User User{ get; set; }
        public virtual ICollection<Ad> Ads{ get; set; }
        public virtual ICollection<Category> Categories{ get;
set; }

        public virtual ICollection<Image> Images{ get; set; }
        public Thing()
        {
            Categories = new List<Category>();
            Ads = new List<Ad>();
            Images = new List<Image>();
        }
    }
}

```

Б.2 Текст файла Ad.cs

```

using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Ad
    {
        public int Id { get; set; }
        public DateTime Created{ get; set; }
        public DateTime Deadline{ get; set; }
        public float Price{ get; set; }
        public int Count{ get; set; }
        public int Status{ get; set; }
    }
}

```

```

        public string City{ get; set; }
        public string Delivery{ get; set; }
        public float Limit{ get; set; }
        public int? ThingId{ get; set; }
        public Thing Thing{ get; set; }
        public virtual ICollection<Proposition> Propositions{
get; set; }

        public Ad()
        {
            Propositions = new List<Proposition>();
        }
    }
}

```

Б.3 Текст файла Image.cs

```

using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Image
    {
        public int Id { get; set; }
        public string Name{ get; set; }
        public string Source{ get; set; }
        public int? ThingId{ get; set; }
        public Thing Thing{ get; set; }
    }
}

```

Б.4 Текст файла User.cs

```

using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Login{ get; set; }
    }
}

```

```

        public string Password{ get; set; }
        public string Name{ get; set; }
        public string Account{ get; set; }
        public float Score{ get; set; }
        public float Balsum{ get; set; }
        public virtual ICollection<Proposition> Propositions{
get; set; }

        public virtual ICollection<Thing> Things{ get; set; }
        public virtual ICollection<Review> Reviews{ get; set; }
        public virtual ICollection<Review> ReviewsFrom{ get;
set; }

        public User()
        {
            Propositions = new List<Proposition>();
            Things = new List<Thing>();
            Reviews = new List<Review>();
            ReviewsFrom = new List<Review>();
        }
    }
}

```

Б.5 Текст файла Review.cs

```

using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Review
    {
        public int Id{ get; set; }
        public DateTime Created{ get; set; }
        public string Information{ get; set; }
        public float Score{ get; set; }
        public int? UserId{ get; set; }
        public User User{ get; set; }
        public int? FromuserId{ get; set; }
        public User Fromuser{ get; set; }
    }
}

```

Б.6 Текст файла Proposition.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Proposition
    {
        public int Id{ get; set; }
        public DateTime Created{ get; set; }
        public float Price{ get; set; }
        public int Count{ get; set; }
        public int Status{ get; set; }
        public int? UserId{ get; set; }
        public User User{ get; set; }
        public int? AdId{ get; set; }
        public User Ad{ get; set; }
    }
}
```

Б.7 Текст файла Category.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Category
    {
        public int Id{ get; set; }
        public string Name{ get; set; }
        public virtual ICollection<Thing> Things{ get; set; }
        public Category()
        {
            Things = new List<Thing>();
        }
    }
}
```

Б.8 Текст файла Payment.cs

```
using System;
using System.ComponentModel.DataAnnotations;

namespace BurenkoSeller.Models
{
    public class Payment
    {
        public int Id{ get; set; }
        public int? PropositionId{ get; set; }
        public Proposition Proposition{ get; set; }
        public float Pay{ get; set; }
        public DateTime Created{ get; set; }
    }
}
```

Б.9 Текст файла BurenkoSellerContext.cs

```
using BurenkoSeller.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace BurenkoSeller.Models
{
    public class BurenkoSellerContext : DbContext
    {
        public DbSet<Category> Categories{ get; set; }
        public DbSet<Thing> Things{ get; set; }
        public DbSet<Image> Images{ get; set; }
        public DbSet<Ad> Ads{ get; set; }
        public DbSet<User> Users{ get; set; }
        public DbSet<Proposition> Propositions{ get; set; }
        public DbSet<Review> Reviews{ get; set; }
        public DbSet<Payment> Payments{ get; set; }
        public BurenkoSellerContext() : base("DefaultConnection")
        {}
        protected override void OnModelCreating(DbModelBuilder
mb)
```

```

    {
        mb.Entity<Thing>().HasMany(c => c.Category)
            .WithMany(s => s.Things)
            .Map(t => t.MapLeftKey("ThingId"))
            .MapRightKey("CategoryId")
            .ToTable("ThingCategory"));
        mb.Entity<Category>().ToTable("Category");
        mb.Entity<Thing>().ToTable("Thing");
        mb.Entity<Image>().ToTable("Image");
        mb.Entity<Ad>().ToTable("Ad");
        mb.Entity<User>().ToTable("User");
        mb.Entity<Proposition>().ToTable("Proposition");
        mb.Entity<Review>().ToTable("Review");
        mb.Entity<Payment>().ToTable("Payment");
    }
}

```

Б.10 Текст файла ThingController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using BurenkoSeller.Models;

namespace BurenkoSeller.Controllers
{
    public class ThingController : Controller
    {
        private readonly BurenkoSellerContext bsc;

        public ThingController (BurenkoSellerContext cont)
        {
            bsc = cont;
        }

        [Authorize]
        public async Task<IActionResult> Index()
        {

```

```

        IQueryable<User> users = bsc.Users;
        var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
        var things = bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories). Include(a => a.Images);
        ViewData["Things"] = things.Where(i => i.UserId ==
user.Id);

        return View();
    }

    [Authorize]
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }
        var thing = await bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories). Include(a =>
a.Images).FirstOrDefaultAsync(b => b.Id == id);
        if (thing == null)
        {
            return NotFound();
        }

        return View(thing);
    }

    [Authorize]
    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    public async Task<IActionResult> Create([Bind("Name,
Information")] Thing thing)
    {
        if (ModelState.IsValid)
        {
            IQueryable<User> users = bsc.Users;
            var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
            thing.UserId = user.Id;
            bsc.Add(thing);

```

```

        await bsc.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(thing);
}

[Authorize]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var thing = await bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories).Include(a =>
a.Images).FirstOrDefaultAsync(b => b.Id == id);
    if (thing == null)
    {
        return NotFound();
    }
    return View(thing);
}

[HttpPost]
public async Task<IActionResult> Edit(int id, [Bind("Id,
Name, Information")] Thing thing)
{
    if (id != thing.Id)
    {
        return NotFound();
    }
    if (ModelState.IsValid)
    {
        try
        {
            bsc.Update(thing);
            await bsc.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ThingExists(thing.Id))
            {
                return NotFound();
            }
            else

```

```

        {
            throw;
        }
    }
    return RedirectToAction(nameof(Index));
}
    ViewData["categories"] = new
SelectList(bsc.Categories, "Id", "Name");
    return View(thing);
}

private bool ThingExists(int id)
{
    return bsc.Thing.Any(e => e.Id == id);
}

}
}

```

Б.11 Текст файла AdController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using BurenkoSeller.Models;

namespace BurenkoSeller.Controllers
{
    public class AdController : Controller
    {
        private readonly BurenkoSellerContext bsc;

        public AdController (BurenkoSellerContext cont)
        {
            bsc = cont;
        }

        [Authorize]
        public async Task<IActionResult> Index()

```

```

        {
            IQueryable<User> users = bsc.Users;
            var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
            var things = bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.UserId == user.Id);
            List<Ad> ads = new List<Ad>();
            foreach (var thing in things)
            {
                ads.Add(bsc.Ads.Include(a =>
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id));
            }
            ViewData["Ads"] = ads;
            ViewData["Things"] = things;
            return View();
        }

        [Authorize]
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var ad = bsc.Ads.Include(a => a.Thing).Include(a =>
a.Propositions).Where(i => i.Id == id);
            if (ad == null)
            {
                return NotFound();
            }

            return View(thing);
        }

        [Authorize]
        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Create([Bind("Id,
Deadline, Price, Count, City , Delivery, Limit, ThingId")] Ad ad)

```

```

{
    if (ModelState.IsValid)
    {
        ad.Created = DateTime.Now;
        ad.Status = 0;
        bsc.Add(ad);
        await bsc.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(ad);
}

[Authorize]
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var ad = await bsc.Ads.Include(a =>
a.Thing).Include(a => a.Propositions).FirstOrDefaultAsync(b => b.Id ==
id);

    if (ad == null)
    {
        return NotFound();
    }
    return View(ad);
}

[HttpPost]
public async Task<IActionResult> Edit(int id, [Bind("Id,
Deadline, Price, Count, Status, City , Delivery, Limit, ThingId")] Ad
ad)
{
    if (id != ad.Id)
    {
        return NotFound();
    }
    if (ModelState.IsValid)
    {
        try
        {
            bsc.Update(ad);
            await bsc.SaveChangesAsync();
        }
    }
}

```

```

        catch (DbUpdateConcurrencyException)
        {
            if (!AdExists(ad.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["things"] = new SelectList(bsc.Things,
    "Id", "Name");
    return View(ad);
}

public async Task<IActionResult> Deacticate(int id)
{
    if (id != ad.Id)
    {
        return NotFound();
    }
    if (ModelState.IsValid)
    {
        try
        {
            ad.Status = 2;
            bsc.Update(ad);
            await bsc.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!AdExists(ad.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
    }
    return RedirectToAction(nameof(Index));
}

```

```

        return View(ad);
    }

    public async Task<IActionResult> UserAds(int? id)
    {
        var things = bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.UserId == id);
        List<Ad> ads = new List<Ad>();
        foreach (var thing in things)
        {
            ads.Add(bsc.Ads.Include(a =>
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id));
        }
        ViewData["Ads"] = ads;
    }

    public async Task<IActionResult> ActiveAds()
    {
        IQueryable<User> users = bsc.Users;
        var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
        var things = bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.UserId == user.Id);
        List<Ad> ads = new List<Ad>();
        foreach (var thing in things)
        {
            data = bsc.Ads.Include(a =>
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id);
            ads.Add(data.Where(i => i.Status == 0));
        }
        ViewData["Ads"] = ads;
    }

    public async Task<IActionResult> DoneAds()
    {
        IQueryable<User> users = bsc.Users;
        var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
        var things = bsc.Things.Include(a =>
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.UserId == user.Id);

```

```

        List<Ad> ads = new List<Ad>();
        foreach (var thing in things)
        {
            data = bsc.Ads.Include(a
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id);

            ads.Add(data.Where(i => i.Status == 1));
        }
        ViewData["Ads"] = ads;
    }

    public async Task<IActionResult> CategoryAds(int id)
    {
        IQueryable<User> users = bsc.Users;
        var user = await users.FirstOrDefaultAsync(a =>
a.Login == User.Identity.Name);
        var things = bsc.Things.Include(a
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.Categories.Contains(id));
        List<Ad> ads = new List<Ad>();
        foreach (var thing in things)
        {
            ads.Add(bsc.Ads.Include(a
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id));
        }
        ViewData["Ads"] = ads;
        return View();
    }

    public async Task<IActionResult> SearchAds(string name,
float price1, float price2)
    {
        var things = bsc.Things.Include(a
a.Ads).Include(a => a.Categories).Include(a => a.Images).Where(i =>
i.Name.Contains(name));
        List<Ad> ads = new List<Ad>();
        foreach (var thing in things)
        {
            var data = bsc.Ads.Include(a
a.Thing).Include(a => a.Propositions).Where(i => i.ThingId ==
thing.Id);

            data = data.Where(i => i.Price >= price1);
            data = data.Where(i => i.Price <= price2);
            ads.Add(data);
        }
    }

```

```
        }  
        ViewData["Ads"] = ads;  
        return View();  
    }  
  
    private bool AdExists(int id)  
    {  
        return bsc.Ad.Any(e => e.Id == id);  
    }  
}  
}
```

ДОДАТОК В
Слайди презентації

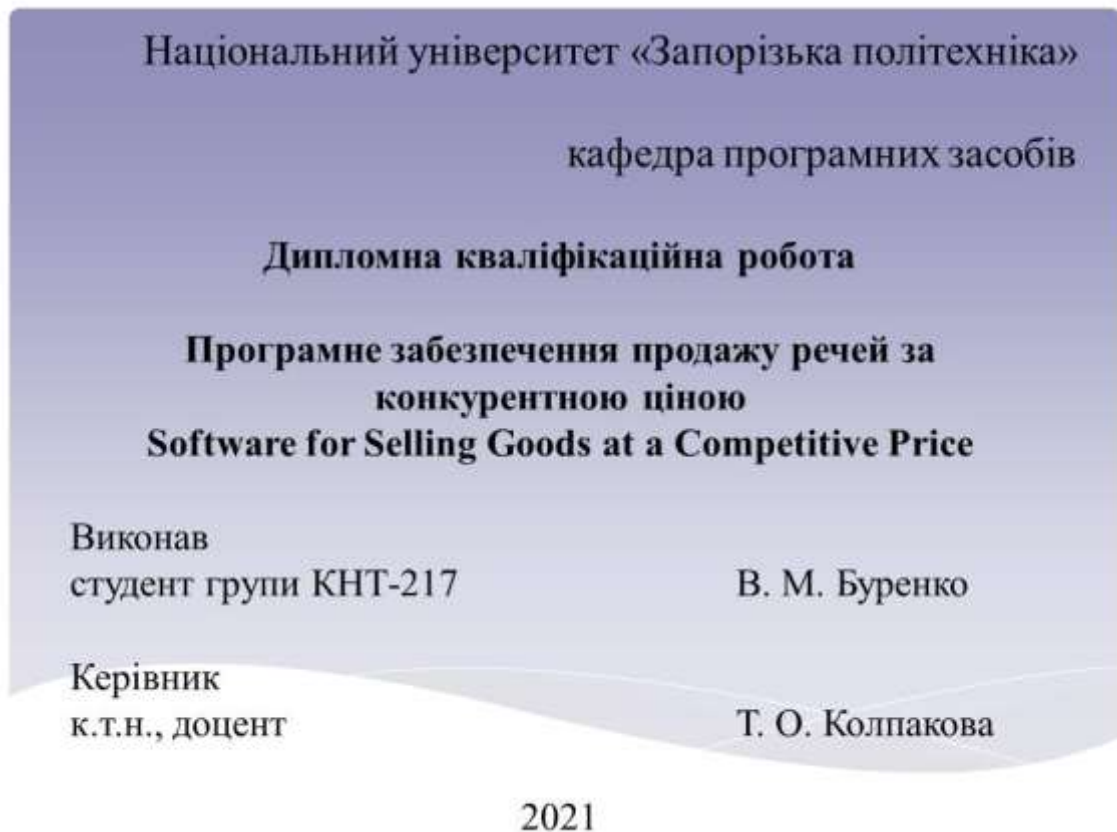
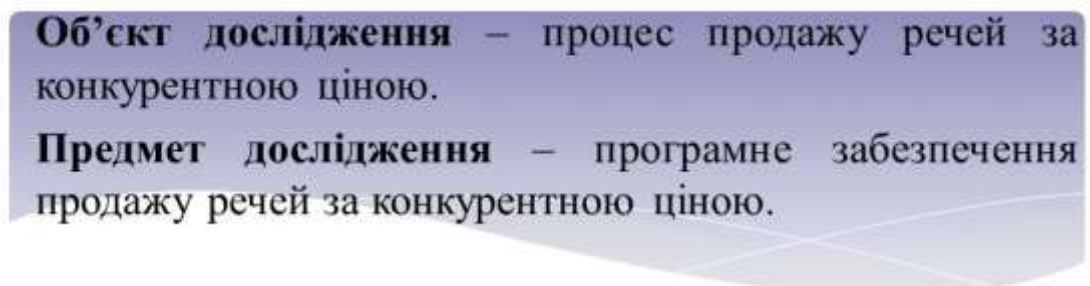


Рисунок В.1 – Слайд № 1



Мета роботи – розроблення програмного забезпечення продажу речей для встановлення конкурентної ціни на основі попиту покупців.

Завдання роботи:

- аналіз предметної області;
- проєктування програмного забезпечення;
- реалізація програмного забезпечення.

Рисунок В.2 – Слайд № 2



Рисунок В.3 – Слайд № 3



Рисунок В.4 – Слайд № 4

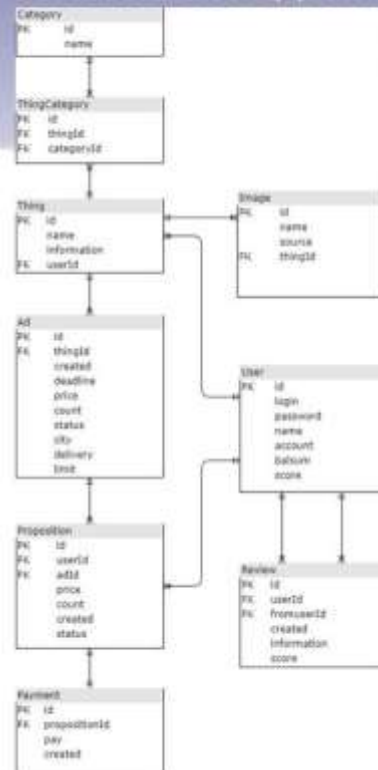
Вибір мови програмування

	C#	Java
Веброзробка	Так	Так
Вебфреймворк	ASP.NET	Spring
Автоматичне керування пам'яттю	Так	Так
Кросплатформність	Так	Так
Швидкість виконання програм	Більша	Менша

5

Рисунок В.5 – Слайд № 5

Схема бази даних



6

Рисунок В.6 – Слайд № 6



Рисунок В.7 – Слайд № 7

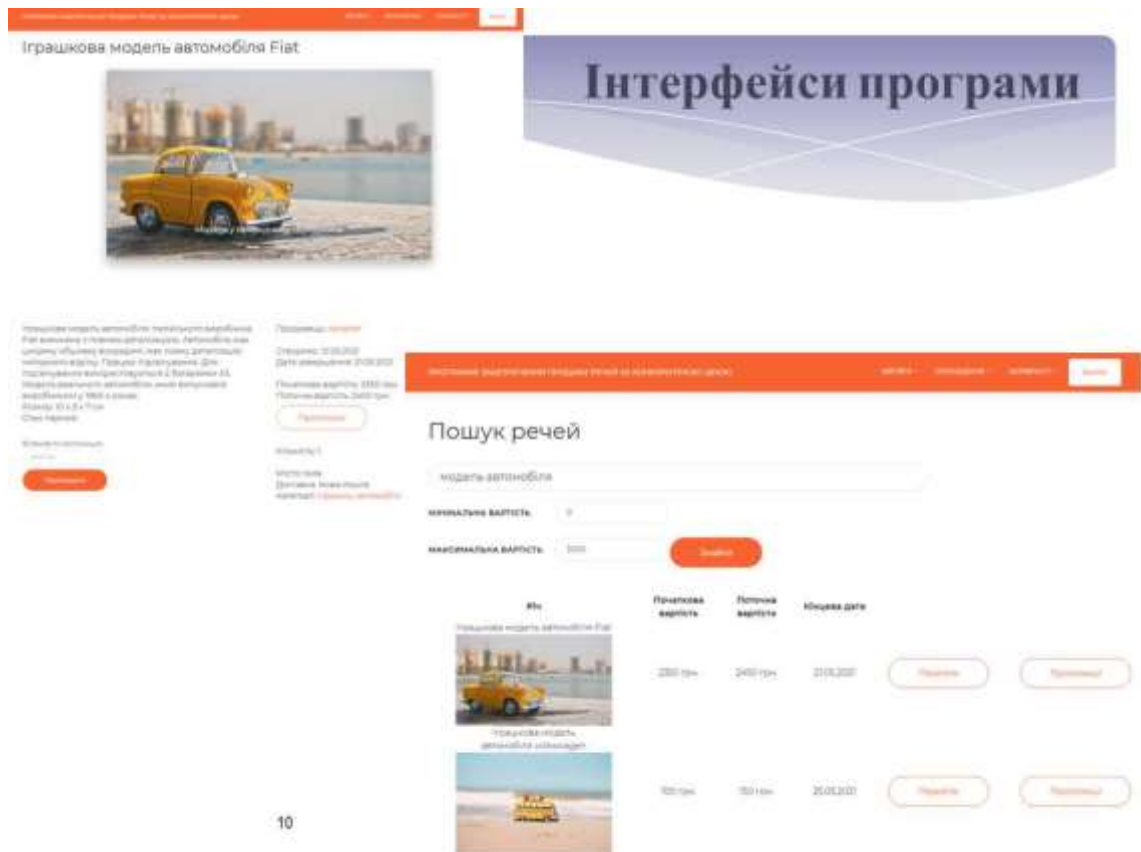


Рисунок В.8 – Слайд № 8



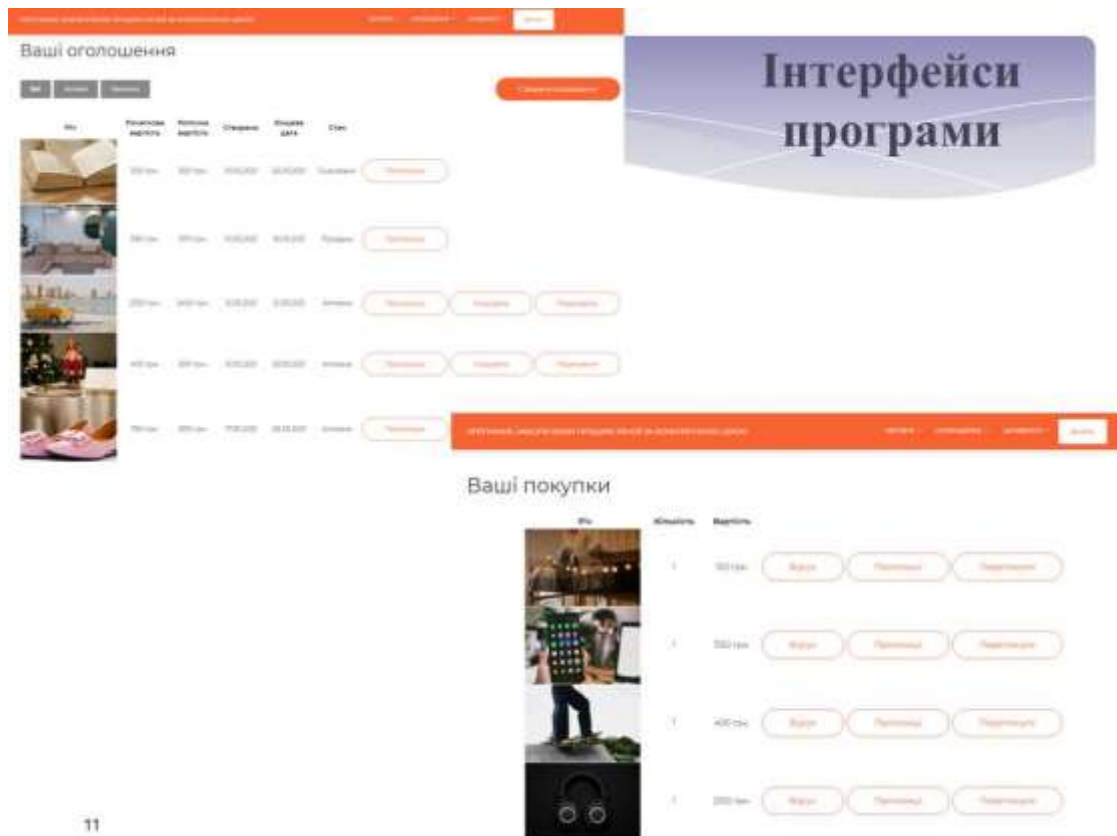
9

Рисунок В.9 – Слайд № 9



10

Рисунок В.10 – Слайд № 10



11

Рисунок В.11 – Слайд № 11

Висновки

Виконання дипломної кваліфікаційної роботи призвело до отримання наступних результатів:

- виконано аналіз проблеми продажу речей за конкурентною ціною, для чого розглянуто проблему встановлення конкурентної ціни, виконано огляд і порівняння програмних аналогів;
- виконано розробку архітектури програми, під час якої визначено функціональні вимоги до програмного забезпечення, промодельовано роботу з програмою на основі діаграм прецедентів, обрано мову програмування C#, технологію ASP.NET і СУБД Microsoft SQL Server для розроблення програмного забезпечення, визначено структуру БД для організації роботи з даними в програмі;
- реалізовано програмне забезпечення, в результаті чого описано алгоритм функціонування програми, алгоритм продажу речей за конкурентною ціною, логіку зміни станів оголошення про продаж речі, описано структуру програмного забезпечення, реалізовані класи програмного застосунку.

12

Рисунок В.12 – Слайд № 12