Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Система самостійного касового розрахунку з визначенням характеристик покупця

Виконав: студент 2 курсу, групи 2АКІТ-22м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Анатолій РИБАК
Ім'я ПРІЗВИЩЕ

Керівник: к.т.н., доцент кафедри КСУ
степінь, звання, посада

Тетяна ГРИЩУК
Ім'я ПРІЗВИЩЕ

«01» 12 2023 р.

Опонент: к.т.н., доцент каф. АІІТ
степінь, звання, посада

Ілона БОГАЧ
Ім'я ПРІЗВИЩЕ

«05» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
_Марія ЮХИМЧУК
«07» 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Рибаку Анатолію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи. Система самостійного касового розрахунку з визначенням характеристик покупця.

керівник роботи Грищук Тетяна Вікторівна

затверджені наказом ВНТУ від "18" вересня 2023 року №247

2. Строк подання студентом роботи 1. грудня 2023 року.

3. Вихідні дані до роботи: система управління базами даних – реляційна; мова запитів – SQL; вхідні дані – база даних із товарами та інформацією про них конкретного закладу, обліковими записами клієнтів, інформація про знижки для кожного покупця, особиста інформація клієнтів, зображення покупців для визначення можливості купівлі обмежених товарів та генерування маркетингових пропозицій; точність системи – 99%, швидкість відповіді системи – до 2 с.

4. Зміст розрахунково–пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури та алгоритмів роботи програмного продукту; розробка методу визначення віку та статі покупця; розробка архітектури системи та програмного забезпечення; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний лист; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; новизна одержаних результатів; практична цінність одержаних результатів; використані фреймворки; алгоритм розпінавання віку та статі покупця; схема бази даних; блок-схема загального алгоритму; блок-схема алгоритму генерування знижок; тестування веб-додатку; апробація та публікація результатів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
5	Бурєннікова Н. В., д.е.н., професор кафедри ЕПВМ.		

7. Дата видачі завдання 09 жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Прим
		початок	закінчення	
1	Обґрунтування вибору методу розробки та постановка задачі	09.10.2023	09.10.2023	
2	Розробка методу визначення віку та статі	17.10.2023	17.10.2023	
3	Розробка архітектури та алгоритмів роботи системи	22.10.2023	04.11.2023	
4	Розробка програмних компонент для системи самостійного касового розрахунку	05.11.2023	16.11.2023	
5	Тестування програмного забезпечення	17.11.2023	23.11.2023	
6	Розрахунок економічної частини	24.11.2023	26.11.2023	
7	Графічні матеріали	27.11.2023	27.11.2023	

Студент Анатолій Р

(підпис) (Ім'я ПРІЗ

Керівник роботи Тетяна ГРИ

(підпис) (Ім'я ПРІЗ

АНОТАЦІЯ

УДК 004.8.9

Рибак А. Ю. Система самостійного касового розрахунку з визначенням характеристик покупця. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2023. 157 с.

На укр. мові. Бібліогр.: 56; рис.: 56; табл. 20.

У магістерській кваліфікаційній роботі розроблено систему самостійного касового розрахунку з визначенням характеристик покупця. У частині з обґрунтуванням вибору методу розробки роботи розглянуто аналіз стану імплементації програмного забезпечення для систем самообслуговування, проведено аналіз аналогів, обґрунтована доцільність роботи. У теоретично-методичній частині було оглянуто методи розв’язання поставленої задачі. У практичній частині розроблено архітектуру та саме програмне забезпечення, розроблено систему розпізнавання характеристик покупців, протестовано та наведено результати його роботи. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 18 плакатів із результатами роботи.

Ключові слова: автоматизація, система самообслуговування, система розпізнавання характеристик, програмне забезпеченя.

ABSTRACT

UDC 004.8.9

Rybak A. Y. Self-checkout system with customer characteristics determination. Master's thesis in the field of 151 – Automation and Computer-Integrated Technologies, educational program – Intelligent Computer Systems. Vinnytsia: VNTU, 2023. 157 p.

In Ukrainian. Bibliography: 56; figures: 56; tables: 20.

The master's thesis presents the development of a self-checkout system with the determination of customer characteristics. The section on justifying the choice of the development method includes an analysis of the implementation state of self-service software systems, a comparison of analogs, and substantiation of the work's relevance. The theoretical-methodological part reviews the methods for solving the formulated problem. In the practical part, the architecture and software itself are developed, a system for recognizing customer characteristics is created, tested, and the results of its operation are presented. The economic section analyzes the technical level and calculates the cost of implementing the development. The illustrative part consists of 18 posters with the results of the work.

Keywords: automation, self-service system, characteristics recognition system, software.

ЗМІСТ

ВСТУП 4
1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ 7
1.1 Огляд систем автоматизованого розрахунку в закладах торгівлі 7
1.2 Аналіз стану імплементації програмного забезпечення для системи самостійного касового розрахунку 17
1.3 Порівняльний аналіз аналогів 20
1.4 Огляд методів розв’язання поставленої задачі 25
1.5 Постановка задач розробки системи самостійного касового розрахунку 28
1.6 Висновки 29
2 РОЗРОБКА МЕТОДУ ВИЗНАЧЕННЯ ВІКУ ТА СТАТІ ПОКУПЦЯ 30
2.1 Варіантний аналіз існуючих методів визначення віку та статі людини 30
2.2 Огляд технічних систем, що використовують моделі штучного інтелекту для обробки зображень 32
2.3 Варіантний аналіз існуючих програмних засобів для визначення віку та статі людини 35
2.4 Розробка методів та алгоритмів визначення характеристик покупця 39
2.5 Розробка прототипу підсистеми визнання віку та статі людини 42
2.6 Оцінка якості роботи розробленого методу 45
2.7 Висновки 47
3 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ 49
3.1 Розробка архітектури програмного продукту 49
3.2 Розробка алгоритмів роботи додатку 57
3.3 Розробка алгоритму рекомендації додаткових товарів на основі індивідуальних характеристик покупця 60
3.4 Розробка бази даних 62
3.5 Розробка структури графічного інтерфейсу користувача 68

3.6 Висновки 75
4 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ СИСТЕМИ САМОСТІЙНОГО КАСОВОГО РОЗРАХУНКУ 76
4.1 Варіантний аналіз і обґрунтування вибору мови програмування 76
4.2 Вибір середовища розробки 81
4.3 Програмна реалізація системи 86
4.4 Тестування програмного забезпечення 98
4.5 Висновки 106
5 ЕКОНОМІЧНА ЧАСТИНА 107
5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки 107
5.2 Розрахунок узагальненого коефіцієнта якості розробки 111
5.3 Розрахунок витрат на проведення науково-дослідної роботи 112
5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором 124
5.5 Висновок 129
ВИСНОВКИ 130
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 131
ДОДАТКИ 137
Додаток А. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень .....
Додаток Б. Технічне завдання. .....
Додаток В. Лістинг програми 143
Додаток Г. Ілюстративна частина. .....

ВСТУП

Актуальність теми. В Україні спостерігається значний ріст попиту на системи самообслуговування. Відзначається, що в українському суспільстві є зростаючий інтерес до новітніх технологій, особливо в галузі роздрібної торгівлі. Впровадження систем самостійного касового розрахунку є стратегічним кроком для підтримки цього технологічного розвитку, що, в свою чергу, покращує конкурентоспроможність та ефективність роздрібних підприємств в Україні.

Системи самостійного касового розрахунку є важливим складовим видом самообслуговування. Цей вид обслуговування стає надзвичайно актуальним та зручним рішенням, особливо в умовах великого потоку клієнтів, допомагаючи уникнути перевантажень на традиційних касах. Проект "Самокаса" у мережі магазинів "Сільпо" в Україні є яскравим прикладом успішного впровадження систем самообслуговування.

Розпізнавання віку покупців у касах самообслуговування виявляється критично важливим аспектом управління продажами обмежених товарів. Ця функціональність не лише допомагає дотримуватись законодавства щодо продажу обмежених продуктів, але й підвищує рівень безпеки та контролю над їхнім обігом. Завдяки цій можливості каси точно визначають вікові групи покупців, які мають право на придбання конкретних товарів. Це зменшує ризик неправомірного продажу заборонених продуктів неповнолітнім та сприяє відповідальній торгівлі. Ця можливість дозволяє не лише відповідати законодавству, але і підтримувати високий стандарт безпеки й етичності у торговій сфері.

Отже, Україна виявляє великий попит на системи самообслуговування, особливо в роздрібній торгівлі. Впровадження новітніх технологій, таких як систем самостійного касового розрахунку з визначенням характеристик покупця, є стратегічним кроком для підвищення конкурентоспроможності підприємств, а також посприяє покращенню обслуговування та підвищенню рівня безпеки у торговій сфері.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є підвищення швидкості та зручності процесу купівлі товарів у різних точках роздрібної торгівлі шляхом розробки програмного забезпечення для систем самостійного касового розрахунку, а також підвищення попиту на більш широкий асортимент товарів роздрібної торгівлі за рахунок удосконалення алгоритмів нарахування знижок та розпізнання характеристик користувача, не турбуючись за безпеку продажу обмежених за віком товарів.

Відповідно до поставленої мети в магістерській кваліфікаційній роботі потрібно вирішити наступні завдання:

- виконати аналіз сучасних систем самостійного касового розрахунку;
- розробити алгоритм додаткової верифікації користувача;
- розробити архітектуру системи;
- розробити метод та алгоритм роботи рекомендаційної системи на основі характеристик покупця;
- провести варіантний аналіз та розробити базу даних системи;
- розробити програмне забезпечення для системи самостійного касового розрахунку базуючись на створеній архітектурі та алгоритмах.

Об’єктом дослідження є процес самостійного касового розрахунку.

Предметом дослідження є методи та алгоритми додаткової верифікації покупців та рекомендації товарів.

Методи дослідження. У магістерській кваліфікаційній роботі використовувалися: теорія алгоритмів для розробки та вдосконалення програмного забезпечення; теорія баз даних для розробки бази даних з товарами точки роздрібної торгівлі, персональними акаунтами клієнтів з їх даними та історією покупок; теорія штучного інтелекту для розробки системи визначення характеристик покупця; методи тестування системи самообслуговування для перевірки коректної роботи програмного продукту.

Новизна отриманих результатів.

Удосконалено алгоритм роботи системи самостійного касового обслуговування, який, на відміну від існуючих, враховує індивідуальні ознаки

покупців, що дозволяє покращити процес обслуговування за рахунок автоматичної верифікації віку та статі покупця.

Практична цінність отриманих результатів.

Практичною цінністю одержаних результатів є програмний додаток системи самостійного касового розрахунку для підвищення зручності, швидкості та безпеки здійснення покупок у різних точках роздрібної торгівлі.

Особистий внесок здобувача. Описані у магістерській кваліфікаційній роботі отримані результати були здобуті автором на самостійній основі.

Апробація результатів роботи. Представлені в роботі результати апробовані в результаті участі в конференції Всеукраїнська науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2024)».

Публікації. Рибак А.Ю. «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», 2023. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19721.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Огляд систем автоматизованого розрахунку в закладах торгівлі

Сучасний розвиток торгівлі вимагає постійного вдосконалення та впровадження новітніх технологій для полегшення та удосконалення процесів розрахунку в торгівельних закладах. Основна мета цього розділу полягає в дослідженні різноманітних систем автоматизованого розрахунку, які використовуються в торгівельних точках та магазинах. У зазначеному контексті, класифікація способів розрахунку в торгівельних закладах розглядається з різних поглядів, включаючи простий касовий розрахунок, використання карток лояльності та програм знижок, системи самообслуговування та інші інноваційні технології. Детальний аналіз допоможе визначити переваги та недоліки кожної системи, а також їх вплив на якість обслуговування та задоволення покупців.

Основна увага приділяється етапам розрахунку в торгівельних закладах, зокрема процесу сканування товарів, розрахунку та видачі чека, а також різноманітним методам оплати. Подробиці кожного етапу дозволяють зрозуміти технічні та функціональні аспекти впровадження автоматизованих розрахункових систем. Однак, разом із зростанням автоматизації, з'являються проблеми, такі як ризик використання банківських карток іншими особами та необхідність додаткової ідентифікації покупця. Аналіз цих аспектів допомагає розробити ефективні заходи безпеки та захисту конфіденційності даних покупців [1].

Важливо відзначити, що вивчення автоматизованих систем розрахунків у торгівлі є актуальним та важливим напрямком, оскільки воно сприяє покращенню якості обслуговування, ефективності та безпеці операцій для всіх учасників торговельного процесу. Із розвитком технологій і зростанням конкуренції на ринку торгівлі, важливість впровадження та оптимізації систем розрахунків стає ключовою для забезпечення конкурентоспроможності підприємств. Такі системи не лише полегшують розрахункові процеси, а й впливають на задоволення

клієнтів, їх комфорт та взаємодію із закладом. У цьому контексті дослідження різноманітних аспектів систем автоматизованого розрахунку набуває стратегічного значення для бізнесу та ринку в цілому.

Поглиблене вивчення систем автоматизованого розрахунку дозволить визначити оптимальні стратегії впровадження та використання цих технологій у торгівельних закладах. Аналіз особливостей кожного методу розрахунку, врахування їх впливу на ефективність обслуговування та задоволення покупців, а також урахування можливих ризиків та викликів дозволять визначити оптимальний баланс між інноваційністю та надійністю в сфері розрахункових систем торгових закладів [2].

У контексті глобального розвитку електронних технологій і змін у споживчому підході, розгляд автоматизованих систем розрахунку набуває значущості для розуміння впливу цих технологій на моделі бізнесу, споживчі звички та загальну конкурентоспроможність сучасних торговельних закладів. За останні роки спостерігається стрімке вдосконалення і впровадження новітніх розрахункових систем, що ставить перед дослідниками завдання ретельного вивчення та аналізу цих технологій в контексті їхнього впливу на функціонування та успіх торговельних закладів.

Огляд розрахункових систем включає аналіз не лише технічних аспектів використання, але й розгляд взаємодії цих систем з покупцями, вплив на швидкість обслуговування та зручність для клієнтів. Крім того, розгляд проблем, пов'язаних із безпекою та конфіденційністю даних, дозволить визначити можливі ризики та способи їх вирішення для забезпечення найвищого рівня захисту. Такий виважений та комплексний аналіз систем автоматизованого розрахунку в торговельних закладах відкриє нові горизонти для подальших досліджень у цій області та надасть практичні рекомендації для вдосконалення ефективності та безпеки розрахункових процесів у сучасній торгівлі.

В першу чергу розглянемо одну з можливих класифікацій способів розрахунку в торгівельних закладах. Сучасні технології та постійний розвиток торгівельних закладів вимагають удосконалення систем розрахунків.

Класифікація способів розрахунку в торгівельних закладах включає різноманітні методи, серед яких ключовими є простий касовий розрахунок, касовий розрахунок з додатковою ідентифікацією за карткою магазину, системи самообслуговування та інноваційні технології оплати.

Класифікують такі способи розрахунку в торгівельних закладах:

- простий касовий розрахунок;
- касовий розрахунок з додатковою ідентифікацією за карткою магазину;
- системи самообслуговування;
- інші інноваційні системи розрахунку.

Простий касовий розрахунок є одним із традиційних методів проведення фінансових операцій в торгівельних закладах. Цей метод передбачає використання касового обладнання та участь касира у процесі обслуговування клієнтів. Нижче розглянемо основні аспекти та учасників простого касового розрахунку [3].

Учасниками є касир або касирка: Основна постачальник послуги, який/яка взаємодіє з клієнтами та проводить операції за допомогою касового обладнання.

Основні принципи та переваги:

- Легкість використання: простий процес розрахунку спрощує взаємодію з касовим обладнанням, що робить його доступним для широкого кола працівників;
- Низькі витрати впровадження: в порівнянні з іншими системами, витрати на впровадження простого касового розрахунку є досить невеликими;
- Швидкий розрахунок: процес оплати відбувається швидко, що забезпечує ефективність обслуговування та скорочення черг.

Етапи процесу оплати покупок за допомогою простого касового розрахунку:

- Сканування товарів: касир сканує штрих-коди на товарах, щоб внести їх до касового чека;
- Обчислення суми до оплати: система автоматично обчислює загальну суму до сплати за вибрані товари;

- Прийняття оплати: клієнт сплачує суму готівкою або іншими доступними способами оплати;
- Видача чека: касир видає клієнту чек, в якому вказана детальна інформація про покупки та оплату.

Простий касовий розрахунок забезпечує швидкий та ефективний процес обслуговування, але його функціональність обмежена порівняно з більш сучасними системами розрахунку.

Картки лояльності та програми знижок розширюють можливості касового розрахунку. Ці картки не лише служать для розрахунків, але й надають покупцеві доступ до різноманітних акцій, знижок та нагород.

Опис процесу розрахунку з використанням карток магазину та їх вплив на покупця: покупець використовує свою картку лояльності під час розрахунку, що дозволяє системі ідентифікувати його та автоматично застосовувати знижки чи накопичені бонуси. Це стимулює лояльність клієнтів та збільшує їх задоволеність від покупок. Крім того, розвиток цифрових технологій сприяє впровадженню мобільних додатків, які інтегрують картки лояльності, роблячи процес розрахунків ще більш зручним та швидшим для клієнтів [4].

Системи самообслуговування надають покупцям можливість самостійно сканувати товари за допомогою спеціальних пристроїв. Це зменшує черги та сприяє швидшому обслуговуванню.

Опис процесу самообслуговування та його переваги для покупців і магазинів: покупець самостійно сканує товари, обирає спосіб оплати та завершує оплату без участі касира. Це збільшує ефективність обслуговування та надає покупцям вищу ступінь самостійності. Однією з можливих актуальних тенденцій є впровадження систем розпізнавання обличчя, що забезпечує ще швидший та безконтактний процес розрахунку. Це підвищує рівень зручності та сучасності технологій у торгівельних закладах.

Огляд новітніх технологій оплати в торгівельних закладах: інноваційні технології включають безконтактні платежі, мобільні додатки та інші рішення. Вони розширюють можливості оплати та забезпечують високий рівень зручності

для покупців. Впровадження новітніх технологій значно полегшує процес покупки, зменшує час розрахунків та стимулює розвиток електронних форм оплати. Зокрема, безконтактні платежі стали популярними через їхню простоту та високий рівень безпеки. Вони дозволяють покупцеві провести оплату, просто доторкнувшись до терміналу, що підвищує ефективність та спрощує весь процес. Загальною тенденцією є поступове відмовлення від готівки на користь електронних форм оплати. Це не лише забезпечує безпеку фінансових операцій, але і створює зручні умови для покупців та підвищує ефективність торговельних закладів. Враховуючи вищезазначені аспекти, стає очевидним, що розвиток і впровадження новітніх технологій у сфері розрахунків є ключовим елементом стратегії торговельних закладів для забезпечення конкурентоспроможності та відповіді на зростаючі очікування споживачів. Оптимізація розрахункових процесів, підвищення зручності для клієнтів та впровадження інновацій – це крок у майбутнє, що дозволить торговельним закладам не лише залишатися в грі, а й активно визначати правила гри на ринку.

Забезпечення високої ефективності обслуговування є ключовим аспектом систем автоматизованого розрахунку в торгівлі. Ефективність визначається швидкістю, надійністю, інтеграцією та легкістю користування.

Характеристики:

- Швидкість виконання операцій: Мінімізація часу на сканування товарів та розрахунок.
- Надійність та стабільність: Беззбійна робота та відновлення після помилок.
- Інтеграція з іншими системами: Ефективна взаємодія з іншими модулями.
- Легкість користування: Інтуїтивний інтерфейс та швидкий доступ.

Вимірювання ефективності:

- Час обробки транзакцій: Від початку до завершення операції.
- Транзакції за годину: Загальна продуктивність системи.
- Сприйняття клієнтів: Зворотній зв'язок стосовно швидкості та якості обслуговування.

- Використання ресурсів: Ефективність використання обчислювальних ресурсів.

Для досягнення високої ефективності важливо вдосконалювати систему відповідно до потреб користувачів та ринкових тенденцій.

У таблиці 1.1 представлено порівняння різних способів розрахунку в торгівельних закладах.

Таблиця 1.1 – порівняння способів розрахунку в торгівельних закладах

Аспекти розрахунку	Простий касовий розрахунок	Касовий розрахунок з додатковою ідентифікацією за карткою магазину	Системи самообслуговування	Інші інноваційні системи розрахунку
Легкість використання	+	+	+	+
Навчання та адаптація	+	+	-	+
Відсутність інвестицій в підготовку персоналу	-	-	+	+
Функціональність	-	+	+	+
Зручність для всіх вікових категорій	+	+	+	-
Лояльність та знижки	-	+	+	+
Ефективність обслуговування	-	-	+	+
Незалежність від Технологій	+	-	-	-
Зменшення черг та очікувань	-	-	+	+

Самостійність клієнтів	-	-	+	+
Різноманітність способів оплати	-	-	+	-
Визначення характеристик покупця	+	+	-	-
Підсумок	5	6	9	8

Аналіз таблиці порівняння різних систем розрахунку в торгівельних закладах вказує на те, що системи самообслуговування видаються перевагами у багатьох аспектах порівняно з іншими методами. Їхня легкість використання, зменшення черг, зручність для клієнтів та можливість самостійності роблять їх привабливими для сучасного споживача. Проте, важливо визначити, що деякі інноваційні системи мають обмеження, зокрема в аспекті ідентифікації покупців за їхнім віком. Це може стати проблемою в ситуаціях, коли необхідно обмежити доступ до певних товарів або послуг згідно з віковими обмеженнями.

Тому в даній магістерській роботі робиться акцент на розробці алгоритму визначення характеристик покупця за допомогою розпізнавання обличчя. Можна припустити, що такий підхід може стати вирішенням вказаної проблеми та допоможе покращити ефективність використання систем самообслуговування в торгівельних закладах. Таким чином, розробка та впровадження алгоритму визначення характеристик покупця через розпізнавання обличчя може покращити функціональність систем самообслуговування та розширити їхнє застосування в різноманітних сценаріях роботи торгівельних закладів.

Процес розрахунку в торгівельних закладах включає в себе ряд важливих етапів, спрямованих на забезпечення точності та швидкості обслуговування клієнтів, зокрема докладний опис етапів сканування товарів, розрахунку та видачі чека, що визначають операції розрахункового процесу [5].

Сканування товарів визначається важливою фазою розрахунку, що використовується для ідентифікації та реєстрації товарів у системі. Покроковий опис цього етапу виглядає наступним чином:

- Підготовка товарів до сканування: перед початком операції, касир або покупець має розташувати товари таким чином, щоб штрих-коди були легко доступні для сканування.
- Сканування штрих-кодів: касир або сканер автоматично сканує штрих-коди на кожному товарі, вводячи інформацію в касову систему для подальшого обчислення вартості.
- Введення додаткової інформації: у випадку необхідності, касир може вручну ввести додаткові дані, такі як кількість товарів чи ручні знижки, для точного розрахунку.
- Перевірка характеристик покупця: етап, на якому система проводить перевірку характеристик покупця, зокрема його віку, при наявності обмежених товарів. У разі необхідності покупець повинен пройти процедуру верифікації для забезпечення відповідності встановленим вимогам.

Після завершення етапу сканування товарів, настає фаза розрахунку та видачі чека, яка включає наступні кроки:

- Обчислення загальної суми: система автоматично визначає загальну суму до оплати на основі цін товарів і застосованих знижок.
- Вибір методу оплати: клієнт обирає метод оплати, такий як готівка, безготівковий переказ або електронний кошт.
- Проведення оплати: операція оплати здійснюється, а система реєструє факт оплати та оновлює стан рахунку.
- Видача чека: після успішної оплати, система генерує чек, який може бути надрукований або представлений у електронному форматі, де детально вказана інформація про придбані товари та здійснену оплату.

Ці етапи не тільки допомагають ефективно виконати розрахунок, але і сприяють високій точності обліку покупок та їхньої вартості, забезпечуючи задоволення як для клієнтів, так і для персоналу торговельних закладів.

Сучасні технології та впровадження систем самообслуговування в торгівельних закладах принесли безліч переваг, проте разом із цими нововведеннями виникають й певні виклики, особливо щодо безпеки та ідентифікації покупців [6].

Ризик використання банківських карток іншими особами: зі зростанням популярності безготівкових розрахунків та використання банківських карток для самостійного розрахунку, з'являється серйозний ризик використання карток іншими особами без їхнього дозволу. Ситуації, коли картки стають об'єктом крадіжок або втрат, можуть призвести до фінансових труднощів та дискомфорту для осіб, які ними володіють.

Необхідно впроваджувати ефективні механізми аутентифікації та авторизації, які зменшать ймовірність незаконного використання банківських карток. Такі заходи включають в себе двофакторну аутентифікацію, використання біометричних даних або підтвердження операцій через мобільний додаток.

Потреба в підтвердженні особи покупця: однією з ключових проблем є необхідність підтвердження особи покупця, особливо при використанні самообслуговувальних систем. Важливо впевнитися, що особа, яка здійснює оплату, є законним власником використовуваної картки. Це викликає необхідність впровадження механізмів додаткової ідентифікації, таких як пін-коди, розпізнавання обличчя чи інші біометричні методи.

Важливо також забезпечити те, щоб такі процедури не заважали зручності користувачів та не збільшували час обслуговування.

Аналіз необхідності додаткової ідентифікації покупця: проведення аналізу необхідності додаткової ідентифікації покупців – ключовий етап у вирішенні цих викликів. З одного боку, важливо забезпечити високий рівень безпеки та захисту від шахрайства. З іншого боку, необхідно уникати виникнення перешкод та

незручностей для клієнтів, сприяючи швидкому та ефективному процесу розрахунку.

Захист приватності та конфіденційності даних: розгляд питань захисту приватності та конфіденційності даних – ще один важливий аспект вирішення проблем самостійного розрахунку. Збір та обробка особистої інформації повинні відбуватися відповідно до встановлених законодавчих вимог та враховувати інтереси та права клієнтів.

Таким чином, вирішення проблем самостійного розрахунку вимагає комплексного підходу, який балансує між безпекою, зручністю та захистом приватності. Оптимальне вирішення цих питань сприятиме подальшому розвитку та впровадженню новітніх технологій в сфері роздрібної торгівлі.

Впровадження автоматизованих систем розрахунків у торгівельних закладах зумовлює новий етап розвитку взаємодії між покупцем і продавцем. Класифікація способів розрахунку дозволяє детально вивчати різноманітні аспекти платіжних систем та їх вплив на ефективність та комфорт покупок [7].

Простий касовий розрахунок: простота та ефективність розрахунків з використанням касових операцій стають важливими складовими для задоволення потреб сучасного споживача.

Касовий розрахунок з додатковою ідентифікацією за карткою магазину: Використання карт лояльності та програм знижок покращує взаємодію із покупцем та стимулює його вибір продуктів.

Системи самообслуговування: впровадження систем самообслуговування робить процес покупки більш зручним для клієнта та ефективним для магазину.

Інші інноваційні системи розрахунку: розгляд новітніх технологій показує широкий спектр можливостей, які сприяють модернізації та оптимізації торгівельних процесів.

Проблеми самостійного розрахунку та необхідність додаткової ідентифікації покупця: незважаючи на переваги, існують проблеми, такі як ризик використання карток іншими особами, що потребують ретельного аналізу та заходів забезпечення безпеки.

Захист приватності та конфіденційності даних: співіснує між безпекою, зручністю та захистом приватності. Оптимальне вирішення цих питань сприятиме подальшому розвитку та впровадженню новітніх технологій в сфері роздрібної торгівлі.

Висвітлені аспекти показують, що майбутнє торгівлі та розрахунків пов'язане з постійним вдосконаленням технологій, забезпеченням безпеки операцій та збалансованим врахуванням інтересів як покупців, так і продавців. Автоматизація та інновації мають потенціал покращити якість обслуговування та зробити процес покупки більш зручним і доступним для різних категорій споживачів.

1.2 Аналіз стану імплементації програмного забезпечення для системи самостійного касового розрахунку

Каси самообслуговування та програмне забезпечення, що використовується для їх функціонування, вважаються ефективними рішеннями на ринку сучасної роздрібної торгівлі. В Україні спостерігається тенденція активного впровадження новітніх технологій самообслуговування, спрямована на отримання конкурентних переваг, і ця тенденція посилюється щороку. Такий розвиток пов'язаний із успішним використанням подібних систем в мережах роздрібної торгівлі на Заході, що призводить до збільшення попиту на ці технології серед українських роздрібних мереж. Перший етап впровадження кас самообслуговування в Україні відзначився в 2013 році, коли в одній із філій мережі "Велика кишеня" у Києві була встановлена каса самообслуговування. Щодо стратегій впровадження цих технологій на українських ринках, можна виділити два основних варіанти: використання програмного забезпечення світових брендів та створення власних систем. Українські мережі роздрібної торгівлі, такі як "Велика кишеня", "Velmart", "Novus" та "Сільпо", є прикладами успішного впровадження самостійних систем касового розрахунку [8].

Незважаючи на різноманіття послуг на зарубіжних ринках, український роздріб не може швидко втілити тенденції щодо оснащення магазинів необхідним обладнанням та програмним забезпеченням для здійснення самообслуговування покупцями. У нашій країні існує проблема високих витрат на такі рішення і невизначеності стосовно попиту на каси самообслуговування серед покупців. Це призводить до низького рівня імплементації проектів з встановлення кас самообслуговування та відповідного програмного забезпечення.

Наявні на даний момент системи самообслуговування на ринку пропонують широкий вибір потенційних можливостей. Глобальні тенденції надання новітнього програмного забезпечення для магазинів постійно розвиваються в Китайській Народній Республіці, країнах Європи та Сполучених Штатах Америки. У 1992 році в Нью-Йорку було задокументовано перший випадок використання каси самообслуговування. Враховуючи міжнародний досвід впровадження програмного забезпечення для касових апаратів самообслуговування, не варто залишати без уваги пропозицію від StrongPoint. StrongPoint (SP), будучи постачальником технологічних рішень, може значною мірою сприяти успішному впровадженню програмного забезпечення для касових апаратів самообслуговування. StrongPoint спеціалізується на наданні індивідуальних рішень в галузі роздрібної торгівлі. Їхній досвід охоплює апаратне забезпечення, програмне забезпечення та послуги з інтеграції, що забезпечує повне розуміння потреб і вимог до касових апаратів самообслуговування. Завдяки таким можливостям як швидка купівля, сканування та оперативна оплата система StrongPoint стала лідером серед конкурентів у галузі програмного забезпечення. Характерною рисою цієї системи є база даних, яка гарантує сканування кожного товару перед пакуванням. Залежно від своїх уподобань торговці можуть інтегрувати це програмне забезпечення з додатковим обладнанням, таким як SP-технологія самообслуговування [9].

Станом на сьогодні, ринок самостійних систем касового розрахунку в Україні зіткнувся з викликами, пов'язаними з обмеженим фінансовим забезпеченням багатьох міні-маркетів. Необхідність імплементації та

вдосконалення технологій самообслуговування стає важливою для збільшення ефективності роздрібної торгівлі. Суттєвий інтерес до цієї тенденції в Україні викликає підвищений конкурентний тиск та прагнення покращити обслуговування клієнтів. Потрібно активізувати ініціативи щодо сприяння малим і середнім підприємствам (МСП) у впровадженні кас самообслуговування. Окрім вищеокресленого, важливо враховувати регіональні особливості та стимулювати споживачів до активного використання цих технологій через інформаційні кампанії та зручний інтерфейс.

Підхід, що полягає у підтримці МСП в Україні у впровадженні касових апаратів самообслуговування, має кілька важливих переваг для країни. По-перше, це підвищує ефективність роботи цих компаній. Впровадження касових апаратів самообслуговування спрощує процес оплати, скорочуючи час очікування клієнтів. Це підвищення ефективності може залучити більше споживачів і підвищити лояльність клієнтів. Підтримка малих і середніх підприємств (МСП) в Україні у впровадженні касових апаратів самообслуговування має кілька важливих переваг для країни. По-перше, це підвищує ефективність роботи цих компаній. Впровадження касових апаратів самообслуговування спрощує процес оплати, скорочуючи час очікування клієнтів. Таке підвищення ефективності може залучити більше споживачів і підвищити їх лояльність.

Крім того, інтеграція касових апаратів самообслуговування в МСП може призвести до економії коштів. Автоматизовані касові системи вимагають менше персоналу, що дозволяє МСП ефективніше розподіляти ресурси та потенційно скорочувати операційні витрати. Застосування сучасних технологій не тільки покращує бізнес-операції, але й демонструє інновації, забезпечуючи конкурентоспроможність українських МСП на ринку.

Через брак фінансування та неоднозначний попит у більшості міні-маркетів автономні контрольно-касові системи досі не набули широкого поширення в роздрібних мережах України. Дана тенденція також може бути пов’язана з такими факторами як недостатня інформація серед роздрібних продавців про функціональні можливості та переваги автономних касових систем. Без

достатньої інформації щодо переваг, які пропонують ці системи, підприємства можуть вагатися з переходом. Проте, навіть з урахуванням цих обмежень, світовий досвід у розгортанні програмних ініціатив касового самообслуговування стане чудовим прикладом для покращення вирішення цієї проблеми в Україні.

1.3 Порівняльний аналіз аналогів

Враховуючи численні переваги впровадження самообслуговування через програмні продукти, станом на 2023 рік обладнані спеціальним чином каси (в основі яких – десктопна програма) використовуються у більшості розвинених країн світу. Слід окреслити лідерів у галузі розробки програмного забезпечення, які спеціалізуються на касах самообслуговування.

TemaBit – це компанія з розробки програмного забезпечення, відома своїм досвідом у технології блокчейн і створенні рішень для різних галузей. Компанія входить до складу однієї з найбільших комерційних і промислових груп України, а саме – Fozzy Group. Команда компанії розробила понад 75% програмного забезпечення, яким у 2023 році користуються понад 60 тисяч співробітників, численна кількість партнерів і мільйони гостей мережі магазинів Silpo, Фора, Fozzy, Thrash! і служби доставки Justin. TemaBit пропонує налаштовані рішення, адаптовані до конкретних потреб бізнесу. Їхні можливості розробки програмного забезпечення пропонують адаптовані рішення для касових апаратів самообслуговування, які задовільняють різні сфери роздрібної торгівлі та унікальні вимоги бізнесу [10].

Однією з найбільш істотних переваг компанії TemaBit є наявність розробок кас самообслуговування в магазинах лінійки Silpo Shop. Їхній проект отримав назву "Самокаса" (‘Self-checkout’). Цей продукт має приємний дизайн, симпліфікований і зрозумілий клієнтам інтерфейс. Етап взаємодії користувачів з продуктом простий: покупець здійснює скан товару, перекладає його на контрольну платформу та обирає зручний для нього тип оплати. Після цього покупка завершується. Платформи, розроблені TemaBit, включають спеціалізовані

датчики на всіх платформах каси. Ці датчики використовуються для порівняння відсканованих товарів із розміщеними на платформі, забезпечуючи точність процесу оформлення замовлення. Каси самообслуговування обладнані для оплати як банківськими картками, так і готівкою. Вони оснащені купюроприймачами та можуть видавати клієнтам здачу, дозволяючи використовувати різні способи оплати.

Однак ці системи мають певні недоліки. У них відсутні такі функції, як авторизація та реєстрація клієнтів, налаштування мови інтерфейсу та можливість застосовувати знижки для клієнтів. Найпомітнішим є відсутність профілювання клієнтів або функцій ідентифікації. По суті, платформи TemaBit зосереджені на точній перевірці товару та підтримці кількох методів оплати. Тим не менш, їм не вистачає певних орієнтованих на клієнта функцій, таких як авторизація, додаткова верифікація та застосування знижок на основі вподобань клієнта. Приклад каси "Самокаса" показано на рисунку 1.1.

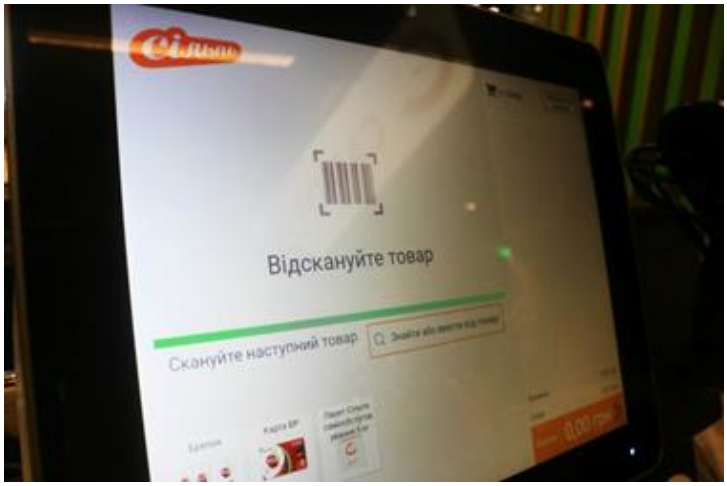

Рисунок 1.1 – Користувацький інтерфейс проекту ‘Self-check’ в мережі магазинів Silpo

Система компанії Erply представляє сучасний підхід до кас самообслуговування в сфері торгівлі. Завдяки програмному забезпеченню Erply, користувачі можуть отримати доступ до інвентарю магазину в режимі реального

часу з будь-якого місця, використовуючи хмарні технології для зберігання даних про магазин, клієнтів та інвентар [11].

Erply, попри зручний інтерфейс і функціональність, має певні обмеження. Зокрема, йому не вистачає можливості для налаштування мови інтерфейсу та профілювання клієнтів, а також додаткової верифікації. Відсутність налаштування мовного інтерфейсу означає, що користувачі можуть не мати змоги перемикати мову системи на свій вибір, що потенційно створює проблеми для тих, для кого мова не є рідною, або для компаній, які працюють у багатомовному середовищі. Крім того, нездатність системи визначати характеристики клієнта або профілювати клієнтів може перешкоджати її гнучкості та персоналізації в обслуговуванні клієнтів, особливо у випадках, коли компанії прагнуть адаптувати свої послуги на основі конкретних атрибутів або переваг клієнтів. Приклад системи Erply зображено на рисунку 1.2.

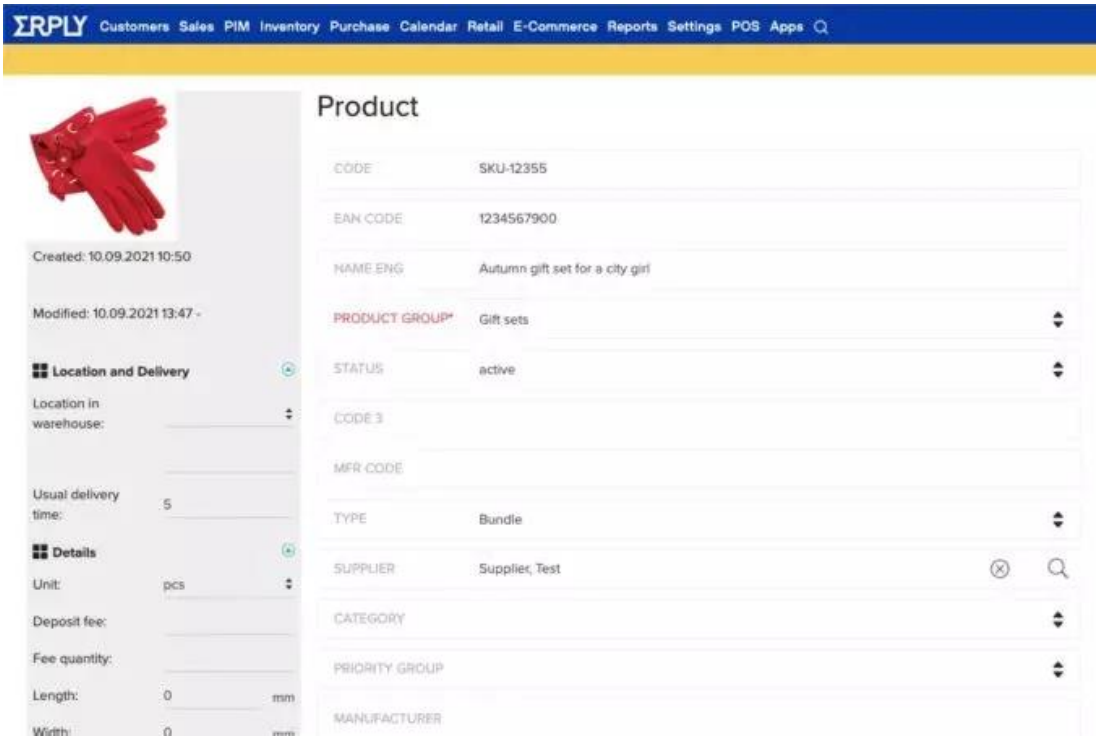

Рисунок 1.2 – Вигляд інтерфейсу Erply зсередини

Компанія SystemGroup розробляє системи самообслуговування, а їхній найбільш успішний проект – це використання їхніх кас в популярних закладах,

таких як мережа фастфуду McDonald's, супермаркети Small & SKIF та торгівельна мережа “Велика Кишеня” [12].

Заклади McDonald's пропонують відвідувачам можливість робити замовлення без касира за допомогою кас самообслуговування. Клієнти можуть використовувати ці каси самообслуговування для розміщення та оплати своїх замовлень за допомогою банківської картки. У касах є інтерактивне меню, де покупці можуть вибрати тип замовлення і спосіб оплати, включаючи можливість оплати готівкою при отриманні замовлення. Крім того, система передає інформацію про нове замовлення персоналу ресторану, який потім негайно починає його готувати. На жаль, незважаючи на великий зручний екран із адаптивним інтерфейсом, ці каси не мають можливостей авторизації та реєстрації користувачів, також додаткової верифікації. Приклад каси McDonald's зображено на рисунку 1.3.

Рисунок 1.3 – Користувацький інтерфейс каси у McDonald's

Узагальнюючи, використання кас самообслуговування різноманітне, і різні програмні рішення відповідають конкретним потребам бізнесу. Хоча кожна система має свої переваги, такі як зручність і доступ у реальному часі до баз даних, вони також мають обмеження, такі як обмеження мовного інтерфейсу та

відсутність деяких функцій, наприклад, реєстрації клієнтів та додаткової верифікації клієнтів. Подальший розвиток технологій кас самообслуговування, ймовірно, вирішить ці обмеження і покращить користувацький досвід у майбутньому.

Після аналізу аналогів було визначено їх сильні та слабкі місця та проведено порівняння з розроблюючою системою самостійного касового розрахунку. Результати порівняння зображено в таблиці 1.2.

Таблиця 1.2 – Порівняльні характеристики програмних продуктів

Критерій	Сільпо	Erply	McDonald's	Власна система
Реєстрація та авторизація користувачів	0	1	0	1
Зміна мови інтерфейсу	0	0	1	1
Алгоритм генерування знижок для клієнтів	0	0	0	1
Визначення характеристик покупця	0	0	0	1

Із зібраної інформації про аналоги програмного забезпечення для касових апаратів самообслуговування можна зробити висновок, що розробка власного програмного продукту є виправданим вибором. Аналіз існуючих аналогів показав, що створення індивідуального програмного рішення дозволить усунути недоліки існуючих аналогів і забезпечить підвищення ефективності та розширення функціональності. Інвестиції в розробку спеціального програмного продукту, спеціально розробленого для касових систем самообслуговування, матимуть переваги перед існуючими аналогами. Цей підхід може потенційно вирішити проблеми, які спостерігаються в поточних варіантах програмного забезпечення, і

призвести до покращення продуктивності та функціональності системи самообслуговування.

1.4 Огляд методів розв’язання поставленої задачі

Слід розглянути існуючу кількість підходів до вирішення проблеми зберігання даних користувачів і продуктів, а також знижок для користувачів. Першим варіантом є використання Oracle, Microsoft Azure, Amazon Web Service або MongoDB Atlas, оскільки вони є гарними прикладами хмарних баз даних. Перелічені хмарні платформи відомі своїм зручним інтерфейсом та простотою використання. Сьогодні існує дві поширені моделі розгортання: або клієнти купують послугу безпосередньо у постачальників хмарних послуг, або вони використовують віртуальні машини для роботи з базою даних безпосередньо в хмарному сховищі. Існує два типи хмарних баз даних: NoSQL і SQL-орієнтовані бази даних [13].

Зберігання конфіденційних даних користувача та інформації про знижки в хмарних базах даних викликає проблеми безпеки. Незважаючи на передові заходи безпеки, які пропонують ці платформи, завжди існує ризик втрати даних або несанкціонованого доступу через кіберзагрози.

Дотримання правил конфіденційності даних, особливо в різних регіонах або країнах, може спричинити проблеми. Управління даними користувачів і знижками з одночасним дотриманням різних законів про захист даних (як-от GDPR, CCPA) вимагає суворих заходів відповідності, що може бути складним у хмарному середовищі.

Покладатися на сервіси хмарних послуг означає, що підприємства залежать від їхньої інфраструктури та надійності. Збої в роботі цих сервісів можуть вплинути на доступ до важливих даних користувача та продукту, порушивши роботу.

Другий спосіб збереження інформації – це створення бази даних власноруч за допомогою фреймворку .Net, технологія Entity Framework і мови

програмування C#. Такий підхід є менш затратним та гнучкішим коли мова йде про розробку бази даних. З огляду на даний фактор, саме це рішення було обрано для системи самостійного касового розрахунку.

Було розроблено алгоритм розрахунку знижок за видами та кількістю товарів, що купуються покупцями. Алгоритм надає знижки на три найпопулярніші товари, які купує споживач. Він враховує активність кожного зареєстрованого клієнта та генерує знижки на основі товарів і кількості, придбаних протягом тижня. Ці знижки оновлюються на щотижневій основі і можуть бути використані на будь-який інший товар, заохочуючи збільшення залучення клієнтів і продажів. Цей алгоритм знижок є стратегічним підходом до стимулювання покупок клієнтів і збільшення продажів. Винагороджуючи лояльність клієнтів і заохочуючи повторні покупки за допомогою персоналізованих знижок, підприємства можуть потенційно збільшити утримання клієнтів і обсяг продажів. Динамічний характер знижок, які оновлюються щотижня та застосовуються до різних товарів, здається ефективною стратегією постійного залучення клієнтів. Однак важливо забезпечити справедливість і прозорість алгоритму розрахунку знижки, щоб зберегти довіру клієнтів.

Вибір веб-додатку для касових апаратів самообслуговування є вигідним рішенням. Веб-додатки пропонують гнучкість у доступності та сумісності з різними пристроями та браузерами, що робить їх зручними для користувачів. Вони часто вимагають менше ресурсів на стороні системи, що може бути вигідним з точки зору продуктивності та ефективності. Однак, хоча веб-додатки пропонують ряд переваг, вони також можуть мати обмеження, такі як залежність від підключення до Інтернету та потенційні проблеми безпеки. Покладення на стабільне підключення до Інтернету може створити проблеми в районах із поганим підключенням. Крім того, надзвичайно важливо забезпечити надійні заходи безпеки для захисту конфіденційних даних транзакцій, що передаються через Інтернет.

Для розробки алгоритму визначення характеристик покупця в системі самостійного касового розрахунку були використані бібліотеки OpenCV. Цей

алгоритм дозволяє системі автоматично визначати стать та вік покупця за допомогою вбудованої камери, що відкриває можливість для продажу обмежених товарів відповідно до встановлених параметрів. Однією з ключових переваг цього підходу є вбудована автоматизація цього процесу, що робить його ефективним і зручним для клієнтів.

У порівнянні з аналогами, де відсутня така можливість, зазвичай потрібно залучати додатковий персонал, що відповідає за визначення віку покупців. У нашому випадку відсутність необхідності в пошуку, наймі та обслуговуванні такого персоналу стає значущою перевагою, оскільки система самостійного касового розрахунку автоматизована та оптимізована. Це сприяє підвищенню ефективності обслуговування та забезпеченню комфортного користувацького досвіду, що може відзначати наш продукт на ринку автоматизованих систем розрахунку.

Усі операції та взаємодії покупця з касою відбуваються при відпраці запитів до бази даних. Таким чином, для програмного продукту були використані технології Web API та протоколи HTTPS. HTTP є прикладним протоколом для розподілених гіпермедійних інформаційних систем та широко використовується у клієнт-серверних взаємодіях [14].

Протокол передачі гіпертексту включає два учасники: клієнта та сервер. Клієнт ініціює з’єднання, надсилає через нього запити, приймає і обробляє відповіді. Сервер очікує на з'єднання, приймає та обробляє запити, а потім надсилає відповіді клієнту.

Програмний продукт використовував графічну структуру Angular для вирішення графічних аспектів, включаючи візуальний інтерфейс і необхідні функції, для веб-додатку. Angular – це популярний фреймворк і водночас платформа з відкритим вихідним кодом, розроблена Google для створення односторінкових програм за допомогою TypeScript. Він пропонує численні переваги та стандартизовану структуру для розробки, що дозволяє створювати великомасштабні додатки у зручний для обслуговування спосіб. Однією з ключових особливостей Angular є використання мови програмування TypeScript.

Використовуючи Angular, можна розробити графічний інтерфейс веб-додатку та реалізувти функції реєстрації клієнтів, авторизації та HTTP-протоколів для зв’язку з базою даних.

Отже, для зберігання інформації про користувачів, товари та нарахування знижок розглянуті два підходи: використання хмарних баз даних (Amazon Web Service, Microsoft Azure, MongoDB Atlas) і створення власної бази за допомогою .Net, Entity Framework, C# для системи самостійного касового розрахунку. Хмарні бази зручні, але мають обмеження, такі як залежність від провайдера. Власна база надає гнучкість і економію, але вимагає більше ресурсів для розробки. Система нарахування знижок використовує веб-додаток, що забезпечує зручний доступ та ефективне тестування. Особливість – використання бібліотек OpenCV для визначення характеристик покупця, що відмінно від аналогів і підкреслює автоматизований характер системи, що може бути конкурентною перевагою.

1.5 Постановка задач розробки системи самостійного касового розрахунку

З метою успішної реалізації будь-якого проєкту важливим етапом є поставка задач. Основні завдання повинні зосереджуватися на створенні зручного для користувача інтерфейсу, забезпеченні надійних заходів безпеки, оптимізації платіжних процедур і забезпеченні бездоганної інтеграції з існуючою інфраструктурою. Окреслити завдання можна наступним чином:

- Провести оцінку методологій впровадження та розробки програмного забезпечення, які використовуються при створенні додатків для систем самообслуговування;
- Розробити архітектуру програмного додатку;
- Створити базу даних, яка буде містити інформацію про товари в магазині, акаунти клієнтів із можливістю збереження історії замовлень та даними споживачів;
- Розробити алгоритм визначення характеристик покупця;

- Створити алгоритм, який визначає критерії знижки на основі таких факторів, як типи та кількість товарів, придбаних протягом певного періоду, та забезпечити їх збереження в базі даних;
- Створити графічний інтерфейс для веб-додатку;
- Розробити систему самостійного касового розрахунку, використовуючи створену архітектуру та алгоритми;
- Ретельно протестувати систему з метою перевірки його ефективності та стабільності;
- Створити документацію для користувачів.

1.6 Висновки

Цей розділ висвітлює різноманітний спектр доступних автоматизованих систем розрахунку, виділяючи їхні функції та варіації. Детальний аналіз поточного стану впровадження програмного забезпечення розкриває як сильні сторони, так і області, які потребують вдосконалення в поточному процесі розробки. Крім того, порівняльний аналіз аналогічних систем дає розуміння про переваги та недоліки схожих систем. Дослідження методологій вирішення підкреслює багатогранність доступних підходів, вносячи цінні перспективи у вирішення визначених проблем. Нарешті, окреслення ключових завдань розробки містить ключові кроки, необхідні для успішного створення та вдосконалення касових систем самообслуговування.

2 РОЗРОБКА МЕТОДУ ВИЗНАЧЕННЯ ВІКУ ТА СТАТІ ПОКУПЦЯ

2.1 Варіантний аналіз існуючих методів визначення віку та статі людини

У даному підрозділі ретельно розглянуто та проаналізовано різні методи визначення віку та статі людини, що використовуються в сучасних технічних системах. Відмічено ключові технічні аспекти та порівняно їх переваги та недоліки для визначення оптимального підходу для системи самостійного касового розрахунку:

- Методи аналізу обличчя: системи визначення віку на основі аналізу обличчя найчастіше використовують глибокі нейронні мережі, зокрема Convolutional Neural Networks (CNN). Ці мережі можуть ефективно визначати особливості, що змінюються з віком, такі як зміна контуру обличчя, з’явлення зморшок тощо. Однією з переваг цього методу є висока точність визначення, особливо при наявності великої кількості тренувальних зображень. Завдяки глибокому навчанню, система може автоматично вивчати та адаптуватися до нових паттернів, що робить її ефективною в різних сценаріях використання. Однак слід відзначити, що для успішної роботи цього методу вимагається висока якість вхідних зображень. У ситуаціях з поганою освітленістю або низькою роздільною здатністю камери, точність може суттєво зменшитися [15].
- Аналіз голосу: інший підхід до визначення віку та статі пов'язаний із аналізом голосу. Цей метод використовує акустичні ознаки, такі як тембр, частота та інтонація голосу, для визначення особистих характеристик. Однією з переваг цього методу є те, що він може працювати в ситуаціях, де аналіз обличчя недоступний або неефективний. Наприклад, при використанні системи у затемнених областях або в місцях з обмеженим доступом до камер. Зауважимо, що аналіз голосу може бути менш точним у шумних середовищах або в разі наявності акцентів, що ускладнює його застосування в певних умовах.

- Використання біометричних даних: деякі системи визначення віку та статі використовують біометричні дані, такі як відбитки пальців чи сканування райдужки ока. Це дозволяє отримати дуже високу точність завдяки унікальності біометричних характеристик. Однак використання біометричних даних має свої обмеження. Це вимагає спеціального обладнання для збору біометричних зразків, що може бути витратним та складним для впровадження в ряд різних середовищ [16].

Таблиця 2.2 зображає порівняльний аналіз переваг та недоліків використання різних методів визначення віку та статі. Це допомагає краще зрозуміти, який метод може бути більш придатним для впровадження у системі самостійного касового розрахунку.

Таблиця 2.2: Переваги та недоліки методів визначення віку та статі.

Метод	Переваги	Недоліки
Аналіз обличчя	Висока точність, широкий функціонал	Залежність від якості зображення
Аналіз голосу	Не вимагає видимого контакту, може бути ефективним у деяких ситуаціях	Може бути менш точним у шумних середовищах
Біометричні дані	Висока точність за наявності спеціального обладнання	Витратний процес, потребує додаткового обладнання

Таблиця 2.2 надає об'єктивний огляд переваг та недоліків трьох ключових методів: аналізу обличчя, аналізу голосу та використання біометричних даних. З огляду на конкретні вимоги нашого проекту, можна визначити, що аналіз обличчя є найбільш підходящим методом.

Отже, після аналізу різних методів визначення віку та статі людини, виявлено, що аналіз обличчя є найбільш перспективним для нашого проекту системи самостійного касового розрахунку. Цей метод надає високу точність та широкий функціонал. Однак, важливо враховувати основне обмеження цього

методу, яким є залежність від якості вхідних зображень. Рішенням цієї проблеми може бути встановлення достатнього освітлення в об'єктах роздрібної торгівлі. Відповідне освітлення може значно покращити якість зображень, що використовуються для аналізу обличчя, забезпечуючи високу точність та надійність визначення віку та статі покупців.

Загалом, аналіз обличчя залишається ефективним методом, а вдосконалення якості вхідних зображень шляхом правильного освітлення сприятиме успішному впровадженню цього підходу у систему самостійного касового розрахунку.

2.2 Огляд технічних систем, що використовують моделі штучного інтелекту для обробки зображень

В останні роки спостерігається стрімкий прогрес у розробці технічних систем, які використовують штучний інтелект для обробки зображень. Цей підрозділ присвячений огляду передових технічних рішень та систем, зокрема тих, які застосовуються для визначення характеристик особи, таких як вік та стать.

Великі компанії, такі як Apple та Microsoft, впроваджують системи розпізнавання обличчя, такі як FaceID та Azure Face API відповідно. Вони застосовують глибокі нейронні мережі, зокрема Convolutional Neural Networks (CNN), для визначення віку та статі. Ці системи надійно працюють в умовах різноманітних освітлених умов та різних ракурсів, забезпечуючи високий рівень точності.

Системи розпізнавання обличчя та визначення віку та статі також отримують широке застосування в роздрібній торгівлі. Магазини використовують ці технології для аналізу даних про покупців, забезпечуючи персоналізовані послуги та рекомендації. Це стає ефективним інструментом для підвищення конкурентоспроможності та збільшення задоволеності клієнтів.

Однією з ключових тенденцій у галузі є розширення можливостей систем визначення характеристик покупців. Науковці працюють над врахуванням не

лише базових ознак, але й індивідуальних особливостей та контексту. Використання біометричних даних та глибокого навчання в цьому контексті відкриває шлях для створення більш гнучких та ефективних систем.

Важливо відзначити, що системи, такі як FaceID від Apple, використовують не тільки основні функції розпізнавання, але і додаткові, такі як виявлення емоцій та визначення вікових категорій. Це робить їх універсальними для різних завдань та сценаріїв використання [17].

У таблиці 2.1 описано сучасні технічні системи з використанням штучного інтелекту для визначення характеристик особи на основі обробки зображень.

Таблиця 2.1 – сучасні технічні системи з використанням штучного інтелекту.

Назва системи	Виробник	Основні функції
FaceID	Apple	Розпізнавання обличчя, визначення віку та статі
Azure Face API	Microsoft	Розпізнавання обличчя, визначення віку та статі
Amazon Rekognition	Amazon	Виявлення емоцій та визначення вікових категорій

FaceID, розроблений компанією Apple, є передовою системою розпізнавання обличчя, яка використовується для визначення віку та статі користувачів. Заснований на технології глибокого навчання, FaceID забезпечує надзвичайно точне розпізнавання обличчя в різних умовах освітлення та під різними кутами. Однією з основних функцій є визначення віку особи, що робить його ефективним інструментом для автентифікації та персоналізації користувачів.

Microsoft Azure Face API – це потужний інструмент для розпізнавання обличчя та визначення віку та статі. Використовуючи технології машинного

навчання, він надає високу точність розпізнавання обличчя в реальному часі. Функція визначення віку та статі робить його відмінним вибором для застосувань, де потрібно аналізувати характеристики користувачів.

Amazon Rekognition, розроблений Amazon, виходить за межі простого розпізнавання обличчя, дозволяючи виявлення емоцій та визначення вікових категорій. Ця система базується на сучасних алгоритмах комп'ютерного зору та машинного навчання. Виявлення емоцій дозволяє отримати більше інформації про стан користувача, а визначення вікових категорій робить її важливим інструментом для аналізу аудиторії та вироблення маркетингових стратегій [18].

На рисунку 2.1 представлена архітектура системи розпізнавання обличчя Face ID від Apple.

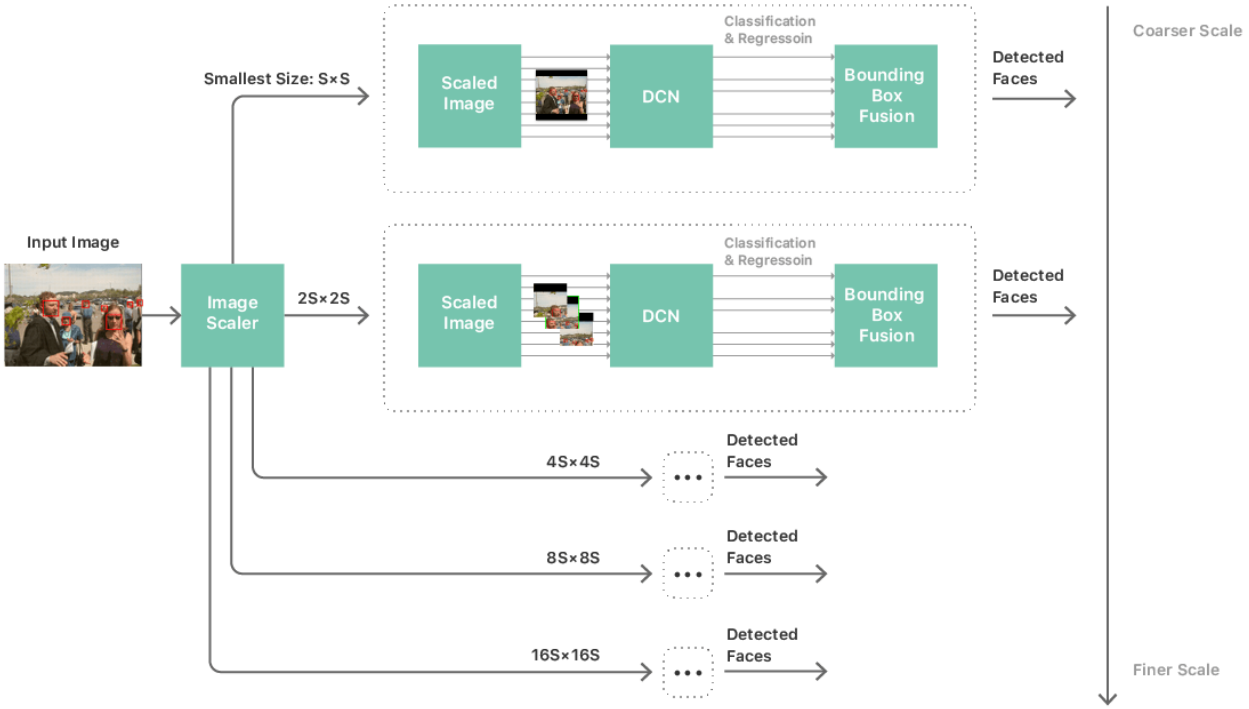

Рисунок 2.1 – Архітектура системи розпізнавання обличчя FaceID

Архітектура системи розпізнавання обличчя FaceID від Apple вражає своєю складністю та високою ефективністю. Основні етапи роботи системи можна розділити на кілька ключових компонентів:

- Збір даних: система FaceID починає свою роботу з великої кількості даних про обличчя користувачів. Ці дані включають в себе різні кути та варіанти освітлення для навчання системи розпізнавання.
- Модель глибокого навчання: ядро системи – це модель глибокого навчання, яка використовується для ефективного вивчення унікальних особливостей обличчя. Зазвичай використовуються Convolutional Neural Networks (CNN), які дозволяють автоматично визначати різні аспекти обличчя, такі як форма, розташування очей, ніздрі та рота.
- Етап розпізнавання: після навчання модель використовується для розпізнавання обличчя в реальному часі. Вона аналізує подане зображення і визначає унікальні особливості, які дозволяють ідентифікувати конкретного користувача.
- Захист від шахрайства: FaceID включає в себе додаткові заходи безпеки, такі як виявлення тривіальних масок чи спроб обхідної реєстрації за допомогою фотографій. Це забезпечує надійний рівень безпеки.
- Інтеграція з іншими функціями: FaceID інтегрований з іншими функціями пристрою, такими як розблокування екрану, авторизація платежів та застосунків, а також персоналізація деяких параметрів пристрою.

Отже, було визначено важливі аспекти, які слід врахувати при подальшій розробці методів та алгоритмів визначення характеристик покупця для системи самостійного касового розрахунку.

2.3 Варіантний аналіз існуючих програмних засобів для визначення віку та статі людини

В даному підрозділі було ретельно розглянуто і порівняно існуючі програмні засоби для визначення віку та статі людини. Кожен із них має свої унікальні характеристики та можливості, які можуть бути інтегровані у систему самостійного касового розрахунку.

Amazon Rekognition володіє рядом функцій, серед яких ключовими для нас є виявлення емоцій та визначення вікових категорій. Система дозволяє аналізувати вираз обличчя та виділяти ключові емоції, що може бути корисним для аналізу реакцій на продукти та послуги. Особливості визначення віку дозволяють робити приблизні оцінки, що важливо для аналізу цільової аудиторії та персоналізації обслуговування [19]. Приклад роботи Amazon Rekognition показано на рисунку 2.2.

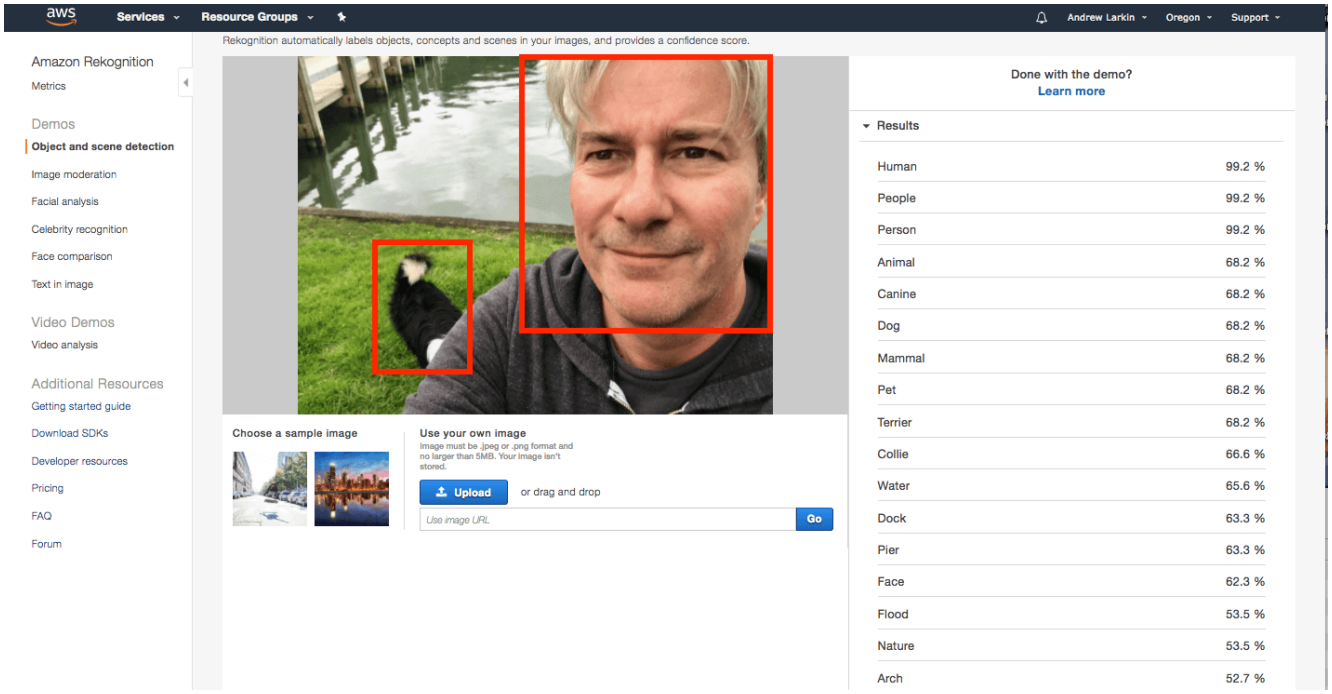

Рисунок 2.2 – Приклад роботи Amazon Rekognition

Microsoft Azure Face API є рішенням, яке комбінує розпізнавання обличчя, визначення віку та статі. Завдяки використанню передових технологій розпізнавання, це API може точно ідентифікувати особу та визначати її основні характеристики. Для системи самостійного касового розрахунку важливою є можливість реального часу, а Azure Face API може забезпечити швидке та ефективне розпізнавання. Приклад роботи Microsoft Azure Face API показано на рисунку 2.3.

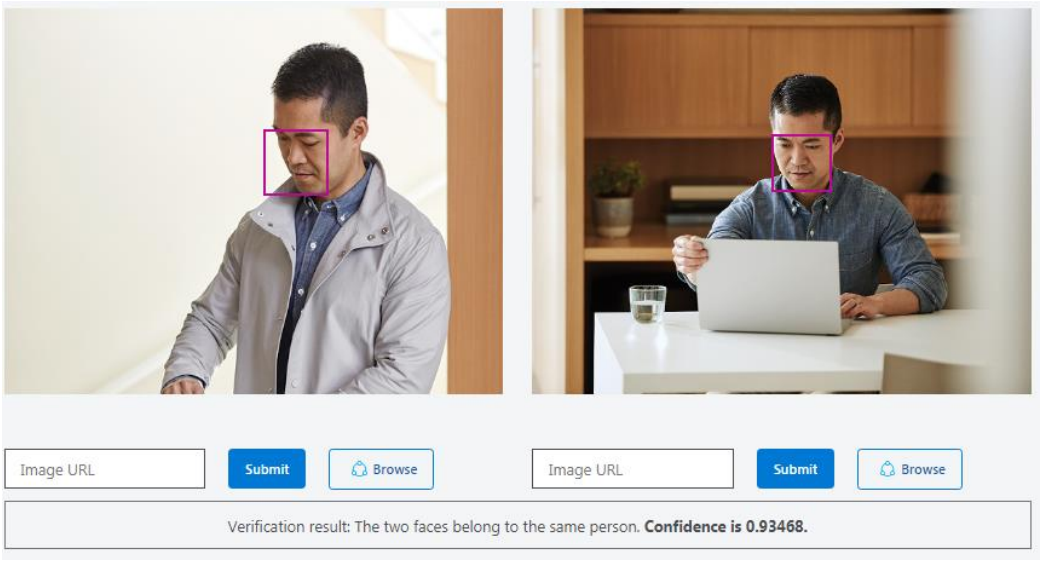

Рисунок 2.3 – Приклад роботи Microsoft Azure Face API

FaceID від Apple є однією з найбільш визнаних систем розпізнавання обличчя. Вона використовується для розблокування пристроїв та авторизації платежів. Специфічність даної технології полягає в її високій точності та можливості врахування різних аспектів обличчя, включаючи вік та стать. Це робить FaceID потенційно важливим компонентом для системи самостійного касового розрахунку [20]. Приклад роботи FaceID від Apple показано на рисунку 2.4.

Рисунок 2.4 – Приклад роботи FaceID від Apple

Таблиця 2.3 – Порівняльний аналіз систем розпізнавання обличчя

Характеристика	Amazon Rekognition	Microsoft Azure Face API	FaceID від Apple
Розпізнавання обличчя	Так	Так	Так
Визначення віку	Так (приблизно)	Так	Так
Визначення статі	Зазначення вікових категорій	Так	Так
Точність	Залежить від умов	Баланс точності та швидкості	Висока точність
Використання в реальному часі	Так	Так	Так
Інтеграційні можливості	Широкі	Широкі	Зорієнтовано на пристрої Apple

Оглянувши характеристики систем розпізнавання обличчя, можна визначити їхню високу ефективність та широкий спектр функцій. Однак в контексті української роздрібної торгівлі, імплементація таких систем може стати викликом. Висока вартість впровадження та підтримки таких технологій може бути недосяжною для більшості закладів роздрібної торгівлі в Україні. Крім того, можливі юридичні проблеми, пов'язані з захистом персональних даних та приватністю, можуть ускладнити впровадження таких систем на практиці. Таким чином, необхідність враховувати не лише технічні аспекти, а й економічні та правові відносини при розгляді можливостей визначення характеристик покупця для системи самостійного касового розрахунку в українських умовах змушує розробити власну систему розпізнавання характеристик покупця.

2.4 Розробка методів та алгоритмів визначення характеристик покупця

У цьому підрозділі будуть представлені методи та алгоритми для визначення віку та статі покупців у системі самостійного касового розрахунку, ілюстровані на різних мовах програмування.

Для точного визначення віку та статі використаємо глибоке навчання та нейронні мережі. На мові програмування Python це зображено на рисунку 2.5.

main.py

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential()
# Додавання шарів до моделі...
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=10, validation_data=(val_images, val_labels))
```

Рисунок 2.5 – Приклад використання глибокого навчання на Python

У JavaScript можна використовувати TensorFlow.js. TensorFlow.js – це варіант фреймворку машинного навчання TensorFlow, спеціально адаптований для використання у веб-середовищі та на мові програмування JavaScript. Він надає зручний спосіб використовувати моделі машинного навчання безпосередньо в браузері або на сервері з використанням Node.js. Приклад використання зображено на рисунку 2.6.

JS

```javascript
const model = tf.sequential();
// Додавання шарів до моделі...
model.compile({ optimizer: 'adam', loss:
'binaryCrossentropy', metrics: ['accuracy'] });
model.fit(trainImages, trainLabels, epochs: 10,
validationData: [valImages, valLabels]);
```

Рисунок 2.6 – Приклад використання глибокого навчання на JavaScript

Так виглядає приклад використання глибокого навчання та нейронних мереж для точного визначення віку та статі покупців. Модель створюється та навчається на зображеннях, використовуючи бібліотеку TensorFlow [21]. Код на Python та JavaScript.

Алгоритми обробки зображень мають ключове значення у визначенні характеристик покупців. На Python можна використовувати OpenCV. Приклад зображено на рисунку 2.7.

main.py

```python
import cv2

def detectFaceContours(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)
    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

    return image
```

Рисунок 2.7 – Приклад визначення віку на Python з використанням OpenCV

Інтеграція з системою касового розрахунку виглядатиме по-різному залежно від мови програмування. На прикладі Python показано на рисунку 2.8.

main.py

```python
def processCustomer(image):
    age, gender = model.predict(image)

    processedImage = detectFaceContours(image)

    # Інтеграція з системою касового розрахунку
    # (додавання отриманих характеристик до чека чи бази даних)

    return processedImage, age, gender
```

Рисунок 2.8 – Приклад інтегрування на мові Python

Оптимізація та навчання на реальних даних – це ключовий етап в процесі розробки алгоритмів та моделей машинного навчання. Цей підпункт в контексті

розділу 2.4 стосується важливості використання реальних даних для підготовки та тренування алгоритмів, а також оптимізації їхньої продуктивності.

Оптимізація в даному контексті означає створення алгоритмів та моделей, які працюють ефективно та швидко. Це важливо для системи самостійного касового розрахунку, оскільки ефективні алгоритми дозволяють обробляти дані в реальному часі, забезпечуючи швидку реакцію на дії покупців [22]. Оптимізація також допомагає знижувати навантаження на апаратне забезпечення, забезпечуючи ефективну роботу системи при різних обсягах даних.

Використання реальних даних для тренування моделей машинного навчання є критично важливим етапом. Реальні дані відображають різноманітність сценаріїв та умов, з якими може стикатися система в реальному світі. Вони включають в себе реальні зображення обличчя покупців, реальні транзакції та звіти про продажі.

Основні аспекти оптимізації та навчання на реальних даних включають:

- Попередня обробка даних: забезпечення якості та структури даних перед подачею їх на вхід моделі.
- Вибір оптимальних алгоритмів: вибір та налаштування алгоритмів машинного навчання, які найкраще підходять для конкретного завдання.
- Крос-валідація: оцінка ефективності моделі на основі різних підвибірок даних для уникнення перенавчання.
- Оптимізація гіперпараметрів: налаштування параметрів моделі для досягнення оптимальної продуктивності.
- Забезпечення реалістичності даних: використання реальних даних, що відображають різноманітність та специфіку умов використання системи.
- Моніторинг та оновлення: постійний моніторинг ефективності моделі та оновлення її відповідно до змін у вхідних даних або умовах використання.

Оптимізація та навчання на реальних даних допомагають створювати систему самостійного касового розрахунку, яка ефективно працює в реальних умовах роздрібної торгівлі.

2.5 Розробка прототипу підсистеми визнання віку та статі людини

У цьому підрозділі буде розглянуто застосування бібліотеки OpenCV для визначення віку та статі покупців з відеопотоку камери в системі самостійного касового розрахунку. OpenCV – це потужна бібліотека для обробки зображень та комп'ютерного зору, яка надає інструменти для вирішення різноманітних завдань, включаючи розпізнавання обличчя.

OpenCV надає інтерфейс для роботи зі зображеннями та відео, включаючи можливості аналізу обличчя. Для визначення віку та статі можна використовувати попередньо навчені моделі, але також можна створювати власні моделі, засновані на даних покупців у конктретній точці роздрібної торгівлі [23].

Основні етапи визначення віку та статі з відеопотоку камери в OpenCV включають:

- Захоплення відеопотоку: Використання OpenCV для отримання відеопотоку з камери.
- Виявлення обличчя: Використання методів виявлення обличчя для локалізації обличчя покупців на кадрах відеопотоку.
- Визначення віку та статі: Застосування моделі для визначення параметрів, таких як вік та стать, на обличчя покупців.
- Виведення результатів: Подання інформації про вік та стать на екрані або передача її для подальшого аналізу системою самостійного касового розрахунку.

Переваги застосування OpenCV

- Легкість інтеграції: OpenCV має зручний інтерфейс, що полегшує інтеграцію з існуючою системою самостійного касового розрахунку.
- Висока швидкодія: Бібліотека оптимізована для ефективної обробки зображень в реальному часі.
- Гнучкість: Можливість використання попередньо навчених моделей чи створення власних, що враховують специфіку покупців у конкретному магазині [24].

Далі буде переглянуто код який реалізує визначення обличчя, віку та статі з відеопотоку камери за допомогою OpenCV та мережі нейронних мереж, навченої для аналізу обличчя. Буде розглянуто основні елементи коду та проаналізовано їхню роботу.

В наступному фрагменті коду завантажуються попередньо навчені моделі для визначення віку та статі, а також для виявлення обличчя. На рисунку 2.9 зображено завантаження моделей.

```
const string faceProto = "models/deploy.prototxt";
const string faceModel = "models/res10_300x300_ssd_iter_140000_fp16.caffemodel";
const string ageProto = @"models/age_deploy.prototxt";
const string ageModel = @"models/age_net.caffemodel";
const string genderProto = @"models/gender_deploy.prototxt";
const string genderModel = @"models/gender_net.caffemodel";
_ageNet = CvDnn.ReadNetFromCaffe(ageProto, ageModel);
_genderNet = CvDnn.ReadNetFromCaffe(genderProto, genderModel);
_faceNet = CvDnn.ReadNetFromCaffe(faceProto, faceModel);
```

Рисунок 2.9 – Завантаження моделей

Далі налаштовується захоплення відеопотоку з камери та запускається окремий потік для обробки відеопотоку. На рисунку 2.10 зображено аналіз обличчя та параметрів.

```
Ссылок: 1
private void DetectFaces(Mat newImage, Mat imageRes)
{
    ...
    for (int i = 0; i < detectionMat.Rows; i++)
    {
        float confidence = detectionMat.At<float>(i, 2);

        if (confidence > 0.7)
        {
            int x1 = (int)(detectionMat.At<float>(i, 3) * frameWidth);
            int y1 = (int)(detectionMat.At<float>(i, 4) * frameHeight);
            int x2 = (int)(detectionMat.At<float>(i, 5) * frameWidth);
            int y2 = (int)(detectionMat.At<float>(i, 6) * frameHeight);

            Cv2.Rectangle(newImage, new Point(x1, y1), new Point(x2, y2), Scalar.Green, LineThickness);

            if (_doAgeGender)
                AnalyzeAgeAndGender(x1, y1, x2, y2, imageRes, newImage);
        }
    }
}
```

Рисунок 2.10 – Аналіз обличчя та параметрів

Після цього наступний фрагмент коду виявляє обличчя на кадрі та, якщо параметр _doAgeGender встановлено в true, аналізує вік та стать. Приклад аналізу віку та статі показано на рисунку 2.11.

```
Ссылок: 1
private void AnalyzeAgeAndGender(int x1, int y1, int x2, int y2, Mat imageRes, Mat newImage)
{
    ...
    GetMaxClass(genderPreds, out int classId, out double classProbGender);
    var gender = _genderList[classId];

    _ageNet.SetInput(blobGender);
    var agePreds = _ageNet.Forward();
    GetMaxClass(agePreds, out int classIdAge, out double classProbAge);
    var ageIndex = classIdAge;

    Scalar textColor = (_ageValues[ageIndex] >= 18) ? Scalar.Green : Scalar.Red;

    var ageValue = _ageValues[ageIndex];
    var ageRange = $"{ageValue} ({ageValue - 1}-{ageValue + 1})";
    var label = $"{gender}, {ageRange}";
    Cv2.PutText(newImage, label, new Point(x1 - 10, y2 + 20), HersheyFonts.HersheyComplexSmall, 1, textColor, 1);
}
```

Рисунок 2.11 – Аналіз віку та статі

Отже, у даному підрозділі проведено розгляд методів визначення віку та статі особи на основі відеопотоку за допомогою OpenCV та моделей глибокого навчання. Реалізація демонструє важливі можливості OpenCV для роботи зі зображеннями та відео. Основні аспекти реалізації включають виявлення обличчя за допомогою нейромережі, аналіз особливостей обличчя та визначення віку та статі. Використання моделей для визначення віку та статі покупців може бути корисним у роздрібній торгівлі для персоналізації обслуговування та аналізу цільової аудиторії. Зазначається, що точність визначення може залежати від якості вихідного зображення та може варіюватися для різних груп віку. Крім того, встановлення достатнього освітлення у місцях роздрібної торгівлі може покращити якість зображень і, отже, результати визначення. У цілому, реалізація вказує на потужність OpenCV як інструменту для аналізу відеопотоку та використання його у поєднанні з моделями глибокого навчання для вирішення конкретних завдань розпізнавання особи та визначення віку та статі.

2.6 Оцінка якості роботи розробленого методу

У цьому підрозділі буде проведено оцінку розробленої системи розпізнавання характеристик покупця на прикладі чоловіка повнолітнього та дитини неповнолітньої. При цьому вік та стать повнолітньої людини будуть відображатися зеленим кольором, вказуючи, що перевірка пройшла успішно. У випадку неповнолітньої дитини вік та стать будуть виділені червоним кольором, вказуючи на невдалу перевірку. Для цього використовується фотографія чоловіка та дитини, отримана за допомогою веб-камери. Після обробки зображень системою розпізнавання виведені результати на екран. Приклад роботи розпізнавання зображено на рисунках 2.12 - 2.13.

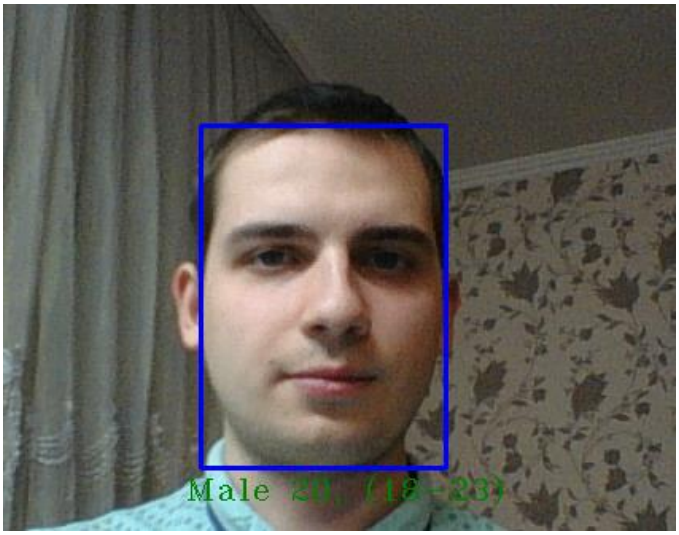

Рисунок 2.12 – Приклад розпізнавання повнолітньої людини

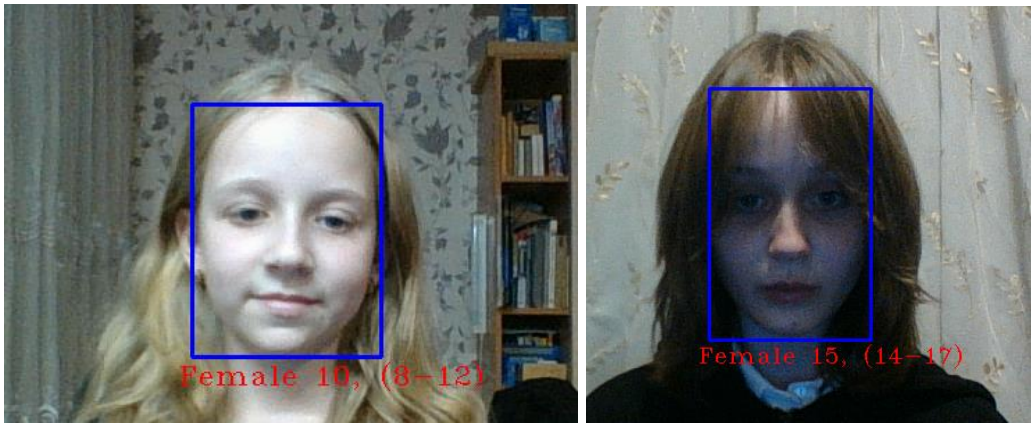

Рисунок 2.13 – Приклад розпізнавання неповнолітньої людини

Результати експерименту ілюструють ефективність системи у визначенні характеристик покупців. Зелений колір для повнолітньої людини підтверджує успішне визначення віку та статі, тоді як червоний колір для дитини свідчить про невдалу перевірку.

Даний підхід впроваджується для покращення безпеки та контролю при користуванні системою самостійного касового розрахунку. Розроблена система надає можливість оперативно та точно визначати основні характеристики покупців, сприяючи оптимізації процесу обслуговування в роздрібних торгових точках.

Також було проведено оцінку розробленої системи розпізнавання характеристик покупця, зокрема визначення віку та статі, використовуючи критерії якості, визначені за допомогою метрик класифікації [25].

Критерії оцінки:

- Accuracy (Точність): Однією з основних метрик є загальна точність моделі, яка обчислюється як відношення суми правильно класифікованих прикладів (True Positives та True Negatives) до загальної кількості прикладів. У нашому випадку, точність моделі складає приблизно 88.46%.
- Precision (Точність): Точність визначає відсоток правильно класифікованих позитивних прикладів (True Positives) серед усіх прикладів, що були класифіковані як позитивні. У нашому випадку, точність становить близько 88.89%.
- Recall (Повернення): Повернення визначає відсоток правильно класифікованих позитивних прикладів (True Positives) серед усіх фактичних позитивних прикладів. У нашому випадку, повернення складає приблизно 80%.
- F1 Score: Сполучає точність і повернення в одну метрику. Обчислюється за формулою F1=2*Precision*Recall/Precision+Recall. У нашому випадку, F1 Score дорівнює приблизно 84.21%, що свідчить про збалансовану ефективність моделі.

- ROC-AUC та Крива ROC: Для визначення здатності моделі визначати позитивні та негативні класи при зміні порогу вирішення використовувалася ROC-AUC (Area Under the Curve) та Крива ROC. Чим більше площа під кривою, тим краще відповідає модель.
- Крива PR та Precision-Recall: Також використовувалася Крива PR (Precision-Recall), яка дозволяє оцінити залежність між точністю та поверненням при різних порогах.

Оцінка системи розпізнавання характеристик покупця підтвердила високу точність та ефективність розробленого методу визначення віку та статі. Застосування метрик класифікації, таких як Accuracy, Precision, Recall та F1 Score, дозволило здійснити докладний аналіз результатів. Отримані результати були перевірені та випробувані на датасеті "Age, Gender, and Ethnicity Face Data", що підтверджує ефективність моделі на реальних даних [26].

Отримані високі показники Accuracy та узгодженість між Precision та Recall свідчать про те, що система ефективно впоралася з визначенням характеристик покупців. Це створює перспективи для успішного впровадження розробленої системи в об'єкти роздрібної торгівлі, підвищуючи рівень автоматизації та комфорту для клієнтів. При цьому важливо продовжувати вдосконалювати алгоритми та розглядати можливості розширення функціоналу для більш широкого застосування.

2.7 Висновки

Результати проведених досліджень та розробки системи самостійного касового розрахунку з визначенням характеристик покупця свідчать про успішність роботи системи та її потенційну ефективність у сфері роздрібної торгівлі. Застосування технологій штучного інтелекту для розпізнавання віку та статі дозволяє покращити якість обслуговування та персоналізацію взаємодії з клієнтами.

Використання глибокого навчання для розпізнавання облич та визначення віку та статі з використанням бібліотек OpenCV дозволило досягти точних результатів. Важливо зазначити, що впровадження подібної системи вимагатиме уважного врахування питань конфіденційності та етичності обробки персональних даних клієнтів. Також слід враховувати можливість виникнення технічних або програмних помилок, що можуть вплинути на коректність роботи системи. Також важливо продовжувати вдосконалювати алгоритми та здійснювати моніторинг ринкових тенденцій у сфері технологій штучного інтелекту для підтримки конкурентоспроможності розробленої системи.

3 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

3.1 Розробка архітектури програмного продукту

У данном підрозділ розпочинається комплексне дослідження складного процесу створення архітектури та алгоритмів, що керують функціональністю касових апаратів самообслуговування. Тут міститься глибокий аналіз і роз'яснення ретельного планування, стратегічного проектування та методологій систематичного розвитку, які використовуються для формування фундаментальної структури та робочих алгоритмів програмного забезпечення для касових апаратів самообслуговування. Розглянемо основні принципи та їх вплив на функціонал системи, а також розгорнемо ключові етапи, спрямовані на визначення структури та логіки функціонування розроблюваного продукту:

- Перший етап, рівень бази даних, є фундаментальним, оскільки він визначає, як система зберігатиме та організовуватиме свої дані. Цей рівень служить основою, визначаючи структуру, методи зберігання та організацію інформації в системі. На цьому етапі здійснюється ретельне планування та проектування для встановлення архітектури бази даних. Це передбачає визначення схеми, яка описує структуру даних, включаючи таблиці, поля, зв'язки та обмеження. Крім того, рішення щодо вибору системи керування базами даних (СКБД), реляційної (наприклад, MySQL, PostgreSQL) або NoSQL (наприклад, MongoDB, Cassandra), приймаються на основі вимог системи до масштабованості, складності даних і продуктивності [27].
- Наступний рівень, рівень доступу до даних (Data Access layer), полегшує взаємодію з базою даних і охоплює механізми обробки даних у системі. Цей рівень в першу чергу передбачає створення, виконання та керування транзакціями та запитами, що забезпечує безперебійне отримання даних, маніпуляції та закриття транзакцій. На цьому етапі основна увага зосереджена на розробці функціональних можливостей, які дозволяють

системі отримувати доступ і маніпулювати даними, що зберігаються в базі даних. Це передбачає створення інтерфейсів і модулів, які дозволяють програмі взаємодіяти з основною базою даних.

- Рівень бізнес-логіки (Business Logic layer) служить основним двигуном системи, відповідальним за обробку та перетворення інформації, введеної користувачем, у дані, які система може зрозуміти та ефективно використовувати. На цьому етапі фокус зміщується в бік впровадження бізнес-правил, алгоритмів і робочих процесів, які керують логікою та поведінкою програми. Цей рівень інкапсулює специфічні для домену правила та операції, необхідні для функціональності та операцій системи [28].
- Сервісний рівень (The Service layer) служить шлюзом, через який проходять усі запити до клієнтської системи. Цей рівень визначає, як система надає послуги та взаємодіє з клієнтом. Основна увага зосереджена на визначенні та реалізації послуг, пропонованих системою, а також на встановленні засобів взаємодії між серверною частиною системи та клієнтськими компонентами. Цей рівень абстрагує основні функції, забезпечуючи чітко визначений інтерфейс для спілкування клієнтів із системою.
- Презентаційний рівень (Presentation layer), відповідає за візуалізацію даних і надання веб-сторінок видимим користувачам. На цьому етапі визначається, як дані будуть представлені для користувача через веб-інтерфейс. Презентаційний рівень містить розробку компонентів інтерфейсу користувача, включаючи веб-сторінки, графічні елементи, форми, меню та інші візуальні аспекти, з якими користувачі взаємодіють під час використання системи. Він передбачає використання таких технологій, як HTML, CSS, JavaScript, і зовнішніх фреймворків (таких як React, Angular або Vue.js) для створення адаптивних і зручних інтерфейсів.

Перелічені рівні взаємодіють, створюючи комплексну архітектуру програмного продукту [29].

На рисунку 3.1 наведено приклад архітектури програмного продукту.

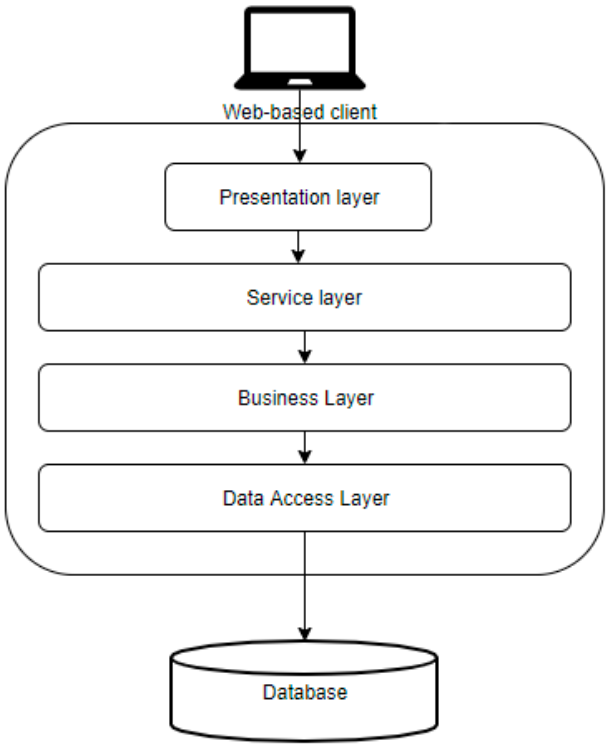

Рисунок 3.1 – Архітектура програмного продукту

Схема ілюструє ієрархічну структуру, що нагадує піраміду з різними рівнями, що представляють компоненти архітектури програмного забезпечення. На найвищому рівні піраміди знаходиться «Web-based client», що вказує на інтерфейс користувача, доступний через веб-браузер. Далі стрілки спрямовані вниз, зображуючи потік взаємодії та даних між різними рівнями. Шари, зображені в порядку спадання, демонструють наступні слої:

- Презентаційний рівень, який відповідає за візуалізацію даних і створення інтерфейсу користувача для веб-клієнта. Він обробляє логіку презентації, включаючи рендеринг веб-сторінок і взаємодії з користувачем.
- Сервісний рівень розташований під рівнем презентації та діє як посередник між інтерфейсом і серверною частиною. Він керує зв’язком між клієнтом і сервером, обробляє запити, реалізує бізнес-логіку та надає різноманітні послуги на рівні презентації.

- Бізнес-рівень поміщений нижче сервісного рівня і складає основну бізнес-логіку програми. Він інкапсулює правила, алгоритми та процеси, які керують функціональністю та операціями системи.
- Рівень доступу до даних розташований ще нижче і відповідає за керування взаємодією між програмою та сховищем даних. Він включає функції для отримання, зберігання та маніпулювання даними з основних систем зберігання даних, таких як бази даних.
- У нижній частині піраміди рівень бази даних представляє фактичні системи зберігання даних, наприклад бази даних. Цей рівень зберігає та керує даними програми.

Схема демонструє багаторівневу архітектуру системи, наголошуючи на розподілі завдань і модульній конструкції різних компонентів.

Приклад загальної архітектури програмного продукту показано на рисунку 3.2.

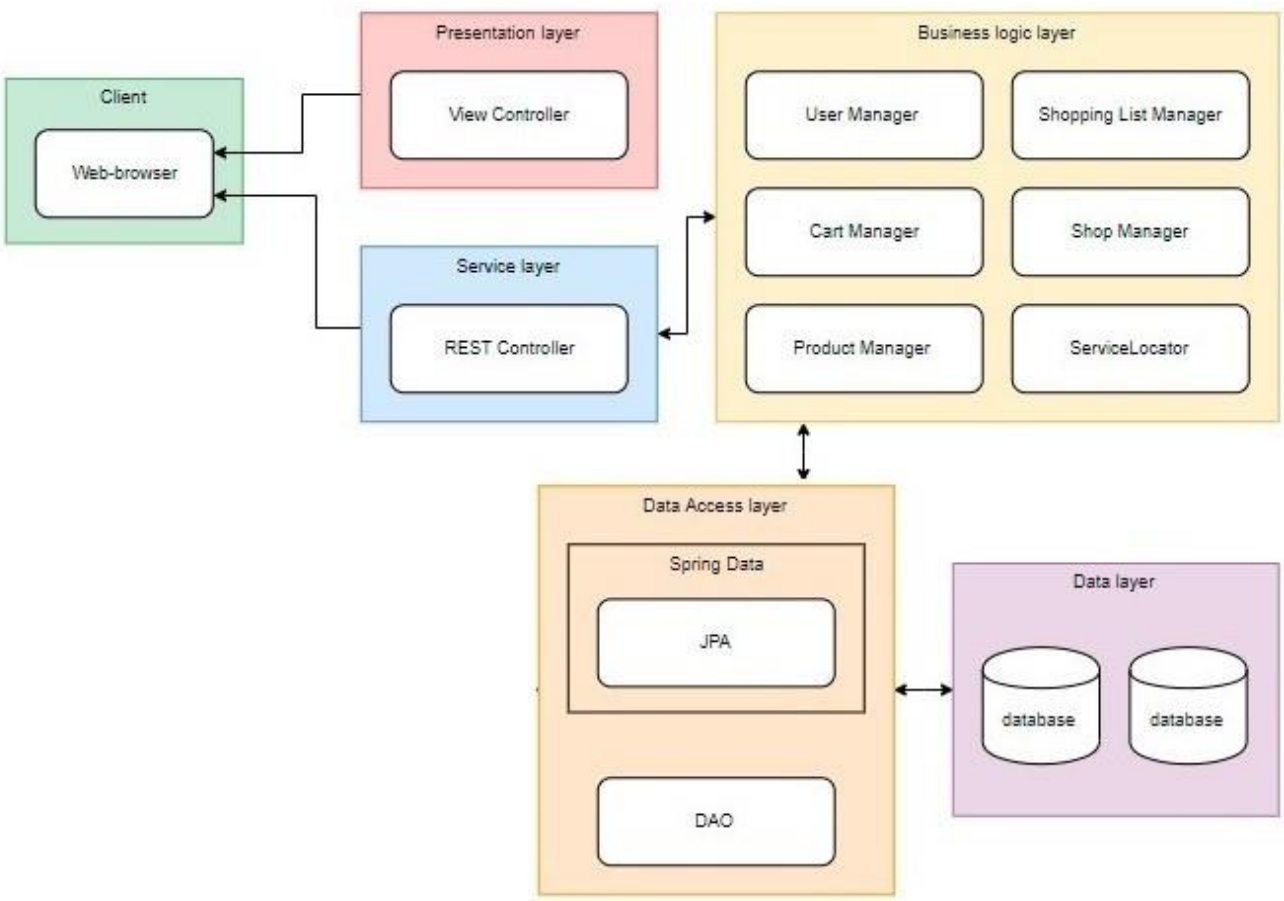

Рисунок 3.2 – Модель багаторівневої архітектури

Схема представляє високорівневу архітектуру програмного продукту. Він складається з різних рівнів компонентів, які взаємодіють, щоб забезпечити функціональність:

Стовпець «Клієнт», розташований ліворуч, представляє кінцевого користувача або інтерфейс користувача програми. Рівень презентації (View Controller), який прилягає до стовпця «Клієнт», керує логікою та взаємодією інтерфейсу користувача. Він відповідає за представлення даних користувачам і фіксацію введених ними даних.

Рівень обслуговування, розташований під рівнем презентації, містить контролери REST, які обробляють вхідні запити HTTP від рівня презентації. Він служить інтерфейсом між інтерфейсом (рівнем презентації) і сервером. Нижчий за рівень обслуговування, бізнес-рівень містить кілька модулів, відповідальних за керування певними функціями програми. Такі модулі, як User Manager, Card Manager, Shopping List Manager, Shop Manager, Product Manager і ServiceLocator, реалізують основну бізнес-логіку та виконують операції, пов’язані з відповідними доменами [30].

Рівень доступу до даних, розташований під бізнес-рівнем, взаємодіє з бізнес-рівнем, щоб забезпечити доступ до даних із базових систем зберігання даних. Він може використовувати такі технології, як Spring Data (JPA) і Data Access Objects (DAO), щоб керувати доступом до даних і постійністю. У нижній частині архітектури рівень даних представляє фактичне сховище даних, наприклад бази даних. Цей рівень зберігає та керує даними програми.

Загалом схема демонструє потік керування та даних через різні рівні архітектури програмного забезпечення, починаючи від інтерфейсу користувача (клієнта) через рівень презентації, рівень обслуговування, бізнес-рівень, рівень доступу до даних і, нарешті, рівень даних, де дані зберігаються та витягуються. Кожен рівень виконує певні завдання та взаємодіє з сусідніми рівнями для полегшення функціональності всієї системи.

Презентація та архітектурні рівні представлені на рисунку 3.3.

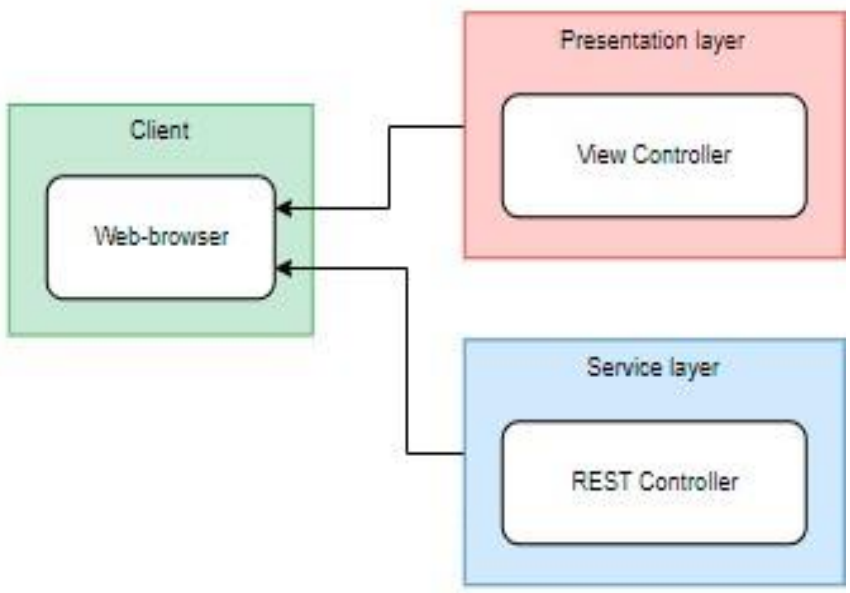

Рисунок 3.3 – Презентації та архітектура рівнів

Onion архітектура, також відома як «чиста архітектура», є особливо корисною в сценаріях, де потрібен високий рівень відокремлення, зручності обслуговування та тестування [31]. З огляду на ці фактори, її розумно використоввати у нашому випадку. Архітектура Onion була запропонована для створення додатків, які легше тестувати та надійніше працюють. Вона вирішує проблеми, які виникають у трьохрівневих та багаторівневих архітектурах, розбирається з багатьма іншими питаннями. Архітектурні шари Onion взаємодіють між собою через інтерфейси. Застосування принципу інверсії залежностей (DIP) в архітектурі Onion дозволяє всьому проекту отримати користь кількома способами:

- Впроваджуючи інтерфейси між рівнями, рівні високого рівня (такі як презентація чи інфраструктура) залежать від абстракцій (інтерфейсів), а не від конкретних реалізацій. Це роз’єднує шари та дозволяє змінювати один шар, не впливаючи на інші. Це забезпечує гнучкість у заміні або модернізації реалізацій без впливу на всю систему.
- DIP полегшує тестування, дозволяючи використовувати макетні об’єкти або заглушки для імітації поведінки під час модульного тестування. Із

залежностями, абстрагованими за інтерфейсами, імітаційні фреймворки можна використовувати для ізоляції компонентів з метою тестування.

- Інверсія залежностей через інтерфейси дозволяє вводити нові реалізації без зміни існуючого коду. Ця гнучкість дозволяє системі ефективніше адаптуватися до мінливих вимог або оновлень технологій.
- Використання інтерфейсів у структурі залежностей забезпечує стабільність в абстракціях. Стабільні інтерфейси, як правило, змінюються рідше, ніж конкретні реалізації, що зменшує ймовірність поломки змін у всій системі.
- Явне використання інтерфейсів і абстракцій покращує читабельність коду та зручність обслуговування. Це допомагає зрозуміти залежності та контракти між різними рівнями, роблячи кодову базу більш зрозумілою та зручнішою для розробників.

По суті, використання інверсії залежностей у всьому проекті в рамках архітектури Onion забезпечує слабкий зв’язок, сприяє тестуванню, підвищує гнучкість і сприяє загальній зручності обслуговування та масштабованості програмної системи. Сама ідея архітектури Onion полягає в потоці залежностей та взаємодії шарів. Глибший шар розташований всередині архітектури – він має менше залежностей. Тому архітектура Onion ідеально підходить для розробки данного програмного продукту. В архітектурі Onion глибокі шари мають обмежену залежність, сприяючи прозорості та легкості тестування. Це ідеальний вибір для розроблюваного програмного продукту, який забезпечує ефективну реалізацію.

Приклад архітектури подано на рисунку 3.4.

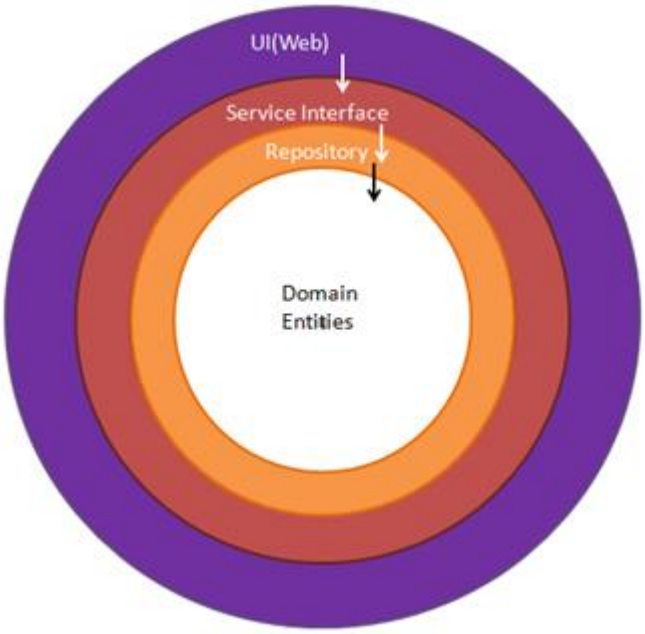

Рисунок 3.4 – Приклад архітектури Onion

Структура архітектури включає різні рівні, кожен з яких виконує конкретні функції:

- Рівень об'єктів домену (Domain Entities Layer): центральна складова, де розташовані об'єкти домену, такі як класи POCO або Edmx. Вони є основою програми та не мають зайвих залежностей, що сприяє їхній автономності.
- Рівень сховища (Repository Layer): служить для створення абстракції між сутностями домену та бізнес-логікою. Використовуючи шаблон доступу до даних, цей рівень взаємодіє з джерелом даних, перетворює їх для бізнес-суб'єкта і зберігає зміни.
- Рівень сервісу (Service Layer): включає інтерфейси для взаємодії між рівнем інтерфейсу користувача та рівнем сховища. Тут також розміщена бізнес-логіка, яка надається інтерфейсами для використання в рівні інтерфейсу користувача.
- Рівень інтерфейсу користувача (UI Layer): зовнішній шар, представляє собою веб-додаток, веб-API або тестовий проект. Принцип інверсії залежностей (DIP), застосований до рівня інтерфейсу користувача (UI), наголошує на абстракції залежностей і використанні інтерфейсів для створення більш гнучкої та зручної для обслуговування системи.

Виходячи з поданої вище інформації, у підрозділі були визначені вимоги до архітектури програмного продукту та обрано архітектуру Onion для подальшої розробки. Її основні переваги полягають у високій тестованості та надійності, що робить її ідеальним вибором для розробки системи самостійного касового розрахунку.

3.2 Розробка алгоритмів роботи додатку

Щоб створити систему самостійного касового розрахунку, потрібно мати комплексний алгоритм, який описує її роботу та функціональні можливості. Крім того, важливим аспектом буде створення структури бази даних для зберігання та керування вхідними та вихідними даними. Структура алгоритму, яка подана на рисунку 3.5, повинна охоплювати різні етапи, такі як отримання даних, обробка, взаємодія з базою даних за допомогою протоколів HTTP, створення квитанцій на основі інформації бази даних, перевірка характеристик користувача і відображення результатів на екрані.

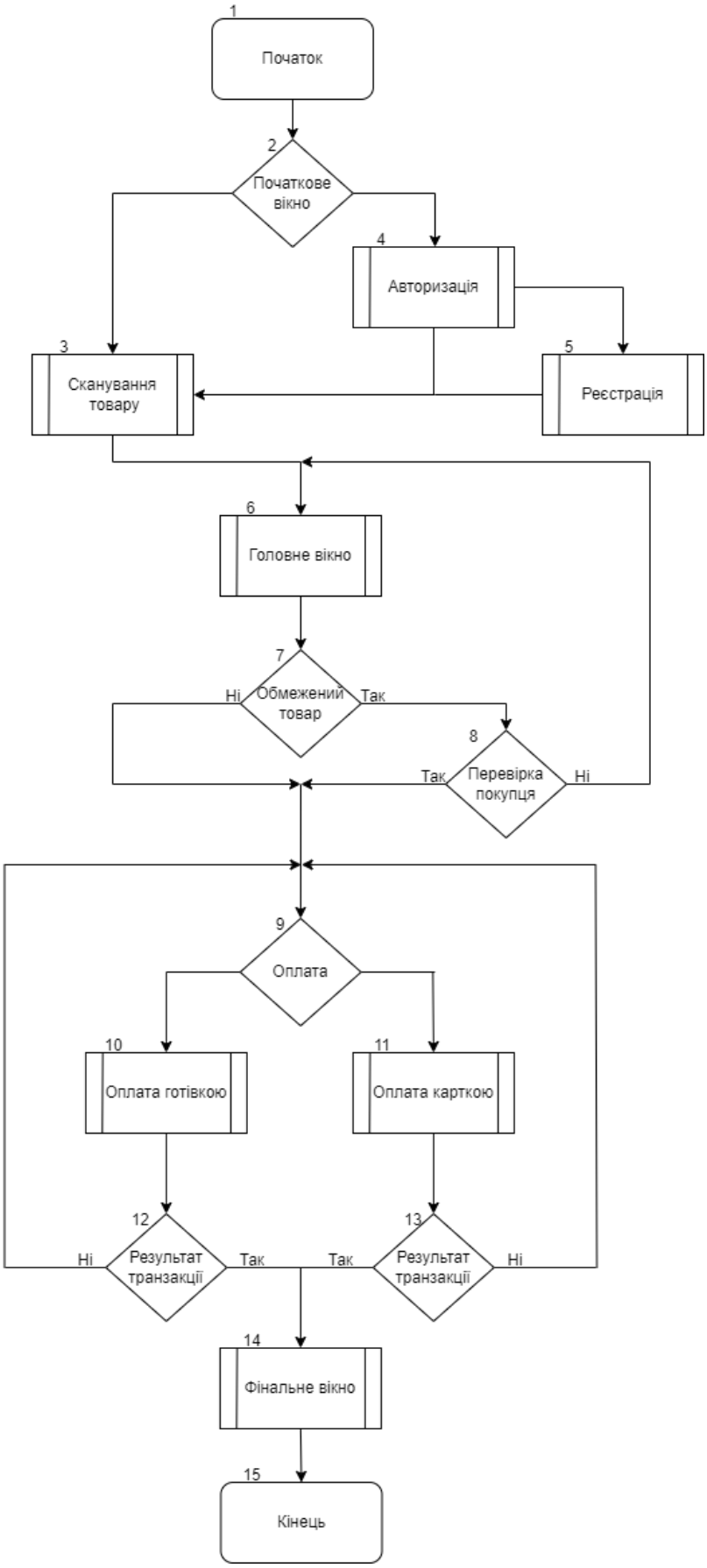

Рисунок 3.5 – Схематичний вигляд комплексного алгоритму функціонування програмного продукту

З метою формування розуміння принципу роботи алгоритму необхідним є провести покроковий аналіз його складових:

1. Початок.
2. Початкове вікно. Тут користувач може відсканувати товари, а також перейти до вікна аутентифікації системи, а потім до вікна реєстрації за необхідності.
3. Сканування товару. Користувачі можуть сканувати всі необхідні товари, і система перевіряє базу даних на наявність кожного товару. Після цього користувач автоматично повертається до головного вікна системи.
4. Аутентифікація. У цьому вікні користувачі можуть увійти в систему і відразу сканувати товари.
5. Реєстрація. У цьому вікні користувачі можуть зареєструватися в системі і також відразу сканувати товари.
6. Головне вікно. У цьому вікні відображаються всі просканові товари та загальна сума покупки. Користувачі можуть вибрати між оплатою готівкою або карткою. Після цього користувач переходить до вікна оплати.
7. Обмежений товар. На цьому етапі перевіряється наявність обмежених по віку товарів.
8. Перевірка покупця. Далі застосовується алгоритм перевірки характеристики покупця і по результату перевірки відбувається перехід на наступне вікно або прохання видалити з кошика обмежений товар.
9. Оплата. Дана функція передбачає можливість для користувачів здійснити проплати за всі покупки, використовуючи різні методи оплати.
10. Готівковий розрахунок. Після обрання даної функції покупець оплачує товари й чекає результат транзакції.
11. Результат оплати (готівка). У разі успішної операції користувач бачить можливість переходу до кінцевого вікна. В іншому випадку, система поверне його до вікна оплати.
12. Безготівковий спосіб. Користувач має змогу обрати варіант оплатити товари карткою. В цьому випадку ним здійснюється оплата з використанням

банківської картки, після чого відбувається процес очікування успішної транзакції.

13. Результат оплати (карта). Результатом успішної транзакції буде перехід користувача до кінцевого вікна. При неуспішній фінансовій обробці покупець повертається до вікна оплати.
14. Завершальний етап. Наприкінці покупки користувач зазвичай зустрічає повідомлення з підтвердженням успішної оплати та подякою за покупку після етапу генерації квитанції.
15. Кінець.

3.3 Розробка алгоритму рекомендації додаткових товарів на основі індивідуальних характеристик покупця

У магістерській кваліфікаційній роботі був створений алгоритм для розрахунку індивідуальних клієнтських знижок. Структуру цього алгоритму можна побачити на блок-схемі, яка зображена на рисунку 3.6.

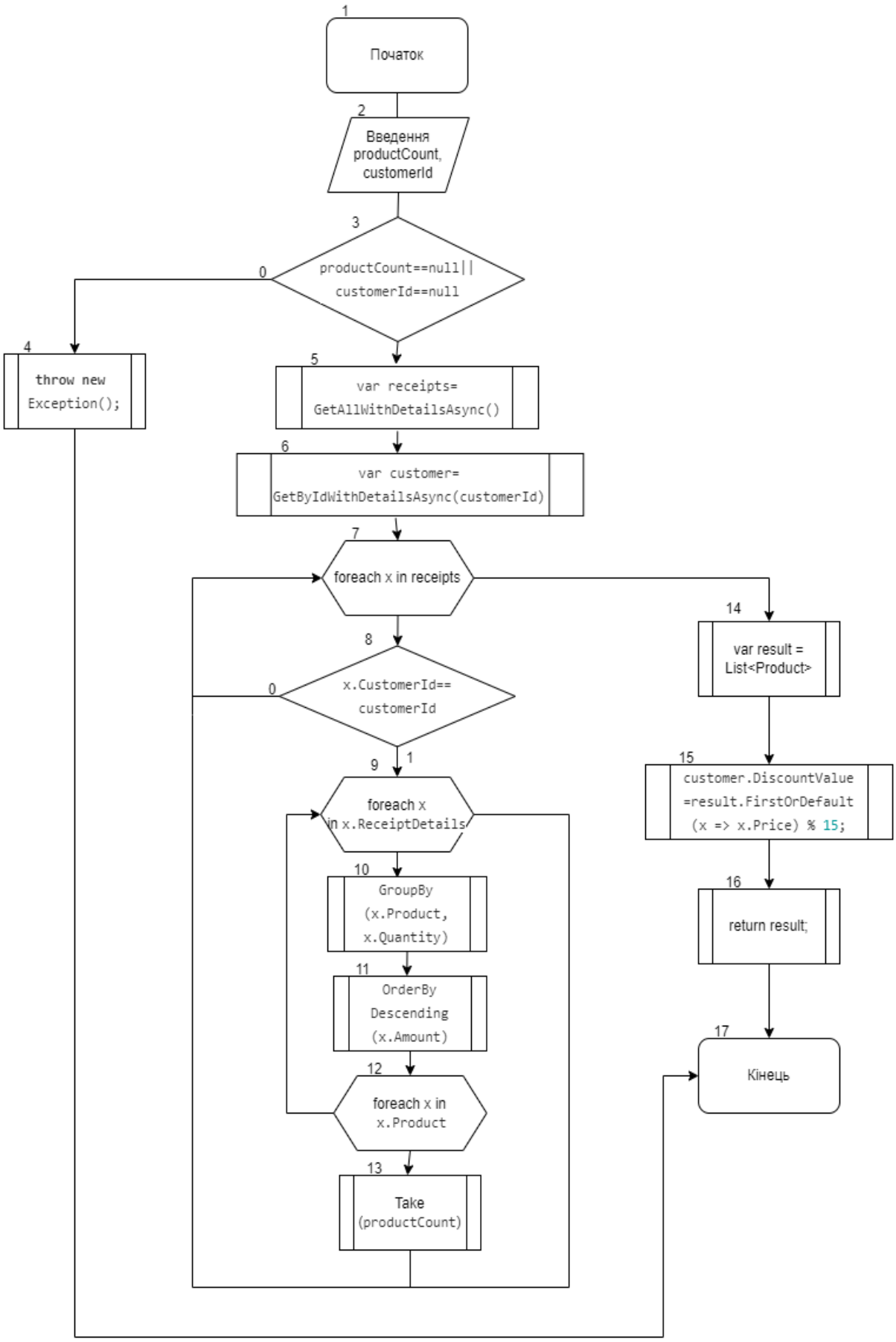

Рисунок 3.6 – Блок-схема алгоритму для розрахунку індивідуальних знижок

Для більш чіткого усвідомлення принципу роботи алгоритму, розглянемо його крок за кроком у більш докладний спосіб:

1. Початок.
2. Введення змінних productCount і customerId.
3. Перевірка вхідних даних на наявність нульових значень.
4. Виключення, якщо перевірка показує наявність нульових значень.
5. Отримання чеків.
6. Отримання інформації про клієнтів.
7. Цикл для визначення найбільш популярного товару.
8. Пошук за унікальним ідентифікатором.
9. Цикл для знаходження товару в деталізованих записах чеків.
10. Групування за товаром та їх кількістю.
11. Сортування за зменшенням за кількістю.
12. Цикл для знаходження конкретного товару.
13. Вибірка за переданою кількістю в алгоритм.
14. Отримання результату з циклу.
15. Встановлення знижки.
16. Повернення результату.
17. Завершення.

Отже, у цьому підрозділі був розроблений основний алгоритм програмного продукту та процедура генерації знижок, а також проведений детальний аналіз їхньої роботи крок за кроком.

3.4 Розробка бази даних

Проектування бази даних є вирішальним етапом у розробці програмного забезпечення, особливо при побудові систем, таких як касові апарати самообслуговування. У цьому розділі розглядаються фундаментальні аспекти та методології, пов’язані з процесом розробки бази даних для таких систем.

Розробка бази даних включає кілька етапів, кожен з яких відіграє вирішальну роль у забезпеченні ефективної та функціональної системи. Він охоплює концептуалізацію, логічне структурування та фізичну реалізацію архітектури бази даних [32].

На етапі проектування необхідно враховувати такі фактори, як цілісність даних, масштабованість, нормалізація та безпека. Ці аспекти є ключовими для створення міцної та надійної структури бази даних [33].

Розуміння життєвого циклу бази даних включає кілька етапів, починаючи від попереднього планування до концептуалізації, логічного структурування та остаточної фізичної реалізації. Кожен етап робить значний внесок у загальну якість і функціональність бази даних [34].

Етапи життєвого циклу бази даних:

- Етап попереднього планування: визначення обсягу проєкту, цілей і початкових вимог.
- Етап перевірки здійсненності: оцінка практичності та життєздатності запропонованої системи бази даних.
- Етап визначення вимог: визначення та документування детальних специфікацій і функцій, необхідних для бази даних.
- Проєктування концептуального типу: створення високорівневої абстрактної моделі бази даних.
- Проєктування логічного типу: переведення концептуальної моделі в більш детальну логічну схему.
- Проєктування фізичного типу: впровадження логічного проєктування у фізичні структури зберігання та методи доступу до даних.

Кожен із цих етапів має важливе значення для забезпечення того, щоб отримана база даних була добре структурованою, ефективною та відповідала функціональним вимогам системи [35].

Порівняльні характеристики наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння баз даних

	Oracle RDBMS	Microsoft SQL Server	MySQL HeatWave	MongoDB	Apache Cassandra
SQL синтаксис	1	1	1	0.5	0
Об’єм даних	1	1	0	0	1
Швидкість читання	1	1	1	0.5	0
Швидкість запису	0.5	0.5	0.5	1	1
Реляційна модель	1	1	1	0	0
Транзакційність	1	1	1	0.5	1
Атомарність	1	1	1	1	1
Консистентність	1	1	1	0.5	1
Ізоляція	1	1	1	0.5	0.5
Довговічність	1	1	1	1	1
Реплікація	0	0.5	0	1	1
Безкоштовна ліцензія	0	1	1	1	1
Підсумок	9.5	11	9.5	7.5	8.5

Враховуючи порівняльний аналіз, найбільш вигідним та задовільним варіантом стає використання бази даних Microsoft SQL Server. Однозначним плюсом цієї системи є підтримка реляційної моделі даних, дотримання принципів ACID, присутність індексів та підтримка стандартного синтаксису SQL [36]. Зокрема, безкоштовна ліцензія вважається безумовною перевагою даної бази даних.

Реалізація бази даних зображена на рисунку 3.7.

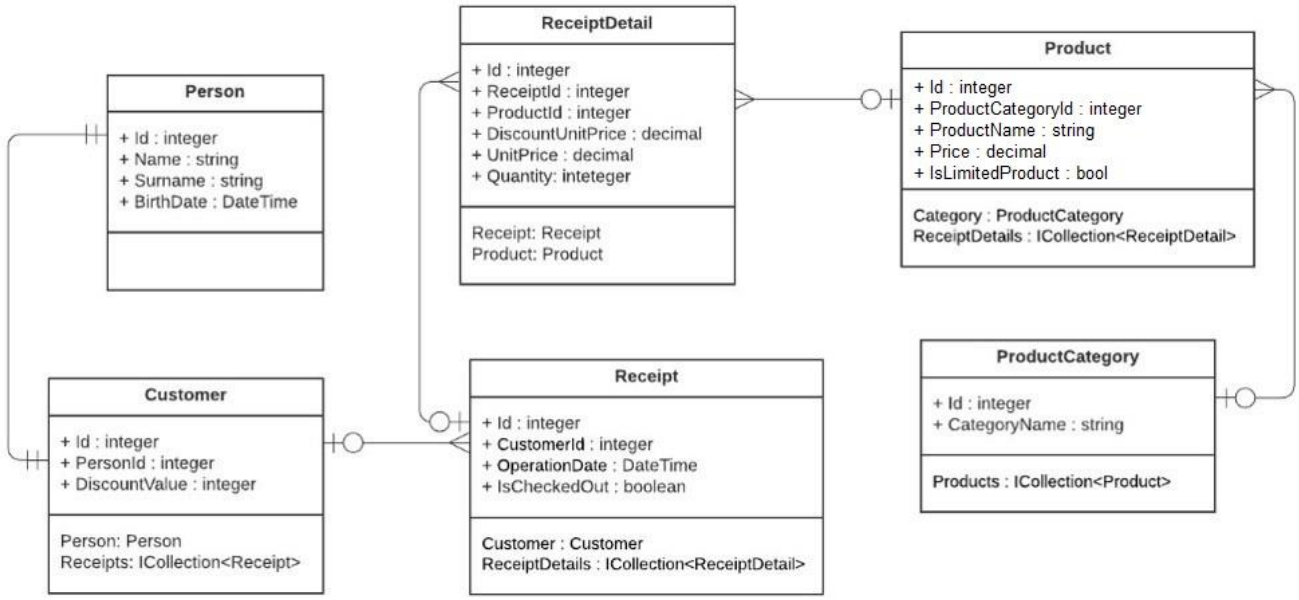

Рисунок 3.7 – Схема бази даних програмного продукту

База даних була реалізована за допомогою Entity Framework, що дозволяє створювати всі сутності, зв'язки та інші елементи бази даних, використовуючи лише мову програмування C# і без жодного SQL коду. Це забезпечує швидкі та гнучкі можливості у створенні різноманітних баз даних [37].

Розроблена база даних включає 6 таблиць з відповідними сутностями та зв'язками між ними. Між таблицями наявні наступні зв'язки:

- Person-Customer: зв'язок "один до одного".
- Customer-Receipt: зв'язок "один до багатьох".
- Receipt-ReceiptDetail: зв'язок "один до багатьох".
- Product-ReceiptDetail: зв'язок "один до багатьох".

- Product-ProductCategory: зв'язок "один до багатьох" [38].

У зв'язку "один до одного" кожен запис в таблиці Person може мати не більше одного відповідного запису в таблиці Customer, і навпаки. Зв'язок "один до одного" встановлюється, якщо обидва пов'язані стовпці є первинними ключами або мають унікальні обмеження [39].

Зв’язок «один до одного» показано на рисунку 3.8.

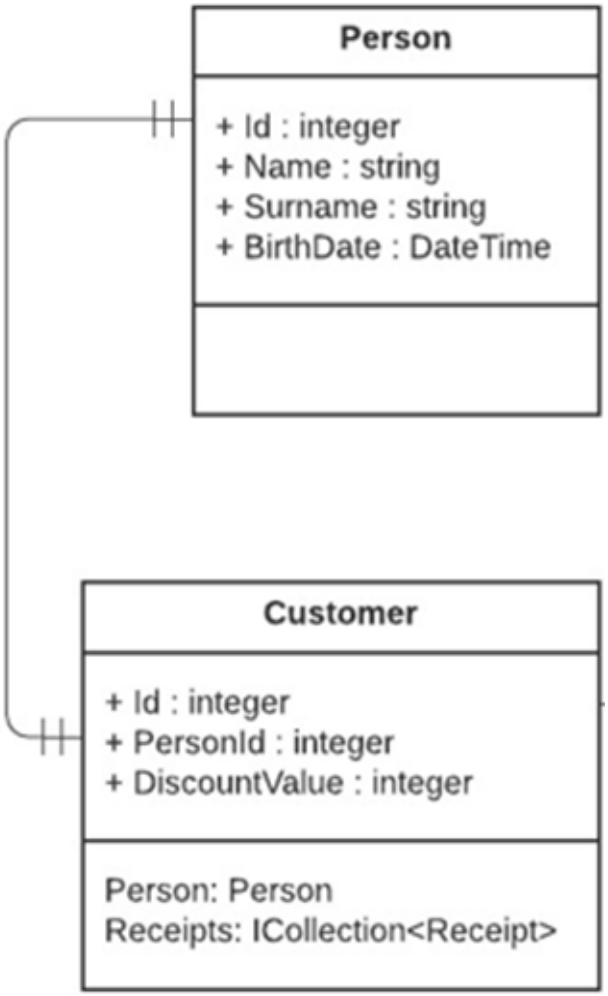

Рисунок 3.8 – Зв’язок «один до одного»

Уявімо таблицю ‘Customer’, у якій кожен рядок представляє іншого клієнта. Тепер припустімо, що ці клієнти зробили покупки, і ці покупки зберігаються в іншій таблиці під назвою ‘Receipt’. У цьому сценарії кожен клієнт може мати кілька квитанцій (що вказують на кілька покупок), але кожна квитанція належить лише одному клієнту. Наприклад стосунки «батько-дитина», коли один із батьків

може мати кількох дітей, але кожна дитина має лише одного батька [40]. Таке розташування демонструє зв’язок під назвою «один до багатьох» між клієнтами та їхніми квитанціями (рисунок 3.9).

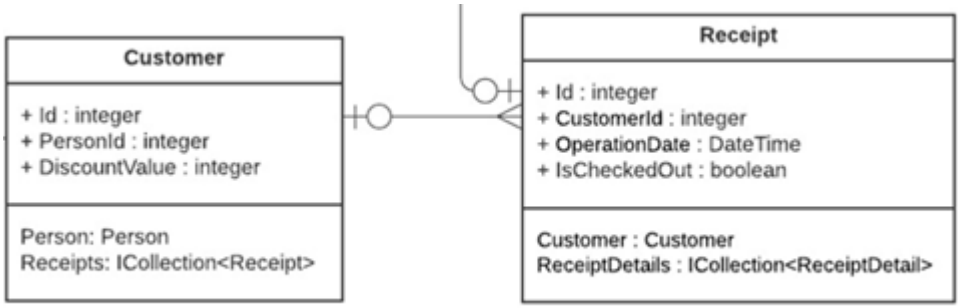

Рисунок 3.9 – Зв’язок «один до багатьох»

У створеній базі даних реалізовані наступні таблиці: Person, Customer, Receipt, ReceiptDetail, Product, ProductCategory.

Таблиця Person:

- Id: унікальний ідентифікатор користувача;
- Name: ім'я користувача;
- Surname: прізвище користувача;
- BirthDate: дата народження користувача.

Таблиця Customer:

- Id: унікальний ідентифікатор покупця;
- PersonId: унікальний ідентифікатор користувача;
- DiscountValue: персональна знижка.

Таблиця Receipt:

- Id: унікальний ідентифікатор замовлення;
- CustomerId: унікальний ідентифікатор покупця;
- OperationDate: дата покупки;
- IsCheckedOut: результат покупки.

Таблиця ReceiptDetail:

- Id: унікальний ідентифікатор детального замовлення;

- ReceiptId: унікальний ідентифікатор замовлення;
- ProductId: унікальний ідентифікатор товару;
- DiscountUnitPrice: ціна за одиницю товару зі знижкою;
- UnitPrice: ціна за одиницю товару;
- Quantity: кількість одиниць певного товару.

Таблиця Product:

- Id: унікальний ідентифікатор товару;
- ProductCategoryId: унікальний ідентифікатор категорії товару;
- ProductName: назва товару;
- Price: ціна товару.

Таблиця ProductCategory:

- Id: унікальний ідентифікатор категорії товару;
- CategoryName: назва категорії товару.

Отже, у цьому розділі були визначені основні вимоги до бази даних, проведено порівняння найбільш популярних баз даних, а також створено власну базу даних, розроблено та проаналізовано сутності, зв’язки та атрибути цієї бази даних.

3.5 Розробка структури графічного інтерфейсу користувача

GUI (графічний інтерфейс користувача) – це візуальний спосіб взаємодії користувачів з електронними пристроями або програмними програмами. Він використовує графічні елементи, такі як піктограми, кнопки, меню та вікна, щоб дозволити користувачам навігацію, взаємодію та виконання завдань більш зручним та інтуїтивно зрозумілим способом порівняно з текстовими інтерфейсами.

GUI був створений для полегшення взаємодії користувача з комп’ютерними системами. Графічні інтерфейси були розроблені для заміни текстових інтерфейсів більш інтуїтивно зрозумілими та візуально інтерактивними елементами. Вони спрямовані на спрощення способу взаємодії користувачів із

програмними програмами, надаючи графічні елементи, такі як піктограми, кнопки, меню та інші візуальні компоненти, які дозволяють користувачам легше й ефективніше переміщатися та виконувати завдання.

У порівнянні з операційною системою з командним рядком або CUI, GUI є більш простими для вивчення та використання, особливо для новачків. Вони не вимагають запам'ятовування команд і не вимагають від користувача знання мов програмування. Структура початкового вікна ілюстрована на рисунку 3.10.

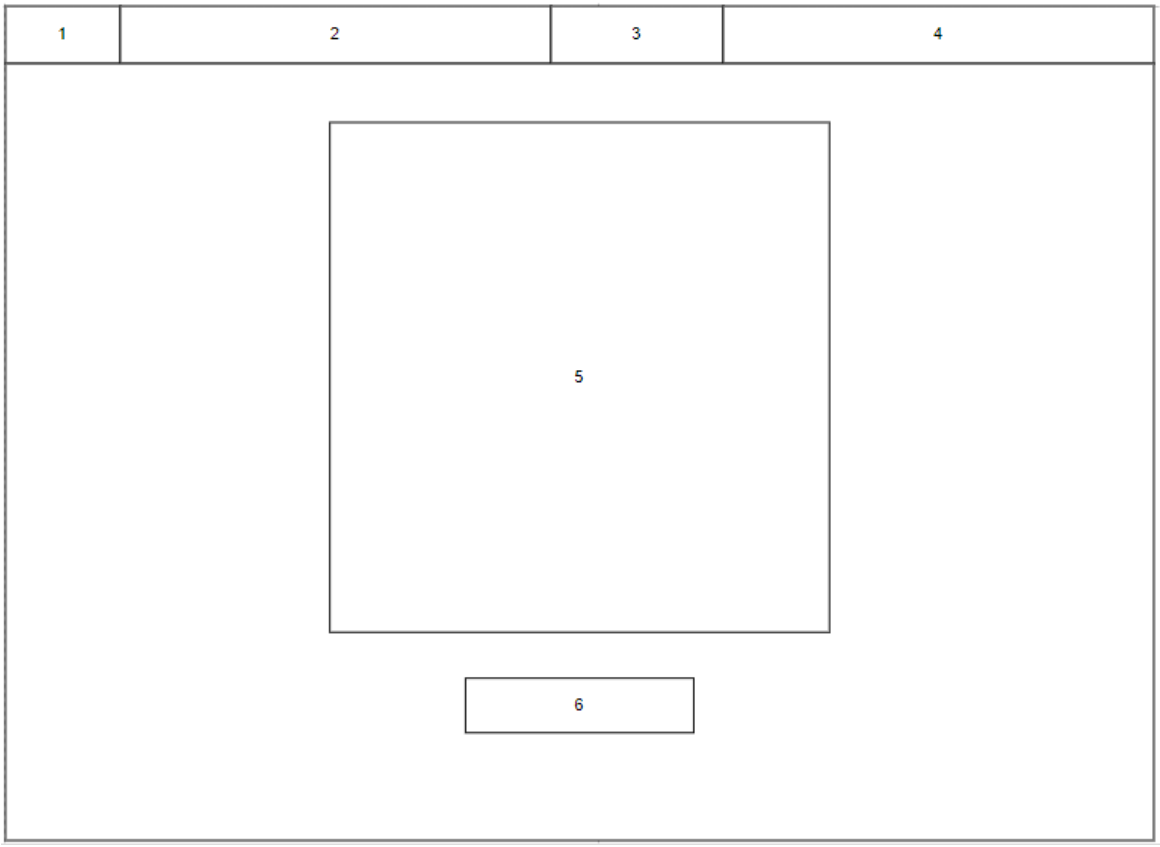

Рисунок 3.10 – Схема початкового вікна

Елементи інтерфейсу основного вікна включають:

1. Кнопка для зміни мови.
2. Сторінка для сканування товарів.
3. Сторінка з кошиком та оплатою товарів.
4. Фінальне вікно.
5. Привітання та логотип.

У головному вікні клієнт бачить вітання та має можливість відразу сканувати товари або авторизуватися. Розглянемо вікно авторизації на рисунку 3.11.

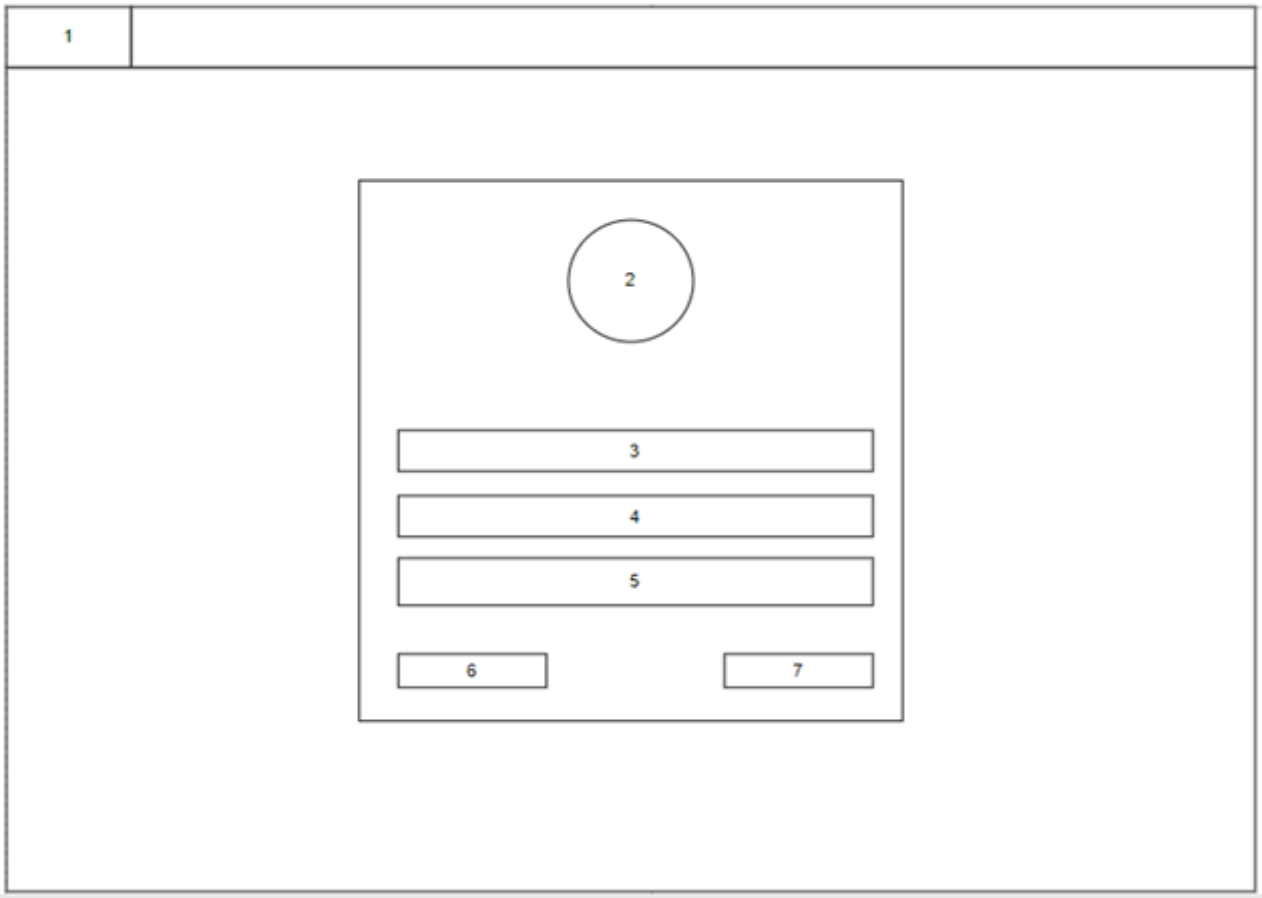

Рисунок 3.11 – Схема вікна з авторизацією

Елементи інтерфейсу основного вікна включають:

1. Кнопка зміни мови.
2. Зображення авторизації.
3. Поле для введення номеру телефону.
4. Поле для введення паролю.
5. Кнопка для входу.
6. Кнопка "Назад".
7. Кнопка для реєстрації.

У цьому вікні користувач може авторизуватися, зареєструватися або повернутися, натисканням кнопки "Назад". Тепер розглянемо головне вікно з товарами та способом оплати на рисунку 3.12.

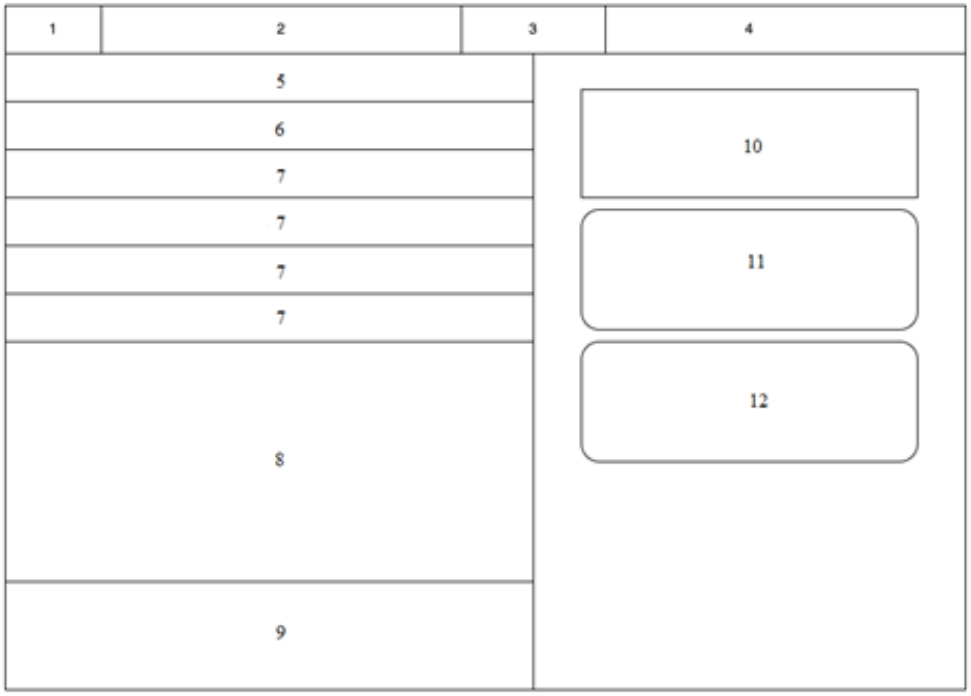

Рисунок 3.12 – Схема головного вікна додатку

Елементи інтерфейсу головного вікна включають:

1. Кнопка зміни мови.
2. Сторінка сканування товару.
3. Сторінка з чеком та оплатою товару.
4. Фінальне вікно.
5. Панель кошика з товаром.
6. Панель з товаром.
7. Товар із інформацією.
8. Вікно для товарів.
9. Панель із інформацією про суму покупки.
10. Панель для вибору способу оплати.
11. Кнопка оплати банківською карткою.
12. Кнопка оплати готівкою.

У цьому вікні користувач бачить всі відсканові товари, суму, яку повинен сплатити за них, і може вибрати спосіб оплати, а також змінити мову інтерфейсу. Розглянемо більш детально схему інтерфейсу вікна з оплатою, що зображена на рисунку 3.13.

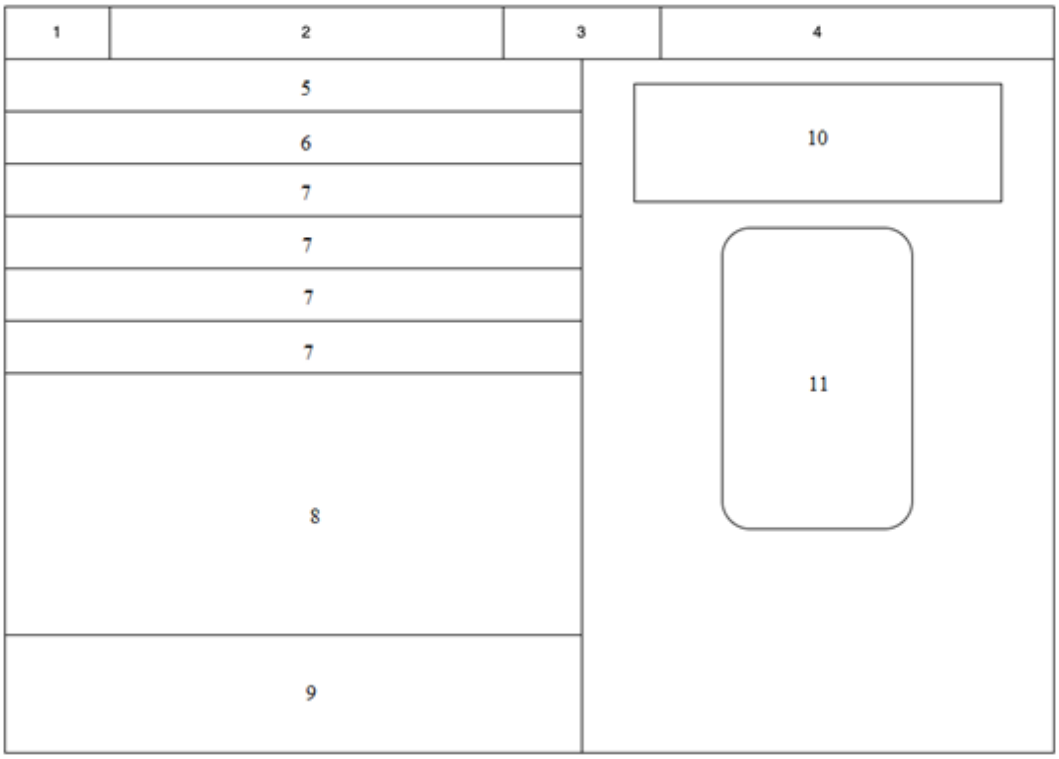

Рисунок 3.13 – Схема вікна з оплатою

Елементи інтерфейсу головного вікна включають:

1. Кнопка зміни мови.
2. Сторінка сканування товару.
3. Сторінка з чеком та оплатою товару.
4. Фінальне вікно.
5. Панель кошика з товарами.
6. Панель з товаром.
7. Товар із інформацією.
8. Вікно для товарів.
9. Панель із інформацією про суму покупки.
10. Панель застосування картки.

11.Логотип карткового терміналу.

На цьому вікні користувач безпосередньо розраховується за придбаний товар безготівковим способом оплати. На наступнумо вікні буде перевірка характеристик покупця (рисунок 3.14).

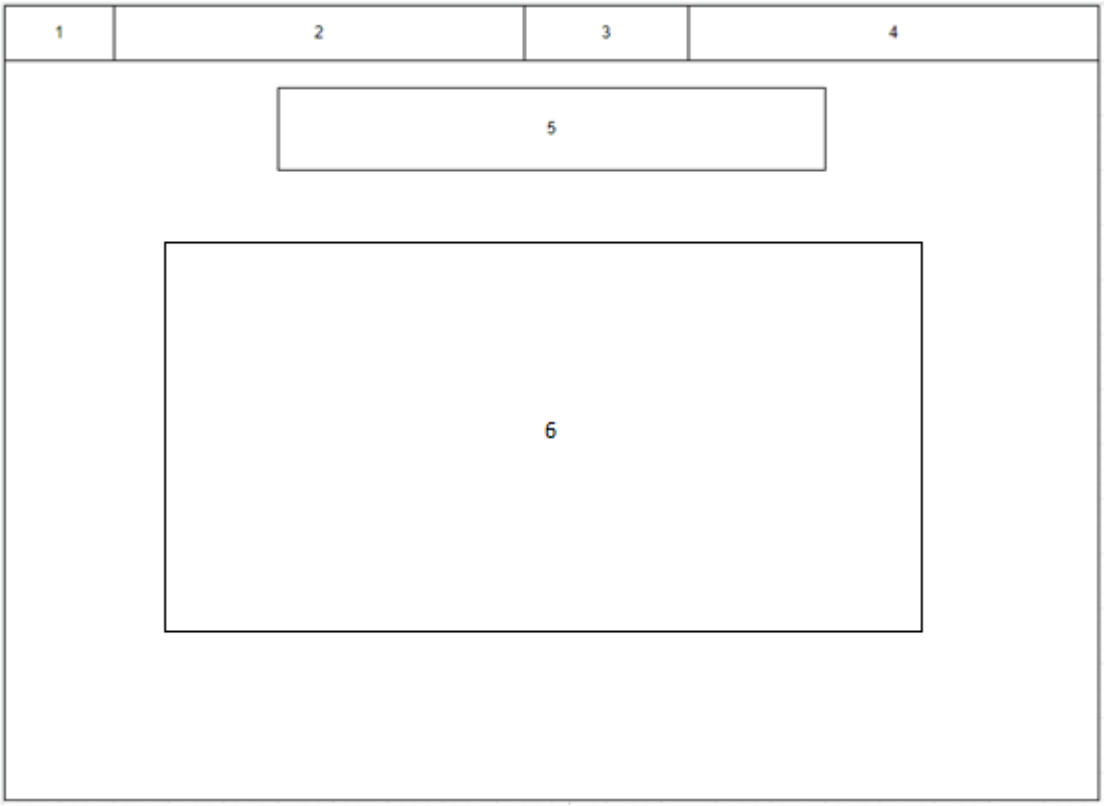

Рисунок 3.14 – Перевірка характеристик покупця

На вікні камери виводиться зображення, яке отримується з веб-камери. Це вікно відіграє ключову роль у роботі системи, оскільки саме тут відбувається процес перевірки та розпізнавання характеристик покупця.

Система в реальному часі аналізує отримане зображення, використовуючи технології штучного інтелекту та глибокого навчання. Найбільш важливі характеристики, такі як вік та стать покупця, визначаються з високою точністю та швидкістю завдяки застосуванню передових алгоритмів розпізнавання облич і аналізу біометричних даних.

Зелене підсвічування вказує на успішну перевірку та розпізнавання покупця як повнолітньої особи, тоді як червоний колір може свідчити про невдалу спробу

або неповну відповідність характеристик. Цей інтерфейс надає операторам чітку зворотну зв'язок та відображає результати автоматизованого аналізу для ефективного використання в умовах роздрібної торгівлі.

На фінальному вікні буде відображено результат успішної транзакції та вираз вдячності за покупку, як показано на рисунку 3.15.

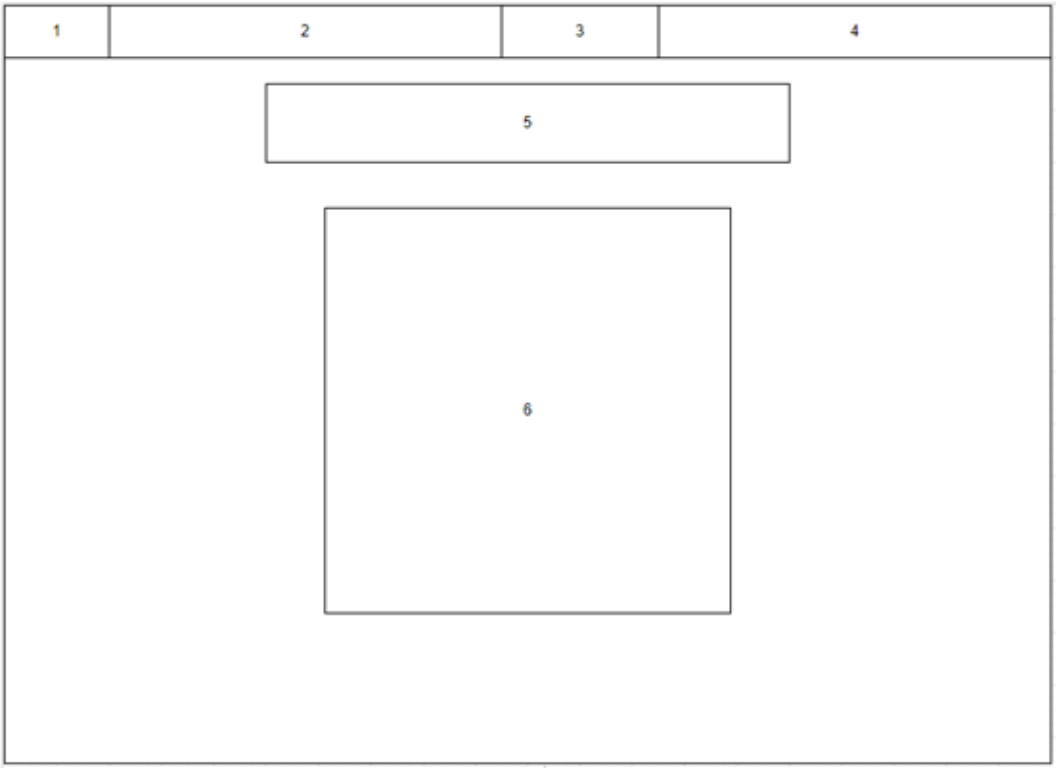

Рисунок 3.15 – Схема фінального вікна

Аналіз графічного інтерфейсу користувача (GUI) у контексті розробки структури інтерфейсу користувача виявив кілька ключових аспектів щодо його функціональності, характеристик та особливостей. Графічний інтерфейс користувача служить ключовим елементом у взаємодії з користувачем, забезпечуючи інтерфейс, який інкапсулює різні функції та полегшує взаємодію користувача з системою.

Графічний інтерфейс демонструє ретельно сплановний макет, особливо помітний у структурній схемі головного вікна програми. Цей структурний дизайн пропонує комплексне уявлення, представляючи чітку навігацію та доступ до різноманітних функцій у програмі. Крім того, для окремих вікон було розроблено

конкретні структурні схеми, кожне з яких адаптовано для виконання окремих цілей і операцій у рамках програми.

3.6 Висновки

Розробка архітектури програмного продукту, алгоритмічної структури та операцій з базою даних склали суть третього розділу. Увагу було зосереджено на розробці архітектури, плнуванні алгоритмічних послідовностей для операцій з базою даних, формулюванні алгоритмів для рекомендацій додаткових продуктів на основі індивідуальних характеристик клієнта, проєктування бази даних і окреслення структури графічного інтерфейсу користувача.

Загалом даний розділ зосередився на ключових аспектах, вирішальних для створення надійної основи програмного забезпечення. Архітектура служить основою для розроблки системи, алгоритм рекомендацій покращує взаємодію з користувачем, сама база даних зберігає важливу інформацію, а графічний інтерфейс користувача формує взаємодію між системою та користувачами. У сукупності ці компоненти створюють основну структуру та функціональність програмного продукту.

4 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ СИСТЕМИ САМОСТІЙНОГО КАСОВОГО РОЗРАХУНКУ

4.1 Варіантний аналіз і обґрунтування вибору мови програмування

Вибір мови програмування є ключовим у розробці програмного забезпечення, впливаючи на різні аспекти успіху програмного продукту. Приймаючи рішення, необхідно враховувати такі ключові фактори, як вимоги до проєкту, масштабованість, безпека, сумісність, вартість і довгострокова підтримка. Розумний вибір забезпечує узгодження з потребами проєкту, сприяє ефективності розробника та сприяє надійності, продуктивності та адаптивності програмного забезпечення в майбутньому. Базовими особливостями, що повинна містити мова програмування для зручності її використання є наступні:

- Простота та легкість вивчення. Дана властивість передбачає наявність чіткого синтаксису, логічну структуру та мінімальну складність концепцій і конструкцій, що сприяє швидкому розумінню та набуттю навичок для програмістів. Ця властивість скорочує криву навчання, дозволяючи розробникам ефективно працювати над проєктами.
- Абстракція. Дана критична характеристика мов програмування стосується здатності мови визначати складні структури та полегшувати їх використання за допомогою абстракції. Це передбачає спрощення та приховування складних деталей реалізації, зосередження на основних функціях і надання чіткого інтерфейсу для взаємодії розробників, не вимагаючи від них розуміння тонкощів основної складності.
- Портативність. За наявності цієї властивості перевага віддається портативним мовам програмування, які можна легко переносити між різними платформами.
- Ефективність. Властивість стосується здатності мови програмування виконувати завдання з оптимальним використанням ресурсів і швидкістю. Згадка про те, що мову можна легко конвертувати, зазвичай означає, що

код, написаний цією мовою, можна ефективно перекласти або перетворити на машинний код або виконуваний формат, що призводить до високопродуктивного виведення.

- Організація та документація. Характеристика підкреслює необхідність для мови програмування мати чітко визначені структури та вичерпну документацію. Цей атрибут гарантує, що синтаксис, правила, функціональні можливості та бібліотеки мови добре структуровані та задокументовані, що робить його зручним для розуміння, вивчення та використання розробниками під час створення програм.
- Доступність інструментів розробки. Це означає наявність повного набору інструментів і ресурсів, що надаються мовою програмування для підтримки розробки програмного забезпечення. Такі інструменти включають інтегровані середовища розробки (IDE), компілятори, відладчики, бібліотеки, фреймворки та інші утиліти, які допомагають розробникам у написанні, тестуванні, налагодженні та ефективному розгортанні програм.
- Інтегроване Середовище Розробки (IDE): забезпечення інтегрованого середовища розробки для зручності програмістів.
- Синтаксис та Семантика: мова повинна мати послідовний синтаксис та семантику.

У даному підрозділі пропонується огляд тих мов програмування, які найбільшою мірою відповідають запропонованим критеріям, а тому є найбільш доцільними для роботи з базами даних: Python, Java та C#.

Python – це відома мова програмування, яку визнають своєю простотою, читабельністю та великими бібліотеками. Його популярність пояснюється його дружнім синтаксисом, що робить його привабливим як для початківців, так і для досвідчених програмістів. Ключові переваги Python полягають у його читабельності, що дозволяє створювати чітку та зрозумілу структуру коду, а також у його універсальності, що дозволяє адаптуватись до різних парадигм програмування. Його величезна бібліотечна екосистема, що включає пакети для аналізу даних, веб-розробки, машинного навчання тощо, робить його

привабливим як надійний інструмент для розробки. Надійність мови пояснюється її добре перевіреними пакетами, активною підтримкою спільноти та регулярними оновленнями, що забезпечує стабільність та ефективність створення різноманітних програм. Загалом читабельність, універсальність, великі бібліотеки та надійність Python роблять його кращим вибором для розробників у широкому діапазоні доменів і програм [41].

Python, попри численні переваги, має деякі недоліки. Одним із помітних недоліків є його швидкість виконання порівняно з мовами нижчого рівня, такими як C або C++. Через свою інтерпретовану природу Python, як правило, повільніше виконується, особливо для завдань з інтенсивним використанням процесора або програм реального часу. Крім того, його глобальне блокування інтерпретатора (GIL) обмежує багатопотоковість, що може перешкоджати продуктивності багатопотокових програм, обмежуючи паралелізм у певних випадках. Хоча було докладено зусиль для усунення цих обмежень, вони залишаються міркуваннями для розробників, які шукають вищої продуктивності в певних областях.

Java – широко поширена мова програмування, відома своєю незалежністю від платформи та універсальністю. Він широко використовується в різних областях, включаючи веб-розробку, мобільні програми, корпоративне програмне забезпечення тощо. Спочатку розроблена Sun Microsystems, Java зараз перебуває під опікою корпорації Oracle після придбання нею Sun у 2010 році. Однією з ключових особливостей Java є її можливість «записати один раз, запустити будь-де», що підтримується віртуальною машиною Java (JVM). Код, написаний на Java, компілюється в байт-код, який можна запускати на будь-якому пристрої чи платформі з сумісною JVM, забезпечуючи переносимість між різними операційними системами [42].

Універсальність Java є однією з її значних сильних сторін, яка в першу чергу демонструється через незалежність від платформи, великі бібліотеки та широке застосування в різних областях. Універсальність Java походить від принципу WORA (Write Once, Run Anywhere), який дозволяє запускати код Java на будь-якому пристрої або платформі з сумісною віртуальною машиною Java

(JVM). Ця незалежність від платформи досягається за допомогою байт-коду, який дозволяє виконувати програми Java на різних платформах без модифікації, підвищуючи їх портативність і адаптивність.

Наступна сильна сторона мови полягає в тому, що Java відома своєю «суворою» або «строгою» типізацією, що відноситься до її статично типізованої природи. Це означає, що Java вимагає явного оголошення типів даних змінних, а перевірка типів виконується під час компіляції. Природа суворої типізації Java підвищує надійність коду, виловлюючи помилки під час компіляції, а не під час виконання, запобігаючи певним типам помилок і сприяючи більш надійному та стабільному коду.

Незважаючи на свою популярність, мова Java має свої недоліки. Одне з головних зауважень до Java – це багатослівність або “зайвість”. Для виконання певних завдань Java, як правило, потребує більше рядків коду порівняно з деякими іншими мовами програмування. Ця багатослівність може зробити код складнішим і важчим для читання, потенційно призводячи до більш тривалого часу розробки та збільшення зусиль на обслуговування.

C# – мова програмування високого рівня, розроблена Microsoft, яку часто порівнюють з іншими мовами, такими як Java і C++. Вона має спільні риси з Java щодо синтаксису та функцій об’єктно-орієнтованого програмування. C# відомий своєю простотою, легким для розуміння синтаксисом і сильною інтеграцією з платформою .NET, пропонуючи широкий спектр бібліотек і інструментів для створення різних програм. Порівняно з C++, C# зазвичай має більш зрозуміле керування пам’яттю завдяки своїй системі збирання сміття, що полегшує розробникам роботу з виділенням і звільненням пам’яті. Крім того, C# має такі функції, як LINQ (Language Integrated Query), які надають потужні можливості надсилання запитів. Загалом C# цінують за його універсальність, особливо у сфері розробки додатків Windows та розробки ігор за допомогою фреймворків, таких як Unity [43].

Об’єктно-орієнтований підхід, який використовується в C#, дозволяє розробникам створювати модульний і багаторазово використовуваний код,

організовуючи функціональні можливості в класи та об’єкти. Ця парадигма пропонує інкапсуляцію, успадкування та поліморфізм, що дозволяє розробникам створювати складні системи, зберігаючи при цьому простоту коду та можливість повторного використання. C# використовує такі функції, як властивості, події та делегати, що дозволяє реалізувати складні програми за допомогою чіткого, добре структурованого коду [44]. Крім того, його інтеграція з платформою .NET надає доступ до величезного набору бібліотек і інструментів, що полегшує розробку різних додатків, таких як настільні програми, веб-сервіси та ігри, що робить C# універсальною мовою для різноманітних цілей розробки програмного забезпечення.

C# володіє численними корисними особливостями, включаючи інкапсуляцію, успадкування, поліморфізм, перевантаження операторів і статичну типізацію. Наприклад, ця мова програмування постійно розвивається, і з кожною новою версією вводиться новий функціонал, такий як покращений синтаксис мови, бібліотеки та оптимізація продуктивності [45].

Серед переваг C# можна виділити:

- Фіксований розмір більшості типів даних, що сприяє "мобільності" C# та як результат робить програмування простішим завданням.
- Автоматичне управління сміттям, що звільняє розробників від необхідності вручну керувати пам'яттю. Середовище CLR використовує специфічний сміттєзбірник для автоматичної очистки пам'яті.
- Розгалужений синтаксичний "цукор", що являє собою ті конструкції, які полегшують написання коду та розуміння його при компіляції.
- Наступна властивість відноситься до низького входового бар’єру через схожість синтаксису C# з іншими мовами програмування», а це, своєю чергою, полегшує перехід для новачків.
- Можливість розробки програм для кількох операційних систем за допомогою єдиної кодової бази в Xamarin вважається функцією платформи.

При цьому, C# налічує недоліки, на які слід також звертати увагу при виборі мови програмування:

- Більша оптимізованість для платформи Windows.
- Безкоштовність мови, переважно спрямована на початківців, студентів та користувачів з невеликими проєктами [46]. Водночас, вартість ліцензій для більш крупних компаній є значною.

В таблиці 4.1 продемонстровано порівняльний аналіз зазначених в пункті мов програмування..

Таблиця 4.1 – Мови програмування: порівняння методом контент-аналізу

Властивість	Python	Java	C#
Простота синтаксису	0	0	1
Об’єктно-орієнтованість	1	1	1
Зручність при роботі із базами даних	0	1	1
Засоби симпліфікації аналізу даних	1	0	1
Наявність доступу до документації	1	1	1
Наявність різноманітних фреймворків, доступних для створення баз даних	1	1	1
Підсумковий результат	4	4	6

Висновок з таблиці 4.1 дозволяє стверджувати, що C# є оптимальним вибором, якщо потрібно створити базу даних для системи самостійного касового розрахунку. Це обумовлено наявністю зручних та необхідних фреймворків та інструментів, а також легким у вивченні та зрозумілим синтаксисом даної мови.

4.2 Вибір середовища розробки

Під час розробки бази даних для системи самостійного касового розрахунку потрібно особливу увагу приділити такому етапу як вибір інтегрованого середовища розробки (IDE). Інтегровані середовища розробки є потужними програмними рішеннями для розробки різноманітного програмного забезпечення.

Серед найбільш популярних IDE для реалізації програм мовою програмування C# слід відзначити Microsoft Visual Studio, Rider та Eclipse IDE.

Visual Studio заслужено визнана найкращою IDE для розробки на C#, що пояснюється тісним пов'язанням обох продуктів із корпорацією Microsoft [47]. Це IDE пропонує ряд переваг, серед яких:

Найкраща підтримка для розробки на C#.

- Великий набір інструментів, оптимізованих для роботи з C#.
- Наявність безкоштовної версії – Community Edition, яка містить все необхідне для незалежного розробника.
- Версія The Community Edition доступна на безплатній основі, що робить її привабливим варіантом для окремих розробників, студентів і невеликих команд. Попри те, що CE безплатний, він пропонує багатий набір функцій, який можна порівняти з іншими випусками, сприяючи доступності та інклюзивності в спільноті розробників.
- IDE пропонує надійну та ефективну підтримку таких мов, як C#, C++, Python, JavaScript тощо, що забезпечує широкі можливості для розробки в різних доменах.
- Завдяки інтеграції хмарного сховища MS Visual Studio Community Edition забезпечує плавний доступ до проектів і файлів із різних місць і пристроїв.

MS Visual Studio не позбавлений недоліків, серед яких слід окреслити наступні:

- Visual Studio потребує багато ресурсів, споживаючи значну кількість пам’яті та процесорної потужності під час роботи над великими проєктами.
- Платна версія передбачає ймовірність втрати налаштувань та корпоративного сервера.
- Складність VS створює труднощі для новачків, вимагаючи додаткового часу, щоб навчитися навігації та ефективного використання набору інструментів.

Представлений вигляд інтерфейсу Visual Studio (рисунок 4.1).

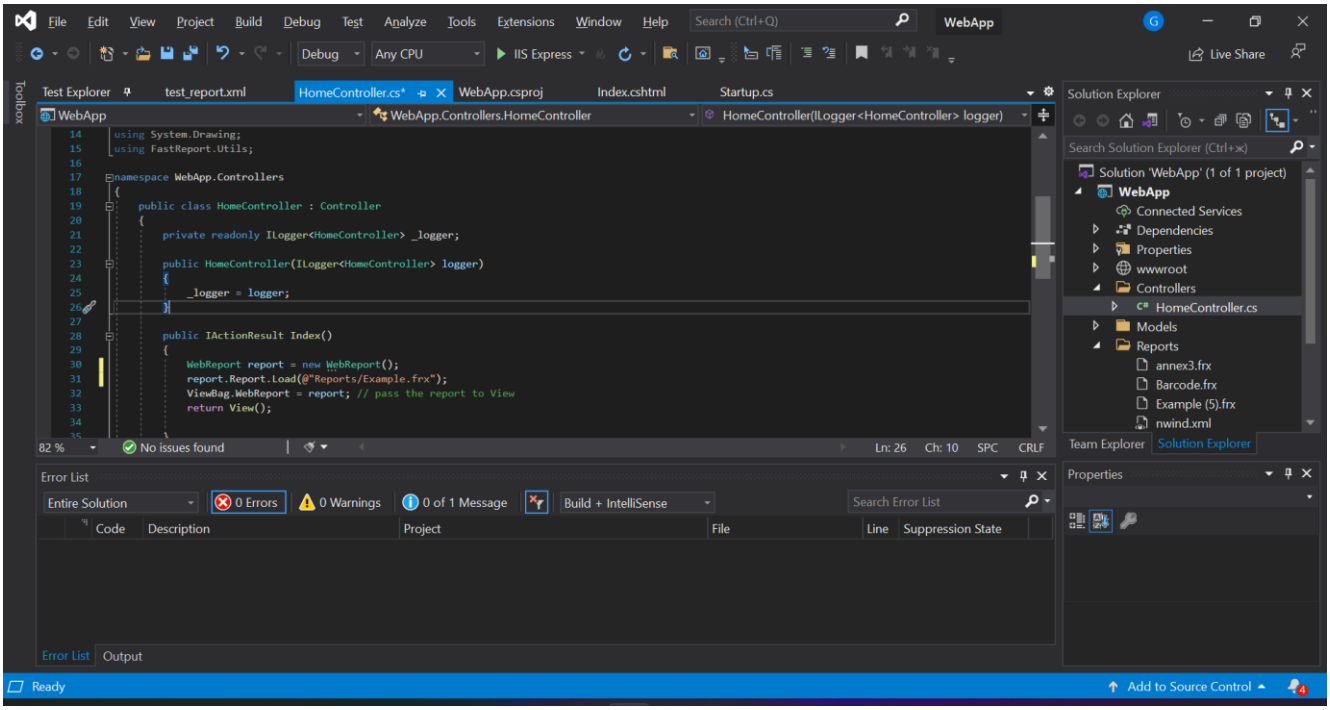

Рисунок 4.1 – Знімок екрана, на якому продемонстрований вигляд інтерфейсу Microsoft VS

Project Rider – це інтегроване середовище розробки (IDE), спеціально розроблене JetBrains для кросплатформної розробки .NET, що забезпечує підтримку C#, F#, VB.NET, ASP.NET та інших пов’язаних технологій [48].

Сильні сторони Project Rider містять в собі наступні характеристики:

- Підтримка між платформами: працює в Windows, macOS і Linux.
- Повна інтеграція з ReSharper: пропонує широкий спектр інструментів аналізу коду, навігації та рефакторингу.
- Підтримка кількох мов: підтримує C#, F#, VB.NET, ASP.NET та інші пов’язані технології.
- Ефективні інструменти налагодження: забезпечує надійні можливості налагодження.
- Розширюваність і підтримка плагінів: дозволяє налаштовувати та розширювати за допомогою різних плагінів.
- Серед недоліків слід виділити:
- Відносно новий: у порівнянні з добре відомими IDE, Project Rider може мати менше функцій або крутішу криву навчання для деяких користувачів.

- Продуктивність: у певних конфігураціях користувачі можуть час від часу відчувати проблеми з продуктивністю.
- Вартість ліцензії: хоча Project Rider доступний за моделлю передплати, деякі користувачі можуть вважати вартість ліцензії відносно високою порівняно з іншими безкоштовними альтернативами.

Фактичний вигляд інтерфейсу можна побачити на рисунку 4.2.

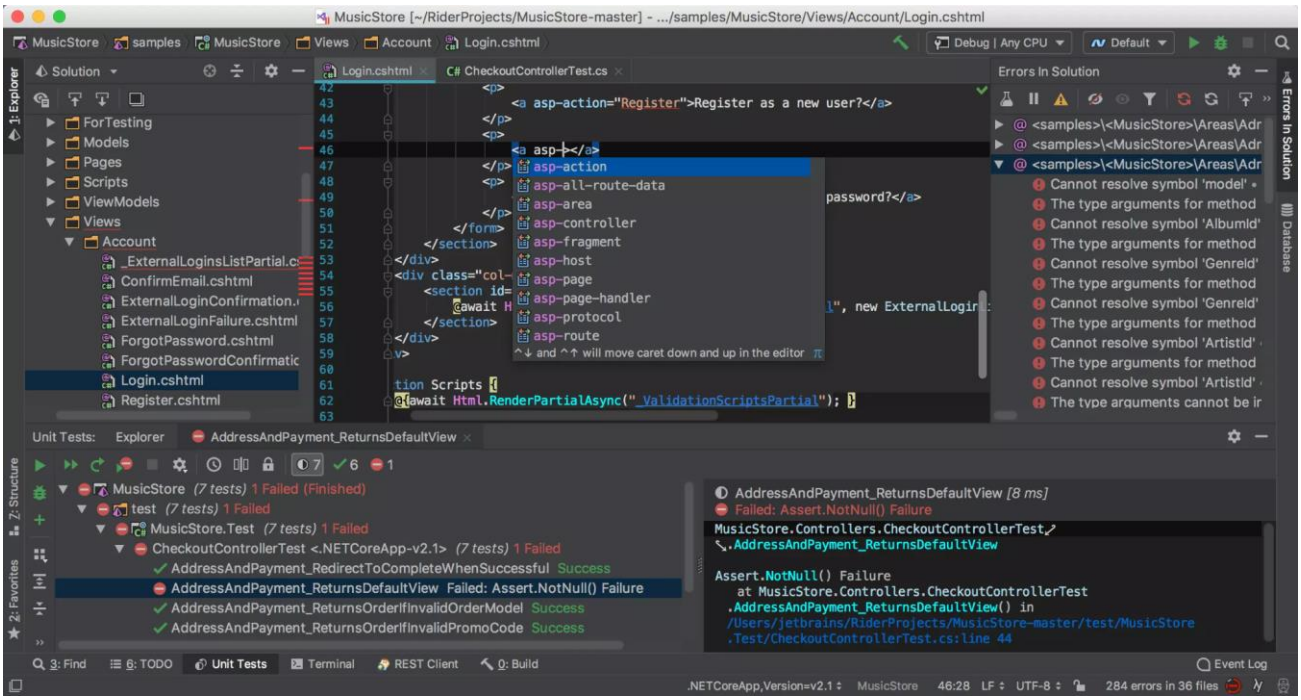

Рисунок 4.2 – Знимок екрана, на якому продемонстрований інтерфейс Project Rider

З метою полегшення розробки мовою програмування C# було створено спеціальний плагін під назвою Eclipse aCute [49]. Цей плагін дозволяє використовувати вбудований редактор C#, який інтегрований в Eclipse IDE, із підтримкою мов завдяки серверу Omni-sharp.

До переваг плагіна варто відзначити:

- Виділення синтаксису кольором для покращення читабельності коду.
- Можливість оголошення змінних, методів та класів безпосередньо з інтерфейсу.

- Можливість розробки під .NET без виходу з інтегрованого середовища розробки.
- Підтримка виконання програм, розроблених з використанням MS Test та xUnit, безпосередньо у середовищі IDE.

Серед недоліків платформи можна відзначити:

- Складність для початківців у освоєнні інтерфейсу.
- Не завжди висока якість розробленого плагіна, оскільки він розробляється спільнотою користувачів.

Приклад інтерфейсу можна переглянути на рисунку 4.3.

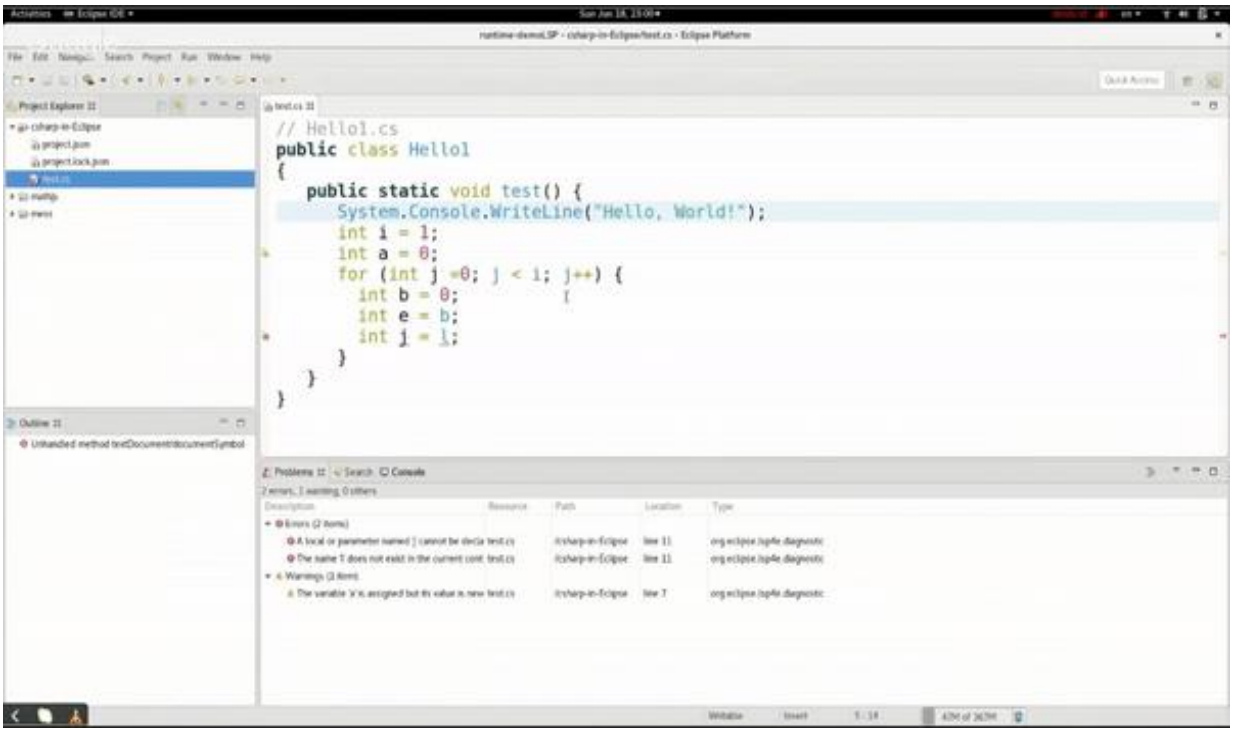

Рисунок 4.3 – Вигляд інтерфейсу Eclipse aCute з екрана ПК

З огляду на визначені вище критерії, було здійснено порівняння IDE, результати якого видно на таблиці 4.2.

Таблиця 4.2 – Порівняльний аналіз IDE на основі окреслених критеріїв

Критерій	Eclipse aCute	Project Rider	Visual Studio
Безплатність	0,5	0	0,5
Універсальність	0,5	1	1
Кросплатформність	1	1	1
Швидкість роботи	0,5	0,5	1
Відсутність завеликих вимог до продуктивності системи	0	0,5	0,5
Магазин розширень	0	0,5	1
Автодоповнення коду	0,5	0,5	1
Підсумковий результат	3	4	6

Отже, для розробки бази даних системи самообслуговування за допомогою C# Microsoft Visual Studio є одним із найбільш відповідних інтегрованих середовищ розробки (IDE). Його різноманітні версії, включаючи Community Edition, надають надійні інструменти та функції, що дозволяє розробникам проектувати, кодувати та ефективно керувати базами даних.

4.3 Програмна реалізація системи

Обрана архітектура для розробки системи самостійного касового розрахунку – Onion. Початковий етап розробки включає створення сутностей для бази даних на доменному рівні. У цьому контексті сутність представляє собою абстракцію, яка існує без конкретних дій.

В адмініструванні баз даних сутність може представляти різні об'єкти – речі, особи, місця чи об'єкти, і дані про ці об'єкти можуть бути збережені. Для візуалізації зв'язків між сутностями використовується інструмент проектування, такий як діаграма взаємодії сутностей (ERD) [50]. В адмініструванні баз даних важливо враховувати, що сутністю вважається лише те, що вимагає збереження

даних. Створення сутностей без конкретної потреби в збереженні даних може бути безсмислене.

При програмуванні баз даних розглядаються різні підходи до представлення типів сутностей та їх властивостей. Наприклад:

- Прийняття усталеної технології: цей підхід передбачає використання добре відомих і визнаних технологій баз даних, таких як MySQL, PostgreSQL, MongoDB або SQL Server.
- Створення спеціального типу бази даних, адаптованого до вимог конкретного проєкту. Цей метод дозволяє точно контролювати структуру даних та оптимізувати продуктивність відповідно до потреб програми.
- Використання комбінації обох вищезазначених методів.

Сучасні інструменти для розробки програмного забезпечення обладнані різноманітними бібліотеками, які полегшують роботу зі складними колекціями та забезпечують ефективну візуалізацію даних (React, TensorFlow, NumPy, jQuery, .NET Core, тощо). Розглянемо зв'язок між сутністю та таблицею бази даних у контексті популярної системи керування базами даних (СУБД), коли тип сутності відповідає таблиці в базі даних. Атрибути сутності мають збігатися з полями в моделі зв'язку сутності (ER), а запис у таблиці бази даних представляє екземпляр цієї сутності. По суті, це окреслює відображення між концептуальним представленням сутностей та їх конкретним представленням у вигляді таблиць і записів у системі реляційної бази даних. В даному контексті, усі сутності збігаються з тими, які були окреслені в розділі 3.4, коли мова йшла про розробку бази даних. На рисунку 4.4 чітко відображена реалізація сутностей, що ілюструє описане вище.

```
public class Customer : BaseEntity
{
    public int DiscountValue { get; set; }
    public int PersonId { get; set; }

    public Person Person { get; set; }
    public ICollection<Receipt> Receipts { get; set; }
}
```

Рисунок 4.4 – Ілюстрація фактичної реалізації сутностей «Покупець»

Після цього були створені інтерфейси для всіх репозиторіїв, а також сами репозиторії.

Репозиторій у цьому контексті – це набір з сутностями у своїй структурі, який може фільтрувати та повертати результати відповідно до вимог конкретної програми [51]. Деталі того, як сам репозиторій зберігає ці об'єкти, вважаються "деталями реалізації". На рисунку 4.5 наведено приклад розроблених інтерфейсів та репозиторіїв.

```
public interface IRepository<TEntity> where TEntity : BaseEntity
{
    Task<IEnumerable<TEntity>> GetAllAsync();

    Task<TEntity> GetByIdAsync(int id);

    Task AddAsync(TEntity entity);

    void Delete(TEntity entity);

    Task DeleteByIdAsync(int id);

    void Update(TEntity entity);
}

public class Repository<TEntity> : IRepository<TEntity> where TEntity : BaseEntity
{
    protected readonly TradeMarketDbContext _context;
    protected readonly DbSet<TEntity> _dbSet;
    public Repository(TradeMarketDbContext context)
    {
        _context = context;
        _dbSet = context.Set<TEntity>();
    }
    public async Task AddAsync(TEntity entity)
    {
        await _dbSet.AddAsync(entity);
    }
    public void Delete(TEntity entity)
    {
        _dbSet.Remove(entity);
    }
    public async Task DeleteByIdAsync(int id)
    {
        var entity = await _dbSet.FindAsync(id);
        _dbSet.Remove(entity);
    }
    public async Task<IEnumerable<TEntity>> GetAllAsync()
    {
        return await _dbSet.AsNoTracking().ToListAsync();
    }
    public async Task<TEntity> GetByIdAsync(int id)
    {
        return await _dbSet.FindAsync(id);
    }
    public void Update(TEntity entity)
    {
        _context.Entry(entity).State = EntityState.Modified;
    }
}
```

Рисунок 4.5 – Реалізація репозиторія та його інтерфейсу

Далі важливо створити всі необхідні зв'язки між таблицями у базі даних, які були описані в розділі 3.4. Функція OnModelCreating використовується для встановлення цих зв'язків у базі даних. Приклад реалізації цієї функції наведено на рисунку 4.6.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Person>()
        .HasOne<Customer>(p => p.Customer)
        .WithOne(c => c.Person)
        .HasForeignKey<Customer>(p => p.PersonId);

    modelBuilder.Entity<Receipt>()
        .HasOne<Customer>(r => r.Customer)
        .WithMany(c => c.Receipts)
        .HasForeignKey(r => r.CustomerId);

    modelBuilder.Entity<ReceiptDetail>()
        .HasOne<Receipt>(rd => rd.Receipt)
        .WithMany(r => r.ReceiptDetails)
        .HasForeignKey(rd => rd.ReceiptId);

    modelBuilder.Entity<ReceiptDetail>()
        .HasOne<Product>(rd => rd.Product)
        .WithMany(p => p.ReceiptDetails)
        .HasForeignKey(p => p.ProductId);

    modelBuilder.Entity<Product>()
        .HasOne<ProductCategory>(p => p.Category)
        .WithMany(pc => pc.Products)
        .HasForeignKey(p => p.ProductCategoryId);
}
```

Рисунок 4.6 – Встановлення зв’язків між таблицями

Щоб завершити налаштування рівня домену потрібно виконати процес міграції. Це робиться для створення бази даних саме в MS SQL Server. З цією метою допомагає Entity Framework, а саме – його функціональність. При використанні команди "add-migration" відбувається процес генерації міграції, яка відповідає за створення всіх необхідних таблиць у базі даних. Автоматично в рамках проєкту створюється тека «Міграції», що містить скрипти. Своєю чергою, ці скрипти керують усіма таблицями сутностей і їх зв’язками. Те, що являє собою міграція, проілюстровано на рисунку 4.7.

```
protected override void BuildModel(ModelBuilder modelBuilder)
{
#warning disable 612, 618
    modelBuilder
        .HasAnnotation("ProductVersion", "3.1.6")
        .HasAnnotation("Relational:MaxIdentifierLength", 128)
        .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

    modelBuilder.Entity("Data.Entities.Customer", b =>
        {
            b.Property<int>("Id")
                .ValueGeneratedOnAdd()
                .HasColumnType("int")
                .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

            b.Property<int>("DiscountValue")
                .HasColumnType("int");

            b.Property<int>("PersonId")
                .HasColumnType("int");

            b.HasKey("Id");

            b.HasIndex("PersonId")
                .IsUnique();

            b.ToTable("Customers");
        });
```

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "Persons",
        columns: table => new
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            Name = table.Column<string>(nullable: true),
            Surname = table.Column<string>(nullable: true),
            BirthDate = table.Column<DateTime>(nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Persons", x => x.Id);
        });

    migrationBuilder.CreateTable(
        name: "ProductCategories",
        columns: table => new
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            CategoryName = table.Column<string>(nullable: true)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_ProductCategories", x => x.Id);
        });
```

Рисунок 4.7 – Приклад використання міграцій при створенні бази даних

Наступним етапом є створення сервісного рівня, який зазвичай відповідає за усю логіку нашого програмного продукту. Перше, що необхідно зробити, це створити моделі для всіх сутностей, які були розроблені на доменному рівні. Це дозволить нам подальше зіставлення потрібних полів за допомогою мапера та забезпечить доступ до всієї необхідної інформації про сутності. Процес створення моделей показано на рисунку 4.8.

```
namespace Business.Models
{
    public class ReceiptModel
    {
        public int Id { get; set; }
        public int CustomerId { get; set; }
        public DateTime OperationDate { get; set; }
        public bool IsCheckedOut { get; set; }
        public ICollection<int> ReceiptDetailsIds { get; set; }
    }
}

namespace Business.Models
{
    public class ProductModel
    {
        public int Id { get; set; }
        public int ProductCategoryId { get; set; }
        public string ProductName { get; set; }
        public string CategoryName { get; set; }
        public decimal Price { get; set; }
        public ICollection<int> ReceiptDetailIds { get; set; }
    }
}
```

Рисунок 4.8 – Реалізація моделей

Наступним кроком є створення мапера для зіставлення полів. Мапінг даних – це процес встановлення відповідності між полями даних (конкретними елементами джерела або самим джерелом) та пов'язаними з ними полями даних у іншому місці призначення [52]. Іншими словами, йдеться про асоціювання та зіставлення полів даних з одного місця в інше, щоб забезпечити точну та синхронізовану передачу даних між різними об’єктами чи системами. Мапінг даних є цінним, оскільки він полегшує перетворення даних з одного формату в

інший. Його корисність полягає в забезпеченні бездоганної інтеграції даних, де він дозволяє структурувати, трансформувати та інтегрувати дані з різних джерел в єдиний формат, забезпечуючи сумісність і послідовність у різних системах або програмах.та розуміти величезні обсяги інформації, що зберігаються у різних місцях.

Мапування даних – це важливий процес, який допомагає узгодити дані з різних джерел, щоб зробити їх придатними для використання та значущими [53]. Після нього стає легше отримувати релевантну інформацію та ідеї, що дозволяє ефективно аналізувати та використовувати дані: функція AutomapperProfile, зображена на малюнку 4.9, ілюструє цей процес, показуючи, як корелюються поля між моделями, допомагаючи в перетворенні та використанні даних для конкретних цілей.

```
public AutomapperProfile()
{
    CreateMap<Receipt, ReceiptModel>()
        .ForMember(rm => rm.ReceiptDetailsIds, r => r.MapFrom(x => x.ReceiptDetails.Select(rd => rd.Id)))
        .ReverseMap();

    CreateMap<Product, ProductModel>()
        .ForMember(pm => pm.ReceiptDetailIds, p => p.MapFrom(x => x.ReceiptDetails.Select(rd => rd.Id)))
        .ForMember(pm => pm.CategoryName, p => p.MapFrom(x => x.Category.CategoryName))
        .ReverseMap();

    CreateMap<ReceiptDetail, ReceiptDetailModel>()
        .ForMember(rd => rd.ReceiptId, r => r.MapFrom(x => x.ReceiptId))
        .ReverseMap();

    CreateMap<Customer, CustomerModel>()
        .ForMember(cm => cm.Name, c => c.MapFrom(x => x.Person.Name))
        .ForMember(cm => cm.Surname, c => c.MapFrom(x => x.Person.Surname))
        .ForMember(cm => cm.BirthDate, c => c.MapFrom(x => x.Person.BirthDate))
        .ForMember(cm => cm.ReceiptsIds, c => c.MapFrom(x => x.Receipts.Select(r => r.Id)))
        .ReverseMap();

    CreateMap<ProductCategory, ProductCategoryModel>()
        .ForMember(pm => pm.ProductIds, pc => pc.MapFrom(x => x.Products.Select(p => p.Id)))
        .ReverseMap();
}
```

Рисунок 4.9 – Вигляд процесу мапування

Процес відображення та кореляції полів між моделями справді важливий. Після завершення відображення та асоціації полів наступним кроком є створення

служб, які інкапсулюють логіку та функції. Ці служби відповідають за виконання різних дій, таких як додавання різноманітної інформації, пошук даних і маніпуляції. Малюнок 4.10 містить приклад функції AddAsync, що ілюструє її роль у додаванні нового продукту до бази даних. Ця функція є частиною служб, відповідальних за керування та обробку операцій з даними в програмі.

```
private readonly IUnitOfWork _uow;
private readonly IMapper _mapper;
private readonly ProductServiceValidation _productValidation = new ProductServiceValidation();

public ProductService(IUnitOfWork uow, IMapper mapper)
{
    _uow = uow;
    _mapper = mapper;
}
public async Task AddAsync(ProductModel model)
{
    _productValidation.ProductValidation(model);
    var result = _mapper.Map<ProductModel, Product>(model);
    await _uow.ProductRepository.AddAsync(result);
    await _uow.SaveAsync();
    _mapper.Map(result, model);
}
```

Рисунок 4.10 – Демонстрація процесу додавання інформації до бази даних

Цей процес включає функцію, яка обробляє додавання нового продукту до бази даних. Він використовує картограф для підключення полів продукту до відповідних областей у базі даних. Перед додаванням продукту виконується етап під назвою ProductValidation, який забезпечує дійсність даних, які збираються додати. Ця перевірка допомагає підтримувати якість і точність інформації. Приклад цього процесу перевірки показано на малюнку 4.11.

```
public void ProductValidation(ProductModel model)
{
    if (model == null)
    {
        throw new MarketException("Model is null!");
    }
    if (model.ProductName == null || model.ProductName == "")
    {
        throw new MarketException("Product name is null!");
    }
    if (model.Price <= 1)
    {
        throw new MarketException("Incorrect price!");
    }
}
```

Рисунок 4.11 – Приклад перевірки (validation) вхідної інформації

Подібна перевірка є наявною майже у всіх сервісах і гарантують безпеку бази даних, оскільки при неправильній вхідній інформації процес перевірки (validation) відобразить exception (виключення).

Наступним прикладом є додавання просканованого товару до квитанції споживача (рисунок 4.12).

```
public async Task AddProductAsync(int productId, int receiptId, int quantity)
{
    var receipt = await _uow.ReceiptRepository.GetByIdWithDetailsAsync(receiptId);
    _receiptValidation.ReceiptValidation(receipt);
    if(receipt.ReceiptDetails == null || !receipt.ReceiptDetails.Any(x => x.ProductId == productId))
    {
        var product = await _uow.ProductRepository.GetByIdAsync(productId);
        _receiptValidation.ReceiptValidation(product);
        var receiptDetail = new ReceiptDetail()
        {
            ReceiptId = receiptId,
            ProductId = product.Id,
            Quantity = quantity,
            UnitPrice = product.Price,
            DiscountUnitPrice = product.Price - ((product.Price * receipt.Customer.DiscountValue) / 100)
        };
        await _uow.ReceiptDetailRepository.AddAsync(receiptDetail);
    }
    else
    {
        var receiptDetail = receipt.ReceiptDetails.FirstOrDefault(x => x.ProductId == productId);
        if(receiptDetail != null)
        {
            receiptDetail.Quantity += quantity;
            receiptDetail.UnitPrice = receiptDetail.UnitPrice * quantity;
            receiptDetail.DiscountUnitPrice = receiptDetail.DiscountUnitPrice * quantity;
        }
    }
    await _uow.SaveAsync();
}
```

Рисунок 4.12 – Вигляд реалізації додавання просканованого товару до квитанції

Цей процес передбачає додавання інформації про засканований товар до квитанції. Функція отримує детальну інформацію про позицію та кількість, яку потрібно додати до квитанції. Потім відбувається перегляд всіх продуктів в базі даних і відшукується потрібний. Далі перевіряється, чи цей товар уже наявний в квитанції. Якщо так, до наявного запису додається необхідна кількість цього товару. Однак, якщо товару ще немає у квитанції, для нього створюється новий запис, а кількість, указана покупцем, призначається цьому товару у квитанції.

Окрім зазначеного на рисунку 4.13 можна побачити розроблену функцію генерації знижок для користувачів.

```
public async Task GenerateDiscount(int productCount, int customerId)
{
    if(productCount == null || customerId == null)
    {
        throw new Exception();
    }
    var receipts = await _uow.ReceiptRepository.GetAllWithDetailsAsync();
    var customer = await _uow.CustomerRepository.GetByIdWithDetailsAsync(customerId);
    var result = receipts.Where(x => x.CustomerId == customerId)
                         .SelectMany(x => x.ReceiptDetails)
                         .GroupBy(x => x.Product, x => x.Quantity,
                         (prod, quant) => new { Product = prod, Amount = quant.Sum() })
                         .OrderByDescending(x => x.Amount).Select(x => x.Product).Take(productCount).ToList();
    customer.DiscountValue = result.FirstOrDefault(x => x.Price) % 15;
    return customer;
}
```

Рисунок 4.13 – Вигляд процесу генерації (створення) знижок

Даний процес має декілька етапів, які можна структурувати наступним чином:

- Отримання даних: функція отримує інформацію про кількість товарів і унікальний ідентифікатор клієнта.
- Деталі клієнта та отримання замовлень: алгоритм отримує інформацію про клієнта та отримує всі його замовлення з бази даних.
- Ідентифікація популярних товарів: зі списку замовлень алгоритм визначає найпопулярніші товари. Ці елементи визначаються на основі їх частоти в замовленнях.
- Застосування знижок: на визначені популярні товари діють знижки або спеціальні пропозиції.

Після цього відбувається перехід до рівня інфраструктури:

- Рівень інфраструктури: цей рівень включає набір контролерів, які полегшують взаємодію між веб-програмою та протоколами HTTP.
- Операції контролера: контролери отримують запити у формі об’єктів HttpRequestMessage, обробляють ці запити за допомогою методів служби та повертають оброблені результати як об’єкти HttpResponseMessage, які служать відповідями.

Приклади контролерів продемонстровані на малюнках 4.14 і 4.15.

Цей процес передбачає збір і аналіз даних про замовлення клієнтів, визначення популярних товарів і застосування знижок на ці товари. Крім того, тут описуються компоненти інфраструктури, які обробляють взаємодію між веб-програмою та протоколами HTTP через контролери.

```
[HttpPost]
public async Task<ActionResult> AddProduct([FromBody] ProductModel value)
{
    try
    {
        await _productService.AddAsync(value);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
    return CreatedAtAction(nameof(AddProduct), value);
}
```

Рисунок 4.14 – Контролер з функцією AddProduct (додавання нового товару)

```
[HttpPost("categories")]
public async Task<ActionResult> AddCategory([FromBody] ProductCategoryModel value)
{
    try
    {
        await _productService.AddCategoryAsync(value);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
    return CreatedAtAction(nameof(AddCategory), value);
}
```

Рисунок 4.15 – Контролер з функцією AddCategory (додавання нової категорії)

Повний лістинг програмного коду наведено у додатку В.

Таким чином, у підпункті було показано етап створення програмного забезпечення на основі обраної архітектури Onion. Початковий етап розробки передбачав створення сутностей на доменному рівні, які є абстракціями та відображають об'єкти або дані, які мають бути збережені в базі даних. Використання інструментів моделювання, таких як діаграми взаємодії сутностей, дозволило наочно відобразити зв'язки між ними. Процес адміністрування баз даних передбачав створення репозиторіїв, встановлення зв'язків між таблицями, використання міграцій для створення бази даних, реалізацію моделей та процес мапування. Додавання інформації до бази даних, перевірка вхідних даних, обробка просканованих товарів, генерація знижок та функції контролерів для додавання нових товарів чи категорій – усе це було ілюстровано через відповідні графічні приклади. Результатом цього етапу є реалізація важливих функцій системи самостійного касового розрахунку, що забезпечує зручність взаємодії користувача з системою. Подальші кроки передбачають удосконалення та розвиток цих функцій для забезпечення надійності, продуктивності та зручності використання програмного продукту.

4.4 Тестування програмного забезпечення

Тестування програмного забезпечення стосується процесу оцінювання та перевірки програмного забезпечення для виявлення будь-яких розбіжностей між очікуваними та фактичними результатами. Тестування може виконуватися як вручну, так і за допомогою автоматизованих інструментів, залежно від вимог та обставин [54].

Існують різні підходи до класифікації тестування програмного забезпечення, такі як «Біла скринька», «Чорна скринька», та «Сірий ящик». Вони визначаються доступом до вихідного коду та рівнем знань про внутрішню реалізацію системи.

У даній роботі визначено три основні типи тестування програмного забезпечення: функціональне, нефункціональне та тестування на сумісність. Кожен з цих типів має свою мету та область застосування.

Функціональне тестування спрямоване на перевірку того, чи працює кожна функція програми згідно з вимогами та документацією. Елементи функціонального тестування включають підготовку тестових даних, виконання тестів, аналіз результатів та порівняння їх з очікуваними.

Нефункціональне тестування охоплює такі аспекти, як продуктивність, навантаження, стрес, зручність використання та сумісність. До нефункціонального тестування входять також тестування продуктивності, юзабіліті та сумісності.

Тестування на сумісність перевіряє функціональність програми в різних апаратних і програмних середовищах, забезпечуючи її коректну роботу в різних умовах.

Таким чином, в процесі розробки програми важливим етапом є тестування, яке включає в себе різноманітні види тестів для перевірки різних аспектів програмного забезпечення.

Модульні тести відіграють ключову роль у розробці програмного забезпечення, особливо в таких системах, як касові системи самообслуговування.

Ці тести ретельно розроблені для вивчення окремих компонентів або блоків системи окремо, щоб переконатися, що вони працюють належним чином. Для касової системи самообслуговування ці тести мають величезне значення, оскільки вони допомагають перевірити точність, функціональність і надійність різних критичних аспектів. Приклади тестів націлених на оцінювання роботи алгоритмів продемонстровані на рисунках 4.16 – 4.18.

```
[Test]
public async Task CustomerService_AddAsync_AddsModel()
{
    //arrange
    var mockUnitOfWork = new Mock<IUnitOfWork>();
    mockUnitOfWork.Setup(m => m.CustomerRepository.AddAsync(It.IsAny<Customer>()));

    var customerService = new CustomerService(mockUnitOfWork.Object, UnitTestHelper.CreateMapperProfile());
    var customer = GetTestCustomerModels.First();

    //act
    await customerService.AddAsync(customer);

    //assert
    mockUnitOfWork.Verify(x => x.CustomerRepository.AddAsync(It.Is<Customer>(x =>
                    x.Id == customer.Id && x.DiscountValue == customer.DiscountValue &&
                    x.Person.Surname == customer.Surname && x.Person.Name == customer.Name &&
                    x.Person.BirthDate == customer.BirthDate)), Times.Once);
    mockUnitOfWork.Verify(x => x.SaveAsync(), Times.Once);
}
```

Рисунок 4.16 – Вигляд процесу тестування додавання нового юзераа

```
[Test]
public async Task ReceiptService_AddAsync_AddsReceipt()
{
    //arrange
    var mockUnitOfWork = new Mock<IUnitOfWork>();
    mockUnitOfWork.Setup(m => m.ReceiptRepository.AddAsync(It.IsAny<Receipt>()));

    var receiptService = new ReceiptService(mockUnitOfWork.Object, UnitTestHelper.CreateMapperProfile());
    var receipt = GetTestReceiptsModels.First();

    //act
    await receiptService.AddAsync(receipt);

    //assert
    mockUnitOfWork.Verify(x => x.ReceiptRepository.AddAsync(It.Is<Receipt>(c => c.Id == receipt.Id)), Times.Once);
    mockUnitOfWork.Verify(x => x.SaveAsync(), Times.Once);
}
```

Рисунок 4.17 – Вигляд процесу тестування додавання нового замовлення

```
[TestCase(1)]
[TestCase(2)]
public async Task ProductService_DeleteAsync_DeletesProduct(int id)
{
    //arrange
    var mockUnitOfWork = new Mock<IUnitOfWork>();
    mockUnitOfWork.Setup(m => m.ProductRepository.DeleteByIdAsync(It.IsAny<int>()));
    var productService = new ProductService(mockUnitOfWork.Object, UnitTestHelper.CreateMapperProfile());

    //act
    await productService.DeleteAsync(id);

    //assert
    mockUnitOfWork.Verify(x => x.ProductRepository.DeleteByIdAsync(id), Times.Once);
    mockUnitOfWork.Verify(x => x.SaveAsync(), Times.Once);
}
```

Рисунок 4.18 – Вигляд процесу тестування видалення продукту, що вже присутній

XUnit.net та Фреймворк Moq є прикладами інструментів, які були використані для тестування коректності алгоритмів. xUnit.net – це платформа тестування з відкритим вихідним кодом, яка використовується для модульного тестування різних програм у .NET. Вона дотримується принципів тестування xUnit і забезпечує платформу для створення автоматизованих тестів, зокрема модульних тестів, для програм .NET. Ця структура працює на основі концепції розробки, керованої тестуванням (TDD), що дозволяє розробникам писати та виконувати тести для перевірки поведінки окремих модулів або компонентів їх коду. xUnit.net пропонує чисту та розширювану архітектуру, яка спрощує процес написання, організації та виконання тестів.

З метою протестувати алгоритми та імітації репозиторія був використаний ще один фреймворк під назвою Moq, що було необхідним кроком. Moq – це популярна імітаційна структура для .NET, яка дозволяє розробникам створювати та використовувати макетні об’єкти під час модульного тестування. Він спеціально розроблений, щоб полегшити створення фіктивних реалізацій інтерфейсів і класів, щоб імітувати поведінку та взаємодію в системі, що тестується.

Загалом двадцять чотири тести було створено, які мали на меті перевірити коректність роботи алгоритмів, що були розроблені. На рисунку 4.19 можна

побачити результати наприкінці тестування. Цей результат допомагає підтвердити, що розроблені алгоритми працюють вірно і відповідають очікуванням, що сприяє високій якості та надійності програмного продукту.

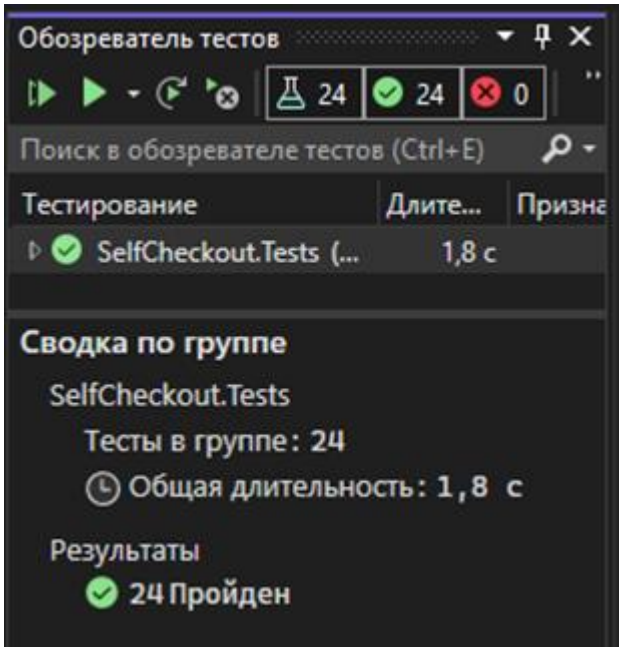

Рисунок 4.19 – Знімок екрана, що демонструє результат перевірки

У якості підсумку слід зазначити, що таким чином було успішно пройдено тестування системи самостійного касового розрахунку. Для валідації правильності функціонування алгоритмів було розроблено та виконано 24 тести, використовуючи фреймворки XUnit.net та Moq Framework.

Також для комфортного використання системи потрібно розробити інструкцію користувача.

Для використання системи самостійного касового розрахунку користувачеві потрібно обрати необхідні товари та підійти до каси самообслуговування. Головна сторінка системи пропонує користувачу вибрати між авторизацією та безпосереднім скануванням своїх продуктів. Приклад головного вікна зображено на рисунку 4.20.

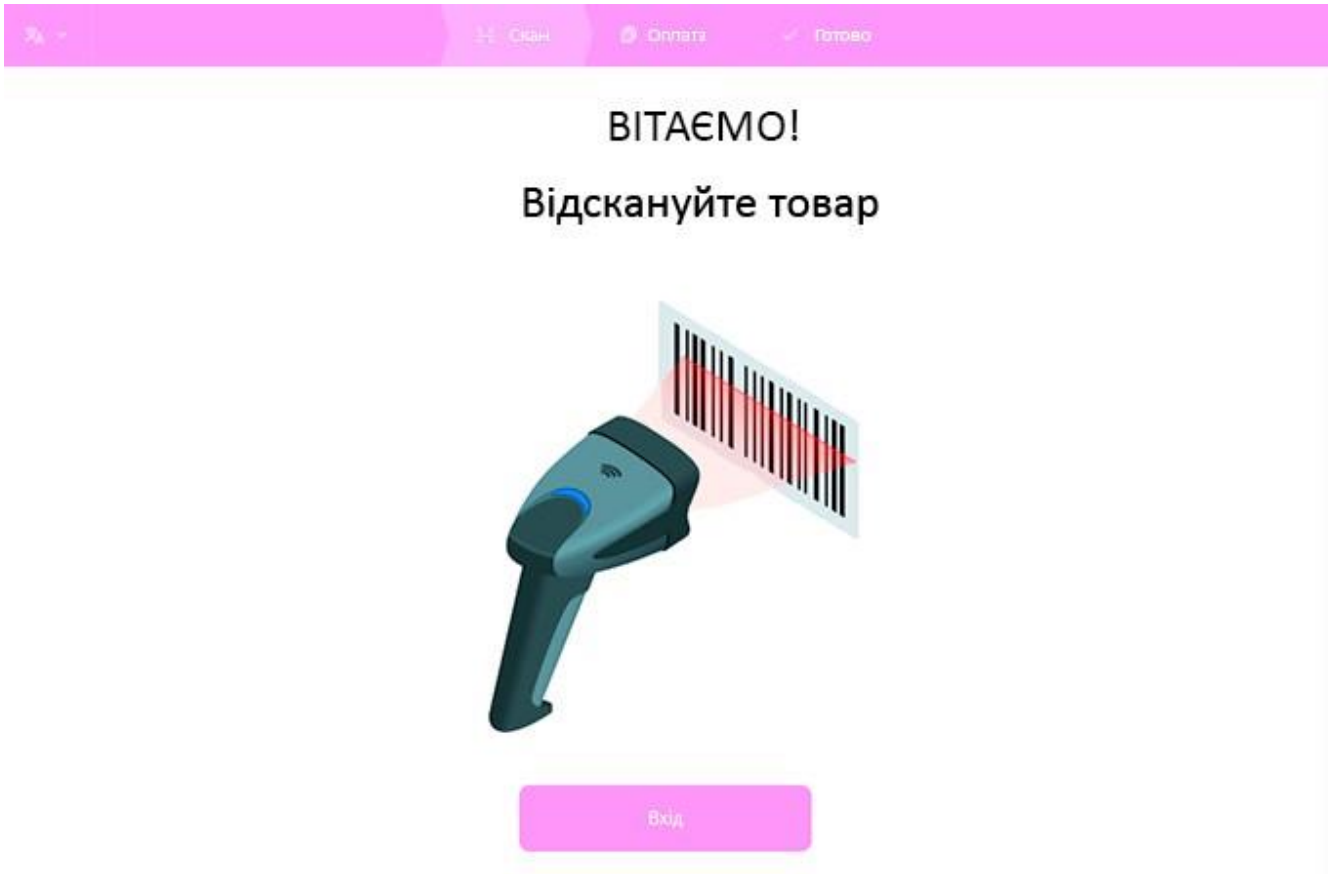

Рисунок 4.20 – Вигляд головного вікна (приклад)

Подальший алгоритм взаємодії з системою виглядатиме наступним чином:

1. Користувач натискає кнопку «Увійти».
2. Відкриється вікно авторизації, у якому користувачеві пропонується ввести номер телефону та пароль для доступу до свого облікового запису в системі.
3. Система надсилає запит до бази даних для порівняння введених даних користувача із записами системи.
4. Якщо дані правильні, користувач може сканувати свої товари під своїм обліковим записом.
5. Алгоритми отримують інформацію про придбані товари та формують знижки.
6. Крім того, якщо користувач не має облікового запису, він може зареєструватися.

Приклад процесу авторизації показано на рисунку 4.21.

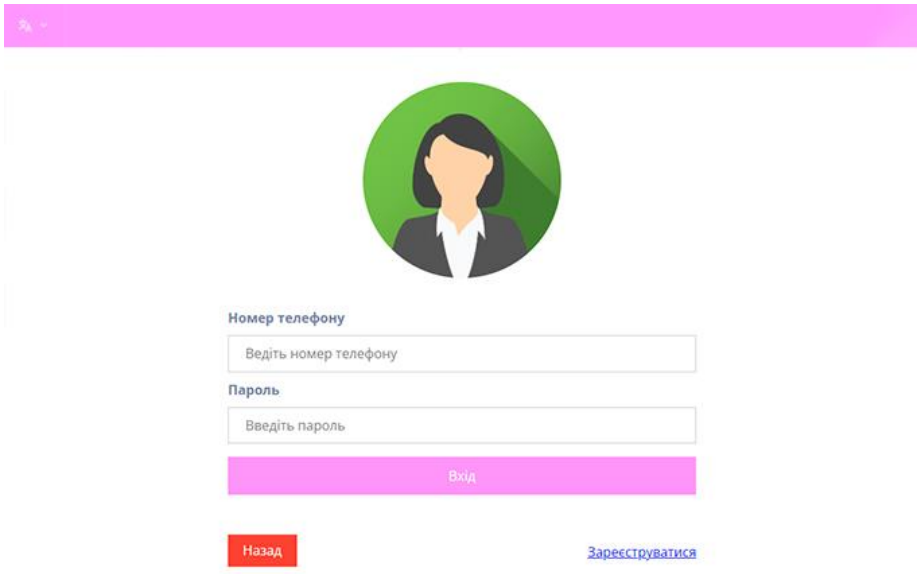

Рисунок 4.21 – Вигляд вікна з авторизацією

Коли користувач пройшов етап авторизації та сканування товарів для нього стає доступним віконце, де наявний перелік товарів та їхньою загальною вартістю. Засканові товари є доступними для перегляду поряд з сумами під кожним з них. Також доступне вікно з вибором зручного способу сплатити кошти за товар. Вигляд вікна з товарами можна побачити на рисунку 4.22.

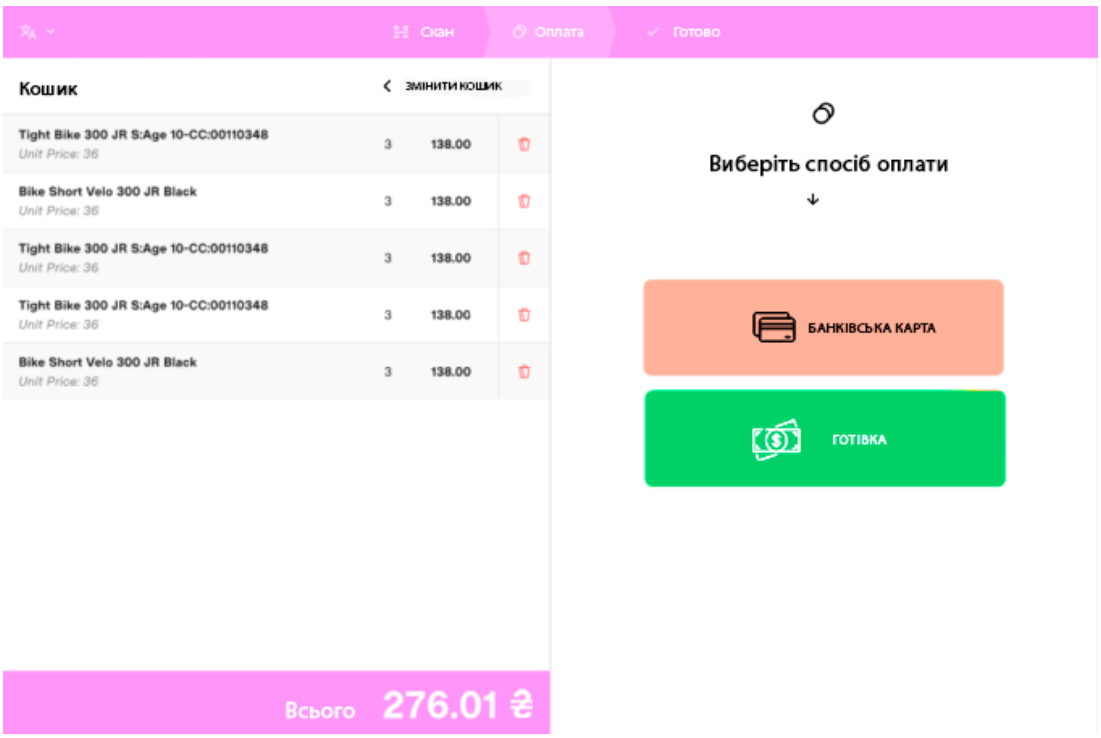

Рисунок 4.22 – Вікно з товарами

Далі буде вікно, де система перевірить характеристики покупця для можливості продажу обмежених товарів. Якщо покупець відповідає вимогам для отримання обмеженого товару, він матиме можливість здійснити покупку цього товару. Якщо покупець не відповідає вимогам для придбання обмежених товарів, система може запропонувати користувачеві видалити ці товари з кошика перед завершенням оплати. Після цього користувач може внести необхідні зміни та продовжити оплату за решту товарів. Приклад продемстровано на рисунку 4.23.

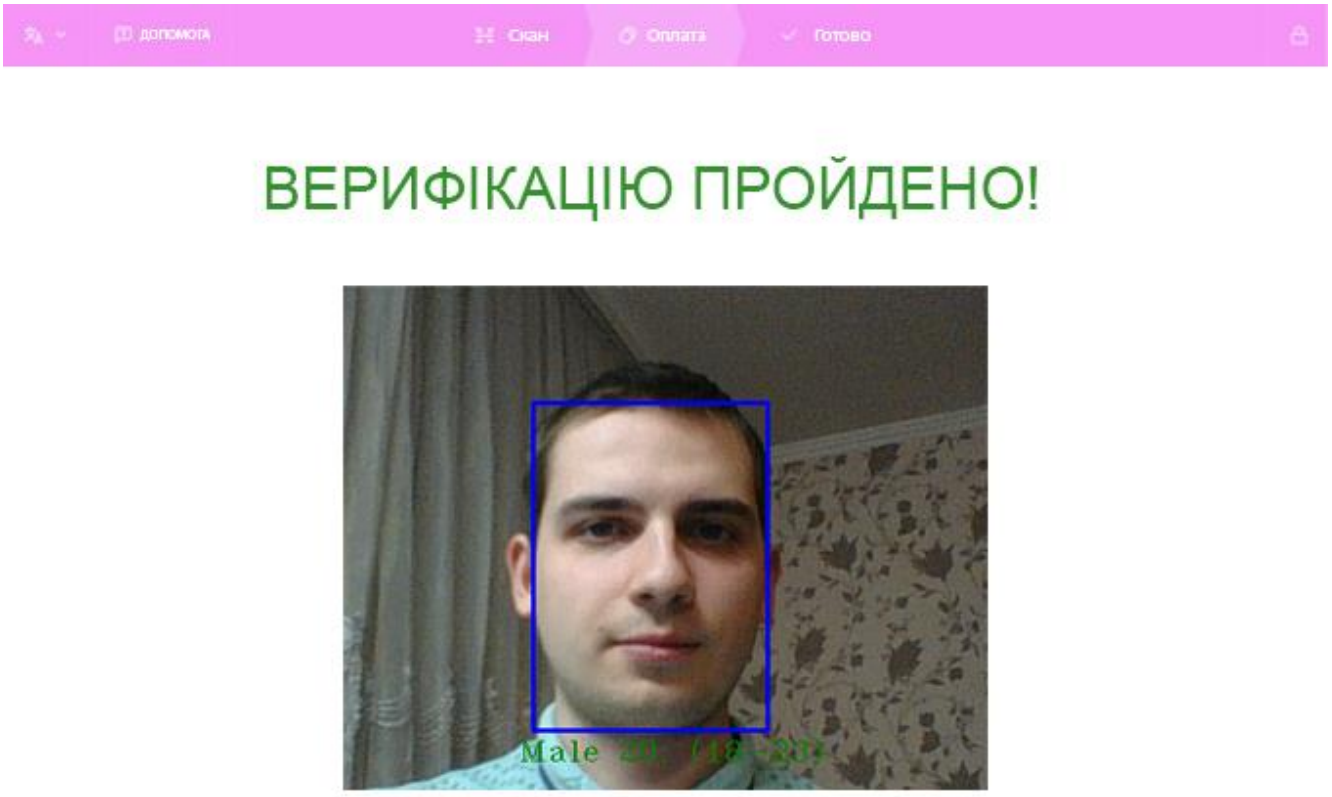

Рисунок 4.23 – Приклад вікна перевірки характеристик покупця

Безпосередньо після того, як користувач обере бажаний спосіб оплати, стане доступним вікно зі здійсненням транзакції. На цьому етапі стає можливим розрахуватися за товари. Одразу після цього на екрані будуть доступні дані з повідомленням про успішну транзакцію (див. Рисунок 4.24-4.25).

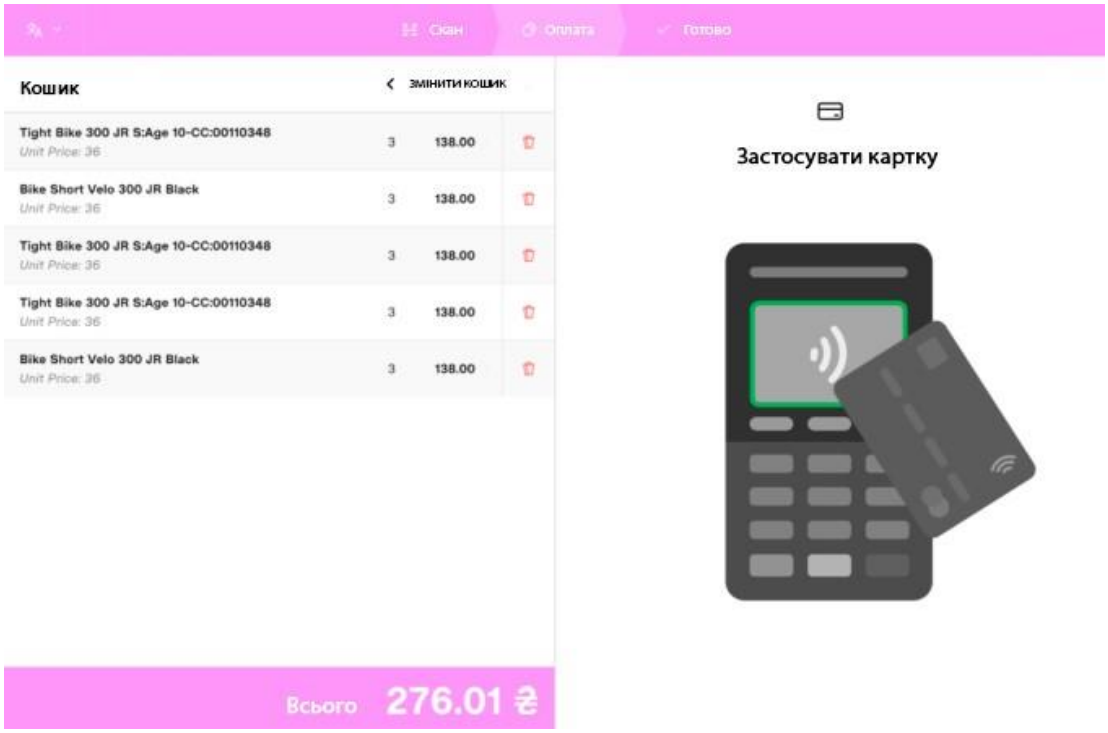

Рисунок 4.24 – Приклад вікна з оплатою

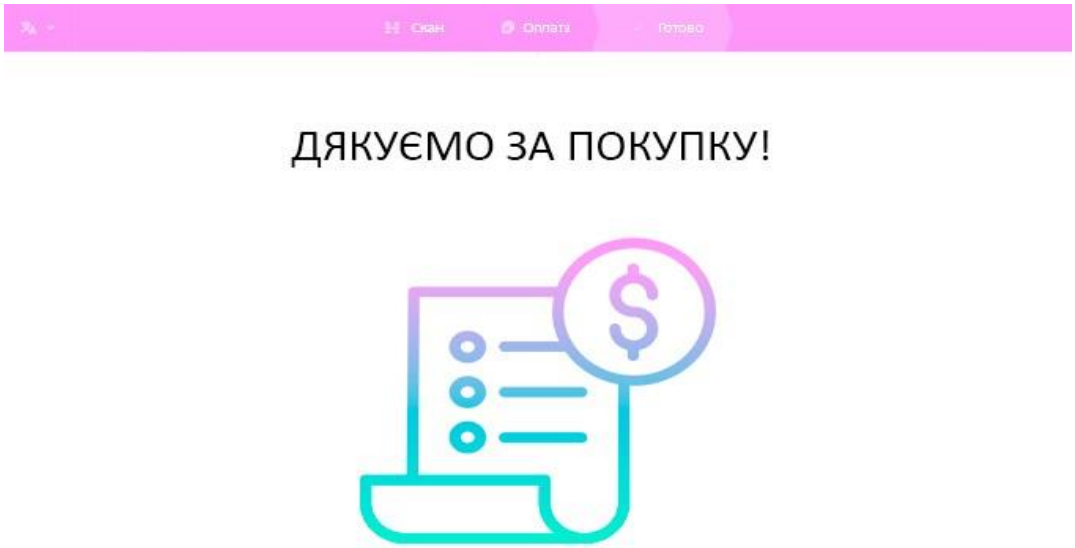

Рисунок 4.9 – Результат транзакції

Разом з розробкою системи самостійного касового розрахунку була підготовлена інструкція для користувача. Ця інструкція детально описує всі необхідні кроки для використання функціоналу додатку. Основні етапи, що розглядаються в інструкції, включають сканування товарів, процес реєстрації та

авторизації користувачів, додавання товарів до кошика, перевірку характеристик покупця, вибір зручного способу оплати та безпосереднє завершення оплати з успішною транзакцією.

Ця інструкція створена з метою забезпечення користувачам чіткого та зрозумілого орієнтування під час використання програмного продутку. Вона слугує довідковим матеріалом для того, щоб кожен користувач міг максимально ефективно та комфортно використовувати функціональні можливості додатку під час покупок та оплати товарів.

4.5 Висновки

Під час роботи над четвертим розділом було досліджено та з’ясовано кілька критичних аспектів розробки системи. По-перше, досліджено, порівняно та обґрунтовано вибір мови програмування, в кінцевому підсумку визначивши C# як найбільш відповідну мову в контексті мети, яка полягає в розробці програмних компонент для системи самостійного касового розрахунку. Мові програмування C# було віддано перевагу через її надійність, універсальність і можливості інтеграції в системну архітектуру.

По-друге, було обговорено вибір середовища розробки (IDE), яким стала програма MS Visual Studio. Це рішення було підкріплено комплексним набором інструментів Visual Studio, його бездоганною інтеграцією з C# та підтримкою різних парадигм розробки.

Крім того, розділ заглибився в програмну реалізацію системи, ілюструючи практичну реалізацію та функціональність програмних компонентів.

В кінцевому результаті була розроблена інструкція користувача, яка крок за кроком розглядає використання функціонала системи самостійного касового розрахунку, включаючи сканування товарів, реєстрацію та авторизацію користувачів, роботу з кошиком, перевірку характеристик покупця, вибір способу оплати та завершення транзакції.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Система самостійного касового розрахунку з визначенням характеристик покупця» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Система самостійного касового розрахунку з визначенням характеристик покупця» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [55].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4

2. Ринкові переваги (наявність аналогів)	2	2	3
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	33	35	35
Середньоарифметична сума балів $СБ_c$	34,3		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [55].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система самостійного касового розрахунку з визначенням характеристик покупця» становить 34,3 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [56]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad , \tag{5.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{ai}} \quad , \tag{5.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *i*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}} \quad ; \tag{5.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 5.4.

Таблиця 5.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірювання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість камер	Шт.	0	1	2	0,2
Середня швидкість обслуговування	С	20	15	1,33	0,25
Розпізнавання штрих-кодів	Скан/с	60	120	0,5	0,1
Сенсорний екран	Дюйми	15,6	19	1,22	0,25
Розмір(висота, ширина, довжина)	Мм	1590x1198x1593	1480x1112x1523	1,5	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2 \cdot 0{,}2 + 1{,}33 \cdot 0{,}25 + 0{,}5 \cdot 0{,}1 + 1{,}22 \cdot 0{,}25 + 1{,}5 \cdot 0{,}2 = 1{,}39.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,39 рази.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Система самостійного касового розрахунку з визначенням характеристик покупця», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [55]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 16950{,}00 \cdot 56 / 22 = 43145{,}45 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-технічної роботи	16950,00	770,45	56	43145,45
Інженер-розробник автоматизованих систем	16120,00	732,73	46	33705,45
Консультант (старший менеджер торгівельної мережі самообслуговування)	16000,00	727,27	10	7272,73
Провідний фахівець	9000,00	409,09	46	18818,18
Всього				102941,82

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система самостійного касового розрахунку з визначенням характеристик покупця» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \quad , \tag{5.5}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \quad , \tag{5.6}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [55];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}70 \cdot 1{,}35 / (22 \cdot 8) = 87{,}37$ *грн.*
$З_{р1} = 87{,}37 \cdot 3{,}00 = 262{,}10$ *грн.*

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Вхідний контроль компонентів системи	3,00	5	1,70	87,37	262,10
Інсталяція програмного забезпечення розробки (моделювання)	6,50	5	1,70	87,37	567,88
Монтаж електронної обчислювальної системи	6,20	3	1,35	69,38	430,15
Монтаж компонентів експериментального пристрою каси самообслуговування	8,10	4	1,50	77,09	624,41
Налагодження безпровідного зв'язку	2,00	4	1,50	77,09	154,18
Налагодження системи самостійного касового розрахунку	4,00	5	1,70	87,37	349,47
Підготовка робочого місця дослідника-розробника автоматизованих систем	7,20	2	1,10	56,53	407,03
Всього					2795,21

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (102941,82 + 2795,21) · 11 / 100% = 11631,07 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (102941,82 + 2795,21 + 11631,07) · 22 / 100% = 25820,98 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Система самостійного касового розрахунку з визначенням характеристик покупця».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_{вj} Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц \quad , \quad (5.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1$ = 2 · 199,00 · 1,04 - 0 · 0 = 413,92 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	199,00	2	-	-	413,92
Папір для заміток (А5)	111,00	4	-	-	461,76
Органайзер офісний	202,00	3	-	-	630,24
Начиння канцелярське	185,00	3	-	-	577,20
Картридж для принтера	1129,00	1	-	-	1174,16
Диск оптичний	25,50	4	-	-	106,08
USB-пам'ять	169,00	2	-	-	351,52
Хлорне залізо	210,00	0,100	-	-	21,84
Дріт монтажний	16,00	1,200	-	-	19,97
Лак УР-231	285,00	0,150	-	-	44,46
Фарба	260,00	0,150	-	-	40,56
Герметик	380,00	0,120	-	-	47,42
Припій ПОС-61	580,00	0,060	-	-	36,19
Флюс	385,00	0,080	-	-	32,03
Всього					3957,36

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Система самостійного касового розрахунку з визначенням характеристик покупця», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 1 \cdot 2450,00 \cdot 1,04 = 2548,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Камера високої роздільної здатності	1	2450,00	2548,00
Сенсорний екран	1	4650,00	4836,00
Сканер штрих-коду	1	560,00	582,40
Корпус каси	1	5620,00	5844,80
Монтажний комплект	2	320,00	665,60
Модуль безпровідного зв'язку	1	1600,00	1664,00
Принтер термодруку	1	1280,00	1331,20
Ваги автоматичні вбудовані	1	2450,00	2548,00
Термінал банківський безготівкової оплати	1	5800,00	6032,00
Всього			26052,00

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 5820,00 \cdot 1 \cdot 1,04 = 6052,80$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Термінал касовий	1	5820,00	6052,80
Сервер інформаційної підтримки	1	25840,00	26873,60
Всього			32926,40

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \qquad (5.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 2899,00 \cdot 1 \cdot 1,04 = 3014,96$ грн.

Отримані результати зведемо до таблиці:

Таблиця 5.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Програмне забезпечення для системи самостійного касового розрахунку	1	2899,00	3014,96
ОС Windows	1	4299,00	4470,96
Прикладний пакет Microsoft Office	1	5240,00	5449,60
Прикладний пакет моделювання процесів MatLab	1	5411,00	5627,44
Всього			18562,96

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (24999{,}00 \cdot 1) / (2 \cdot 12) = 1041{,}63$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер ПЕОМ розробника автоматизованих систем	24999,00	2	1	1041,63
Обчислювальний комплекс та комп'ютеризована система проектування	38999,00	2	1	1624,96
Робоче місце розробника	10250,00	5	1	170,83
Пристрій графічного виводу інформації	8750,00	4	1	182,29
Оргтехніка	8900,00	4	1	185,42

Приміщення лабораторії	346000,00	25	1	1153,33
Монтажне обладнання	6700,00	5	1	111,67
Станція паяльна	7505,00	5	1	125,08
Всього				4595,21

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7,50$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,30 \cdot 350,0 \cdot 7,50 \cdot 0,95 / 0,97 = 787,50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер ПЕОМ розробника автоматизованих систем	0,30	350,0	787,50
Обчислювальний комплекс та комп'ютеризована система проектування	0,42	400,0	1260,00
Робоче місце розробника	0,08	350,0	210,00
Пристрій графічного виводу інформації	0,22	3,5	5,78
Оргтехніка	0,45	1,2	4,05
Монтажне обладнання	0,15	40,0	45,00
Станція паяльна	0,10	25,0	18,75
Термінал касовий	0,20	100,0	150,00

Сервер інформаційної підтримки	0,25	100,0	187,50
Всього			2668,58

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Система самостійного касового розрахунку з визначенням характеристик покупця» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (5.15)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (102941,82 + 2795,21) · 20 / 100% = 21147,41 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{H_{сп}}{100\%}, \quad (5.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (102941,82 + 2795,21) · 30 / 100% = 31721,11 грн.

5.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 57%.

$I_в$ = (102941,82 + 2795,21) · 57 / 100% = 60270,11 грн.

5.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 105%.

$В_{нзв}$ = (102941,82 + 2795,21) · 105 / 100% = 111023,88 грн.

Витрати на проведення науково-дослідної роботи на тему «Система самостійного касового розрахунку з визначенням характеристик покупця» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.19)$$

$B_{заг}$= 102941,82 +2795,21 +11631,07 +25820,98 +3957,36 +26052,00 +32926,40 + 18562,96 + 4595,21 +2668,58 +21147,41 +31721,11 +60270,11 +111023,88 = =456114,09 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.20)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 456114,09 / 0,95 = 480120,10 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Система самостійного касового розрахунку з визначенням характеристик покупця» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів пристрою, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	650	1250	1000	750

N – кількість споживачів які використовували аналогічний пристрій у році до впровадження результатів нової науково-технічної розробки, приймемо 9500 осіб;

$Ц_б$ – вартість пристрою у році до впровадження результатів розробки, приймемо 55000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості пристрою від впровадження результатів науково-технічної розробки, приймемо 1312,50 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [55]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (1312,50·9500,00+56312,50·650)·0,83·0,3·(1-0,18/100%)=10019495,44 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ = (1312,50·9500,00+56312,50·1900)·0,83·0,3·(1-0,18/100%)=24391853,25 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ = (1312,50·9500,00+56312,50·2900)·0,83·0,3·(1-0,18/100%)=35889739,50 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(1312,50·9500,00+56312,50·3650)·0,83·0,3·(1-0,18/100%)=44513154,19 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,14;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =10019495,44/(1+0,14)[1]+24391853,25/(1+0,14)[2]+35889739,50/(1+0,14)[3]+
+44513154,19/(1+0,14)[4]=8789031,09+18768739,04+24224551,89+26355360,68=
=78137682,68 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot \quad , \quad (5.23)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2,2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 480120,10 грн.

$ВВ = k_{інв} \cdot$ = 2,2 · 480120,10 = 1056264,22 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.24}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 78137682,68 грн;

PV – теперішня вартість початкових інвестицій, 1056264,22 грн.

$E_{абс} = ПП - PV$ = 78137682,68 - 1056264,22 = 77081418,47 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.25}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 77081418,47 грн;

PV – теперішня вартість початкових інвестицій, 1056264,22 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+77081418,47/1056264,22)$^{1/4}$= 1,93.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{5.26}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,4.

$\tau_{мін}$ = 0,12+0,4 = 0,52 < 1,93 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Система самостійного касового розрахунку з визначенням характеристик покупця» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.27}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,93 = 0,52 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновок

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Система самостійного касового розрахунку з визначенням характеристик покупця» становить 34,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,39 рази.

Також термін окупності становить 0,52 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Система самостійного касового розрахунку з визначенням характеристик покупця».

ВИСНОВКИ

У магістерській кваліфікаційній роботі успішно створено систему самостійного касового розрахунку, обладнану розпізнаванням характеристик покупця, що стала ще одним кроком у напрямку оптимізації та покращення процесу роздрібної торгівлі. Програмний додаток, створений в середовищах Visual Studio 2022 та Microsoft Visual Studio Code, сприяє прискоренню та спрощенню процесу покупки товарів у різноманітних закладах роздрібної торгівлі за допомогою кас, що містять концепцію самообслуговування.

Глибокий аналіз проблеми показав, що чинні рішення мають свої обмеження. Вирішено використовувати мови програмування C# та Angular для графічного інтерфейсу, а також бібліотеку Entity Framework для бази даних. Розроблено архітектуру системи, удосконалено алгоритми та оптимізовано графічний інтерфейс.

Однією з ключових вдосконалень є інтеграція розпізнавання характеристик покупця, зокрема віку та статі. Використовуючи нейронні мережі та моделі глибокого навчання, реалізовано точне визначення особливостей обличчя, що сприяє покращенню персоналізації обслуговування в роздрібних магазинах.

Процес розпізнавання віку та статі базується на аналізі зображень через моделі штучного інтелекту. Розглянуто варіанти використання систем розпізнавання обличчя, таких як FaceID від Apple та Azure Face API від Microsoft, які демонструють високу точність визначення характеристик особи.

Успішно впроваджені інструменти та технології роботи з відомими аналогами підтверджують ефективність використаних рішень. Виведення розпізнавання покупця на новий рівень виявляє великий потенціал для полегшення та удосконалення процесу роздрібної торгівлі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розуміння впровадження технологій роздрібної каси самообслуговування з використанням аналізу необхідних умов [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/363579388_Understanding_the_implementation_of_retail_self-service_checkout_technologies.

2. Впровадження онлайн-касових апаратів: переваги, міркування та вказівки [Електронний ресурс] – Режим доступу до ресурсу: https://www.oecd.org/tax/forum-on-tax-administration/publications-and-products/implementing-online-cash-registers-benefits-considerations-and-guidance.pdf

3. Каса самообслуговування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/self-service.

4. Переваги каси самообслуговування: [Електронний ресурс] – Режим доступу до ресурсу: //nashkraj.ua/uk/blog/perevagy-kasy-samoobslugovuvannya-kso/.

5. What are self-checkout systems? [Електронний ресурс] – Режим доступу до ресурсу: https://research.aimultiple.com/self-checkout/#:~:text=Self%2Dcheckouts%20(SCOs)%20are,they%20scan%20them%20through%20barcodes.

6. Програмне забезпечення кас самообслуговування [Електронний ресурс] – Режим доступу до ресурсу: https://sprintingretail.com/blog/retail-self-checkout-systems/.

7. Каси самообслуговування Self-Checkout [Електронний ресурс] – Режим доступу до ресурсу: https://selfservice4u.com/ua/ kassa-samoosbluzhivaniya-self-checkout.

8. How to Implement Self Checkout In Your Retail Business [Електронний ресурс] – Режим доступу до ресурсу: https://staxpayments.com/blog/how-to-implement-self-checkout-in-retail-business/.

9. What Are The Ways to Implement Self Checkout System? [Електронний ресурс] – Режим доступу до ресурсу: https://www.solumesl.com/en/insights/implement-self-checkout-retail.

10. SELINE – SELF-CHECKOUT SYSTEM [Електронний ресурс] – Режим доступу до ресурсу: https://mint-innovations.com/en/product/seline.

11. Self-checkout systems [Електронний ресурс] – Режим доступу до ресурсу: https://bs2.lt/product/self-checkout-systems/.

12. Термінали самообслуговування у McDonald's: світові технології з українською специфікою [Електронний ресурс] – Режим доступу до ресурсу: https://systemgroup.com.ua/ua/project/terminaly-samoobsluzhivaniya-v-mcdonalds-mirovye-tehnologii-s-ukrainskoy-specifikoy.

13. 6 Ways to Implement Self-Checkout Right [Електронний ресурс] – Режим доступу до ресурсу: https://www.netguru.com/blog/self-checkouts-tips.

14. Offer customers the convenience they're looking for with self-checkout [Електронний ресурс] – Режим доступу до ресурсу: https://www.ncr.com/retail/convenience-fuel-retail/self-checkout.

15. Facial Age Estimation Using Machine Learning Techniques: An Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/364739009_Facial_Age_Estimation_Using_Machine_Learning_Techniques_An_Overview.

16. Age Detection Using Artificial Intelligence [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@evertongomede/age-detection-using-artificial-intelligence-719b4368075b.

17. Facial Age Estimation Using Machine Learning Techniques: An Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.mdpi.com/2504-2289/6/4/128.

18. Age & Gender Detection: Top Use Cases [Електронний ресурс] – Режим доступу до ресурсу: https://www.plugger.ai/blog/age-gender-detection-top-use-cases.

19. AGE ESTIMATION FROM FACIAL IMAGES USING MACHINE LEARNING [Електронний ресурс] – Режим доступу до ресурсу: chrome-

extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.theseus.fi/bitstream/handle/10024/804002/Nguyen_Dang.pdf?sequence=2&isAllowed=y.

20. Age estimation from faces using deep learning: A comparative analysis [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S1077314220300424.

21. Age and Gender Detection Using Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/07/age-and-gender-detection-using-deep-learning/.

22. How-to-Use Machine Learning for Buying Behavior Prediction: A Case Study on Sales Prospecting [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/omdena/how-to-use-machine-learning-forbuyingbehaviorprediction-a-case-study-on-sales-prospecting-c496edd894cd.

23. PredictingConsumerPurchase behaviorusingAutomatic MachineLearning [Електронний ресурс] – Режим доступу до ресурсу: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.divaportal.org/smash/get/diva2:1707587/FULLTEXT01.pdf.

24. Multiple Face Detection and Recognition in Real Time [Електронний ресурс] – Режим доступу до ресурсу: https://www.codeproject.com/Articles/239849/Multiple-Face-Detection-and-Recognition-in-Real-2.

25. Face Detection and Recognition with C# EmguCV [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/mesutpiskin/face-detection-and-recognition.

26. AGE, GENDER AND ETHNICITY (FACE DATA) CSV [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/nipunarora8/age-gender-and-ethnicity-face-data-csv.

27. Роберт Мартін: Чиста архітектура: мистецтво розробки програмного забезпечення: Фабула, 2019, 416с.

28. Paul Clements, Rick Kazman: Software Architecture in Practice: Addison-Wesley Professional, 2012, 322p.

29. Mark Richards, Neal Ford: Fundamentals of Software Architecture: O'Reilly Media, Inc., 2020, 345p.

30. Архітектура та проектування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://learn.ztu.edu.ua/mod/book/view.php?id=278.

31.What is Onion Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://codewithmukesh.com/blog/onion-architecture-in-aspnet-core.

32. Craig Mullins: Database Administration: Addison-Wesley, 2002, 703p

33. Що таке база даних? [Електронний ресурс] – Режим доступу до ресурсу: https://apeps.kpi.ua/shco-take-basa-danykh.

34. Типи баз даних: особливості, відмінності та приклади [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/types-of-databases/

35. Що таке дані? [Електронний ресурс] – Режим доступу до ресурсу: https://futurenow.com.ua/shho-take-bazy-danyh-yih-pryznachennya-ta-vydy/

36. SQL база даних Для чого призначена база даних? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukraine.com.ua/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html.

37. Що таке база даних. [Електронний ресурс] – Режим доступу до ресурсу: https://hostiq.ua/wiki/ukr/database/.

38. 11 типів сучасних баз даних: короткий опис, схеми і приклади БД [Електронний ресурс] – Режим доступу до ресурсу: https://senior.ua/articles/11-tipv-suchasnih-baz-danih-korotkiy-opis-shemi--prikladi-bd.

39. БАЗИ ДАНИХ ТА ІНФОРМАЦІЙНІ СИСТЕМИ [Електронний ресурс] – Режим доступу до ресурсу: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://ep3.nuwm.edu.ua/9129/3/%D0%A5%D0%B0%D1%80%D1%96%D0%B2%20%D0%9D.%D0%9E.pdf.

40. Basic Database Operations Using C# [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/basic-database-operations-using-c-sharp/.

41. Python Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/python_ intro.asp.

42. Java Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/java/java_intro.asp.

43. Mark J. Price: C# 9 and .NET 5 - Modern Cross-Platform Development: Packt Publishing, 2020, 432p.

44. What is Entity Framework? [Електронний ресурс] – Режим доступу до ресурсу: https://www.entityframeworktutorial.net /what-is-entityframework.aspx.

45. Jon Skeet: C# in Depth: Eric Lippert, 2019, 237p.

46. C# [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/C Sharp.

47. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/visualstudio/ get-started/visual-studio-ide?view=vs-2022.

48. Project Rider [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/rider/.

49. Eclipse [Електронний ресурс] – Режим доступу до ресурсу: https://www.eclipse.org/community/eclipse_newsletter/2017/ august/article3.php.

50. Основні поняття Сутностей [Електронний ресурс] – Режим доступу до ресурсу: https://uadoc.zavantag.com/text/33068/ index-1.html.

51. Репозиторії та інформаційні системи [Електронний ресурс] – Режим доступу до ресурсу: http://ep3.nuwm.edu.ua/9129/.

52. What is Data Mapping? [Електронний ресурс] – Режим доступу до ресурсу: https://ua.talend.com/resources/data-mapping/.

53. Paul C. Jorgensen: Software Testing: A Craftsman’s Approach, Fourth Edition: Hardcover – Import, 2012, 305p.

54. Тестування програмного забезбечення [Електронний ресурс] – Режим доступу до ресурсу: https://www.quality-assurance-group.com/shho-take-testuvannyaprogramnogo-zabezpechennya-ta-yake-jogoznachen- nya.

55. В. О. Козловський О. Й. Лесько, В. В. Кавецький Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. – Вінниця: ВНТУ. – 2021. – 42 с.

56. В. В. Кавецький, В. О. Козловський, І. В. Причепа Економічне обґрунтування інноваційних рішень: [практикум] – Вінниця : ВНТУ. – 2016. – 113 с.

ДОДАТКИ

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: « Система самостійного касового розрахунку з визначенням характеристик покупця»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,2% Схожість 3,8%

Аналіз звіту подібності (відмітити потрібне)

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Анатолій РИБАК
(підпис) (прізвище, ініціали)

Керівник роботи _____ Тетяна ГРИЩУК
(підпис) (прізвище, ініціали)

Додаток Б. Технічне завдання

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о. Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
"6" жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Система самостійного касового розрахунку з визначенням характеристик покупця

08-33.МКР.005.00.000 ТЗ

Студент групи 2АКІТ-22м

_____Анатолій РИБАК

Керівник к.т.н., доцент кафедри КСУ

_____Тетяна ГРИЩУК

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Система самостійного касового розрахунку з визначенням характеристик покупця.

1.2. Галузь застосування – магазини, супермаркети, торгові центри, заклади харчування.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від №247 від 18-09-2023р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення швидкості та зручності процесу купівлі товарів у різних точках роздрібної торгівлі шляхом розробки програмного забезпечення для систем самостійного касового розрахунку, а також підвищення попиту на більш широкий асортимент товарів роздрібної торгівлі за рахунок удосконалення алгоритмів нарахування знижок та розпізнання характеристик користувача, не турбуючись за безпеку продажу обмежених за віком товарів.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Каси самообслуговування Self-Checkout [Електронний ресурс] – Режим доступу до ресурсу: https://selfservice4u.com/ua/ kassa-samoosbluzhivaniya-self-checkout.

2. Facial Age Estimation Using Machine Learning Techniques: An Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.mdpi.com/2504-2289/6/4/128.

3. OpenCV C#: What is it, How to Use and its Applications [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/opencv-csharp.

4. C# [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/C Sharp.

5. What is Angular?: Architecture, Features, and Advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular.

6. Тестування програмного забезбечення [Електронний ресурс] – Режим доступу до ресурсу: https://www.quality-assurance-group.com/shho-take-testuvannyaprogramnogo-zabezpechennya-ta-yake-jogoznachen-nya.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Вивчення характеристик розробки;
- Оптимізація процесів постачання купівлі товару;
- Розпізнвання характеристик покупця;
- Керування базою даних;
- Вивчення можливостей системи.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- База даних MS SQL;
- Visual Studio 2022;
- WebStorm 2022.

5.2.2. Умови експлуатації системи:

- робота на системах самостійного касового розрахунку;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «09»_09__ 2023 р.

2. Аналіз технічного завдання «15»__10___ 2023 р.
3. Аналіз методів розв’язання поставленої задачі «29»__10___ 2023 р.
4. Розробка прототипу підсистеми визнання віку та статі людини «10»___11__ 2023 р.
5. Розробка програмного забезпечення системи «20»___11__ 2023 р.
6. Тестування програмного забезпечення «30»___11__ 2023 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «26»__11__ 2023 р.
2. Розробка моделі бази даних системи «27»__11___ 2023 р.
3. Розробка демонстраційних плакатів «28»__11___ 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28»__11_ 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «11»__12_ 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «20»__12_ 2023 р.

Додаток В. Лістинг програми

(довідковий)

Фрагмент лістингу програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Data.Entities
{
   public class Product : BaseEntity
   {
      public string ProductName { get; set; }
      public decimal Price { get; set; }

      public int ProductCategoryId { get; set; }
      public ProductCategory Category { get; set; }
      public ICollection<ReceiptDetail> ReceiptDetails { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Data.Entities
{
   public class Person : BaseEntity
   {
      public string Name { get; set; }
      public string Surname { get; set; }
      public DateTime BirthDate { get; set; }
      public Customer Customer { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Data.Entities
{
   public class Customer : BaseEntity
   {
      public int DiscountValue { get; set; }
      public int PersonId { get; set; }

      public Person Person { get; set; }
      public ICollection<Receipt> Receipts { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Data.Entities
{
   public class ProductCategory : BaseEntity
   {
      public string CategoryName { get; set; }

      public ICollection<Product> Products { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Data.Entities
{
   public class Receipt : BaseEntity
   {
      public DateTime OperationDate { get; set; }
      public bool IsCheckedOut { get; set; }

      public int CustomerId { get; set; }

      public Customer Customer { get; set; }
      public ICollection<ReceiptDetail> ReceiptDetails { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace Data.Entities
{
   public class ReceiptDetail : BaseEntity
   {
      public decimal DiscountUnitPrice { get; set; }
      public decimal UnitPrice { get; set; }
      public int Quantity { get; set; }

      public int ReceiptId { get; set; }
      public int ProductId { get; set; }

      public Receipt Receipt { get; set; }
      public Product Product { get; set; }
   }
}

using Data.Interfaces;
using Data.Entities;
using Data.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;

namespace Data.Repositories
{
   public class Repository<TEntity> : IRepository<TEntity> where TEntity : BaseEntity
   {
      protected readonly TradeMarketDbContext _context;
      protected readonly DbSet<TEntity> _dbSet;
      public Repository(TradeMarketDbContext context)
      {
         _context = context;
         _dbSet = context.Set<TEntity>();
      }
      public async Task AddAsync(TEntity entity)
      {
         await _dbSet.AddAsync(entity);
      }
      public void Delete(TEntity entity)
      {
         _dbSet.Remove(entity);
      }
      public async Task DeleteByIdAsync(int id)
      {
         var entity = await _dbSet.FindAsync(id);
         _dbSet.Remove(entity);
      }
```

```
        public async Task<IEnumerable<TEntity>> GetAllAsync()
        {
            return await _dbSet.AsNoTracking().ToListAsync();
        }
        public async Task<TEntity> GetByIdAsync(int id)
        {
            return await _dbSet.FindAsync(id);
        }
        public void Update(TEntity entity)
        {
            _context.Entry(entity).State = EntityState.Modified;
        }
    }
}

using Data.Data;
using Data.Entities;
using Data.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Data.Repositories
{
    public class CustomerRepository : Repository<Customer>, ICustomerRepository
    {
        public CustomerRepository(TradeMarketDbContext context) : base(context)
        {
        }

        public async Task<IEnumerable<Customer>> GetAllWithDetailsAsync()
        {
            return await _context.Customers.Include(x => x.Receipts).ThenInclude(i => i.ReceiptDetails)
                                    .Include(x => x.Person)
                                    .ToListAsync();
        }

        public async Task<Customer> GetByIdWithDetailsAsync(int id)
        {
            return await _context.Customers.Include(x => x.Receipts).ThenInclude(x => x.ReceiptDetails)
                                    .Include(x => x.Person)
                                    .FirstOrDefaultAsync(x => x.Id == id);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using Data.Data;
using Data.Entities;
using Data.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace Data.Repositories
{
   public class ProductRepository : Repository<Product>, IProductRepository
   {
      public ProductRepository(TradeMarketDbContext context) : base(context)
      {
      }

      public async Task<IEnumerable<Product>> GetAllWithDetailsAsync()
      {
         return await _context.Products.Include(x => x.Category)
                              .Include(x => x.ReceiptDetails)
                              .ToListAsync();
      }
      public async Task<Product> GetByIdWithDetailsAsync(int id)
      {
         return await _context.Products.Include(x => x.Category)
                              .Include(x => x.ReceiptDetails)
                              .FirstOrDefaultAsync(x => x.Id == id);
      }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Data.Data;
using Data.Entities;
using Data.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace Data.Repositories
{
   public class ReceiptRepository : Repository<Receipt>, IReceiptRepository
   {
      public ReceiptRepository(TradeMarketDbContext context) : base(context)
      {
      }

      public async Task<IEnumerable<Receipt>> GetAllWithDetailsAsync()
      {
         return await _context.Receipts.Include(x => x.ReceiptDetails).ThenInclude(i => i.Product)
                              .Include(x => x.ReceiptDetails).ThenInclude(i => i.Product.Category)
```

```
                .Include(x => x.Customer)
                .ToListAsync();
        }
        public async Task<Receipt> GetByIdWithDetailsAsync(int id)
        {
            return await _context.Receipts.Include(x => x.ReceiptDetails).ThenInclude(i => i.Product)
                .Include(x => x.ReceiptDetails).ThenInclude(i => i.Product.Category)
                .Include(x => x.Customer)
                .FirstOrDefaultAsync(x => x.Id == id);
        }
    }
}

// <auto-generated />
using System;
using Data.Data;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace Data.Migrations
{
    [DbContext(typeof(TradeMarketDbContext))]
    partial class TradeMarketDbContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "3.1.6")
                .HasAnnotation("Relational:MaxIdentifierLength", 128)
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            modelBuilder.Entity("Data.Entities.Customer", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<int>("DiscountValue")
                        .HasColumnType("int");

                    b.Property<int>("PersonId")
                        .HasColumnType("int");

                    b.HasKey("Id");

                    b.HasIndex("PersonId")
```

```
                .IsUnique();

            b.ToTable("Customers");
        });

        modelBuilder.Entity("Data.Entities.Person", b =>
        {
            b.Property<int>("Id")
                .ValueGeneratedOnAdd()
                .HasColumnType("int")
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            b.Property<DateTime>("BirthDate")
                .HasColumnType("datetime2");

            b.Property<string>("Name")
                .HasColumnType("nvarchar(max)");

            b.Property<string>("Surname")
                .HasColumnType("nvarchar(max)");

            b.HasKey("Id");

            b.ToTable("Persons");
        });

        modelBuilder.Entity("Data.Entities.Product", b =>
        {
            b.Property<int>("Id")
                .ValueGeneratedOnAdd()
                .HasColumnType("int")
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            b.Property<decimal>("Price")
                .HasColumnType("decimal(18,2)");

            b.Property<int>("ProductCategoryId")
                .HasColumnType("int");

            b.Property<string>("ProductName")
                .HasColumnType("nvarchar(max)");

            b.HasKey("Id");

            b.HasIndex("ProductCategoryId");

            b.ToTable("Products");
        });

        modelBuilder.Entity("Data.Entities.ProductCategory", b =>
```

```
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("int")
                    .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                b.Property<string>("CategoryName")
                    .HasColumnType("nvarchar(max)");

                b.HasKey("Id");

                b.ToTable("ProductCategories");
            });

        modelBuilder.Entity("Data.Entities.Receipt", b =>
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("int")
                    .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                b.Property<int>("CustomerId")
                    .HasColumnType("int");

                b.Property<bool>("IsCheckedOut")
                    .HasColumnType("bit");

                b.Property<DateTime>("OperationDate")
                    .HasColumnType("datetime2");

                b.HasKey("Id");

                b.HasIndex("CustomerId");

                b.ToTable("Receipts");
            });

        modelBuilder.Entity("Data.Entities.ReceiptDetail", b =>
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("int")
                    .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                b.Property<double>("DiscountUnitPrice")
                    .HasColumnType("float");

                b.Property<int>("ProductId")
                    .HasColumnType("int");
```

```
            b.Property<int>("Quantity")
                .HasColumnType("int");

            b.Property<int>("ReceiptId")
                .HasColumnType("int");

            b.Property<double>("UnitPrice")
                .HasColumnType("float");

            b.HasKey("Id");

            b.HasIndex("ProductId");

            b.HasIndex("ReceiptId");

            b.ToTable("ReceiptsDetails");
        });

    modelBuilder.Entity("Data.Entities.Customer", b =>
        {
            b.HasOne("Data.Entities.Person", "Person")
                .WithOne("Customer")
                .HasForeignKey("Data.Entities.Customer", "PersonId")
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();
        });

    modelBuilder.Entity("Data.Entities.Product", b =>
        {
            b.HasOne("Data.Entities.ProductCategory", "Category")
                .WithMany("Products")
                .HasForeignKey("ProductCategoryId")
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();
        });

    modelBuilder.Entity("Data.Entities.Receipt", b =>
        {
            b.HasOne("Data.Entities.Customer", "Customer")
                .WithMany("Receipts")
                .HasForeignKey("CustomerId")
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();
        });

    modelBuilder.Entity("Data.Entities.ReceiptDetail", b =>
        {
            b.HasOne("Data.Entities.Product", "Product")
                .WithMany("ReceiptDetails")
                .HasForeignKey("ProductId")
                .OnDelete(DeleteBehavior.Cascade)
```

```
                .IsRequired();

            b.HasOne("Data.Entities.Receipt", "Receipt")
                .WithMany("ReceiptDetails")
                .HasForeignKey("ReceiptId")
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();
        });
#pragma warning restore 612, 618
    }
  }
}
using System;
using Microsoft.EntityFrameworkCore.Migrations;

namespace Data.Migrations
{
  public partial class InitialCreate : Migration
  {
    protected override void Up(MigrationBuilder migrationBuilder)
    {
      migrationBuilder.CreateTable(
        name: "Persons",
        columns: table => new
        {
          Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
          Name = table.Column<string>(nullable: true),
          Surname = table.Column<string>(nullable: true),
          BirthDate = table.Column<DateTime>(nullable: false)
        },
        constraints: table =>
        {
          table.PrimaryKey("PK_Persons", x => x.Id);
        });

      migrationBuilder.CreateTable(
        name: "ProductCategories",
        columns: table => new
        {
          Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
          CategoryName = table.Column<string>(nullable: true)
        },
        constraints: table =>
        {
          table.PrimaryKey("PK_ProductCategories", x => x.Id);
        });

      migrationBuilder.CreateTable(
        name: "Customers",
        columns: table => new
```

```
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            DiscountValue = table.Column<int>(nullable: false),
            PersonId = table.Column<int>(nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Customers", x => x.Id);
            table.ForeignKey(
                name: "FK_Customers_Persons_PersonId",
                column: x => x.PersonId,
                principalTable: "Persons",
                principalColumn: "Id",
                onDelete: ReferentialAction.Cascade);
        });

    migrationBuilder.CreateTable(
        name: "Products",
        columns: table => new
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            ProductName = table.Column<string>(nullable: true),
            Price = table.Column<decimal>(nullable: false),
            ProductCategoryId = table.Column<int>(nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Products", x => x.Id);
            table.ForeignKey(
                name: "FK_Products_ProductCategories_ProductCategoryId",
                column: x => x.ProductCategoryId,
                principalTable: "ProductCategories",
                principalColumn: "Id",
                onDelete: ReferentialAction.Cascade);
        });

    migrationBuilder.CreateTable(
        name: "Receipts",
        columns: table => new
        {
            Id = table.Column<int>(nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            OperationDate = table.Column<DateTime>(nullable: false),
            IsCheckedOut = table.Column<bool>(nullable: false),
            CustomerId = table.Column<int>(nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Receipts", x => x.Id);
            table.ForeignKey(
```

```
            name: "FK_Receipts_Customers_CustomerId",
            column: x => x.CustomerId,
            principalTable: "Customers",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateTable(
    name: "ReceiptsDetails",
    columns: table => new
    {
        Id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        DiscountUnitPrice = table.Column<double>(nullable: false),
        UnitPrice = table.Column<double>(nullable: false),
        Quantity = table.Column<int>(nullable: false),
        ReceiptId = table.Column<int>(nullable: false),
        ProductId = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_ReceiptsDetails", x => x.Id);
        table.ForeignKey(
            name: "FK_ReceiptsDetails_Products_ProductId",
            column: x => x.ProductId,
            principalTable: "Products",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_ReceiptsDetails_Receipts_ReceiptId",
            column: x => x.ReceiptId,
            principalTable: "Receipts",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });

migrationBuilder.CreateIndex(
    name: "IX_Customers_PersonId",
    table: "Customers",
    column: "PersonId",
    unique: true);

migrationBuilder.CreateIndex(
    name: "IX_Products_ProductCategoryId",
    table: "Products",
    column: "ProductCategoryId");

migrationBuilder.CreateIndex(
    name: "IX_Receipts_CustomerId",
    table: "Receipts",
    column: "CustomerId");
```

```
            migrationBuilder.CreateIndex(
                name: "IX_ReceiptsDetails_ProductId",
                table: "ReceiptsDetails",
                column: "ProductId");

            migrationBuilder.CreateIndex(
                name: "IX_ReceiptsDetails_ReceiptId",
                table: "ReceiptsDetails",
                column: "ReceiptId");
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "ReceiptsDetails");

            migrationBuilder.DropTable(
                name: "Products");

            migrationBuilder.DropTable(
                name: "Receipts");

            migrationBuilder.DropTable(
                name: "ProductCategories");

            migrationBuilder.DropTable(
                name: "Customers");

            migrationBuilder.DropTable(
                name: "Persons");
        }
    }
}

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Threading;
using System.Windows.Forms;
using OpenCvSharp;
using OpenCvSharp.Dnn;
using OpenCvSharp.Extensions;
using Point = OpenCvSharp.Point;
using Size = OpenCvSharp.Size;

namespace Demo10_WinFormAgeAndGender
{
    public partial class Form1 : Form
    {
        private bool _run = true;
        private bool _doFaceDetection = true;
        private bool _doAgeGender = false;
```

```
        private VideoCapture _capture;
        private Mat _image;
        private Thread _cameraThread;
        private bool _fps = false;
        private Net _faceNet;
        private Net _ageNet;
        private Net _genderNet;
        private const int LineThickness = 2;
        private const int Padding = 10;
        private readonly List<string> _genderList = new List<string> { "Male", "Female" };
        private readonly List<string> _ageList = new List<string> { "(0-2)", "(4-6)", "(8-12)", "(14-17)",
"(18-23)", "(25-32)", "(38-43)", "(50+)" };


        public Form1()
        {
            InitializeComponent();
            Load += Form1_Load;
            Closed += Form1_Closed;
            fps.Click += buttonFPS_Click;
            AgeAndGender.Click += btnAgeGender_Click;
        }

        private void Form1_Closed(object sender, EventArgs e)
        {
            _cameraThread.Interrupt();
            _capture.Release();
        }

        private void buttonFPS_Click(object sender, EventArgs e)
        {
            _fps = !_fps;
        }

        private void btnAgeGender_Click(object sender, EventArgs e)
        {
            _doAgeGender = !_doAgeGender;
        }


        private void Form1_Load(object sender, EventArgs e)
        {
            // # detect faces, age and gender using models from
https://github.com/spmallick/learnopencv/tree/08e61fe80b8c0244cc4029ac11e44cd0fbb008c3/AgeGe
nder
            const string faceProto =
"C:\\Users\\Ace\\Desktop\\Demo10_WinFormAgeAndGender\\Demo10_WinFormAgeAndGender\\D
emo10_WinFormAgeAndGender\\models/deploy.prototxt";
            const string faceModel =
"C:\\Users\\Ace\\Desktop\\Demo10_WinFormAgeAndGender\\Demo10_WinFormAgeAndGender\\D
emo10_WinFormAgeAndGender\\models/res10_300x300_ssd_iter_140000_fp16.caffemodel";
```

```
        const string ageProto = @"C:\Users\Ace\Desktop\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\models/age_deploy.prototxt";
        const string ageModel = @"C:\Users\Ace\Desktop\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\models/age_net.caffemodel";
        const string genderProto = @"C:\Users\Ace\Desktop\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\models/gender_deploy.prototxt";
        const string genderModel = @"C:\Users\Ace\Desktop\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\Demo10_WinFormAgeAndGender\models/gender_net.caffemodel";
        _ageNet = CvDnn.ReadNetFromCaffe(ageProto, ageModel);
        _genderNet = CvDnn.ReadNetFromCaffe(genderProto, genderModel);
        _faceNet = CvDnn.ReadNetFromCaffe(faceProto, faceModel);

        _capture = new VideoCapture(0);
        _image = new Mat();
        _cameraThread = new Thread(new ThreadStart(CaptureCameraCallback));
        _cameraThread.Start();
    }

    private void CaptureCameraCallback()
    {
        while (true)
        {
            if (!_run) continue;
            var startTime = DateTime.Now;

            _capture.Read(_image);
            if (_image.Empty()) return;
            var imageRes = new Mat();
            Cv2.Resize(_image, imageRes, new Size(640, 480));
            var newImage = imageRes.Clone();

            if (_doFaceDetection) DetectFaces(newImage, imageRes);

            if (_fps) CalculateFps(startTime, newImage);

            var bmpWebCam = BitmapConverter.ToBitmap(imageRes);
            var bmpEffect = BitmapConverter.ToBitmap(newImage);


            pictureBox1.Image = bmpEffect;
        }
    }

    private static void CalculateFps(DateTime startTime, Mat imageRes)
    {
        var diff = DateTime.Now - startTime;
        var fpsInfo = $"FPS: Nan";
        if (diff.Milliseconds > 0)
```

```
        {
            var fpsVal = 1.0 / diff.Milliseconds * 1000;
            fpsInfo = $"FPS: {fpsVal:00}";
        }

        Cv2.PutText(imageRes, fpsInfo, new Point(10, 20), HersheyFonts.HersheyComplexSmall, 1, Scalar.White);
    }

    private void DetectFaces(Mat newImage, Mat imageRes)
    {
        int frameHeight = newImage.Rows;
        int frameWidth = newImage.Cols;

        using var blob = CvDnn.BlobFromImage(newImage, 1.0, new Size(300, 300), new Scalar(104, 117, 123), false, false);
        _faceNet.SetInput(blob, "data");

        using var detection = _faceNet.Forward("detection_out");
        using var detectionMat = new Mat(detection.Size(2), detection.Size(3), MatType.CV_32F, detection.Ptr(0));

        for (int i = 0; i < detectionMat.Rows; i++)
        {
            float confidence = detectionMat.At<float>(i, 2);

            if (confidence > 0.7)
            {
                int x1 = (int)(detectionMat.At<float>(i, 3) * frameWidth);
                int y1 = (int)(detectionMat.At<float>(i, 4) * frameHeight);
                int x2 = (int)(detectionMat.At<float>(i, 5) * frameWidth);
                int y2 = (int)(detectionMat.At<float>(i, 6) * frameHeight);

                Cv2.Rectangle(newImage, new Point(x1, y1), new Point(x2, y2), Scalar.Green, LineThickness);

                if (_doAgeGender)
                    AnalyzeAgeAndGender(x1, y1, x2, y2, imageRes, newImage);
            }
        }
    }

    private void AnalyzeAgeAndGender(int x1, int y1, int x2, int y2, Mat imageRes, Mat newImage)
    {
        var x = x1 - Padding;
        var y = y1 - Padding;
        var w = (x2 - x1) + Padding * 3;
        var h = (y2 - y1) + Padding * 3;
        Rect roiNew = new Rect(x, y, w, h);
        var face = imageRes[roi: roiNew];

        var meanValues = new Scalar(78.4263377603, 87.7689143744, 114.895847746);
```

```
        var blobGender = CvDnn.BlobFromImage(face, 1.0, new Size(227, 227), mean: meanValues, swapRB: false);
        _genderNet.SetInput(blobGender);
        var genderPreds = _genderNet.Forward();

        GetMaxClass(genderPreds, out int classId, out double classProbGender);
        var gender = _genderList[classId];

        _ageNet.SetInput(blobGender);
        var agePreds = _ageNet.Forward();
        GetMaxClass(agePreds, out int classIdAge, out double classProbAge);
        var ageRange = _ageList[classIdAge];

        if (IsAgeUnderLimit(ageRange, 18))
        {
            var label = $"{gender} {GetAgeValue(ageRange)}, {ageRange}";
            Cv2.PutText(newImage, label, new Point(x1 - 10, y2 + 20), HersheyFonts.HersheyComplexSmall, 1, Scalar.Red, 1);
            Cv2.Rectangle(newImage, new Point(x1, y1), new Point(x2, y2), Scalar.Blue, LineThickness);
            return;
        }

        var labelGreen = $"{gender} {GetAgeValue(ageRange)}, {ageRange}";
        Cv2.PutText(newImage, labelGreen, new Point(x1 - 10, y2 + 20), HersheyFonts.HersheyComplexSmall, 1, Scalar.Green, 1);
        Cv2.Rectangle(newImage, new Point(x1, y1), new Point(x2, y2), Scalar.Blue, LineThickness);
    }

    private string GetAgeValue(string ageRange)
    {
        var ageValues = ageRange.Trim('(', ')').Split('-');
        if (ageValues.Length == 2 && int.TryParse(ageValues[0], out int minAge) && int.TryParse(ageValues[1], out int maxAge))
        {
            var averageAge = (minAge + maxAge) / 2;
            return $"{(int)averageAge}";
        }
        return "";
    }


    private bool IsAgeUnderLimit(string ageRange, int limit)
    {
        var ageValues = ageRange.Trim('(', ')').Split('-');
        if (ageValues.Length == 2 && int.TryParse(ageValues[0], out int minAge) && int.TryParse(ageValues[1], out int maxAge))
        {
            var averageAge = (minAge + maxAge) / 2;
            return averageAge < limit;
        }
        return true;
```

```
        }



        private void GetMaxClass(Mat probBlob, out int classId, out double classProb)
        {
            using var probMat = probBlob.Reshape(1, 1);
            Cv2.MinMaxLoc(probMat, out _, out classProb, out _, out var classNumber);
            classId = classNumber.X;
            Debug.WriteLine($"X: {classNumber.X} – Y: {classNumber.Y} ");
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
        }
    }
}
```

Додаток Г. Ілюстративна частина

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА САМОСТІЙНОГО КАСОВОГО РОЗРАЗУНКУ З ВИЗНАЧЕННЯМ ХАРАКТЕРИСТИК ПОКУПЦЯ

1. Титульний слайд
2. Актуальність теми
3. Мета, об'єкт та предмет дослідження
4. Задачі дослідження
5. Новизна одержаних результатів
6. Практична цінність одержаних результатів
7. Використані фреймворки
8. Використання архітектури Onion
9. Використання Microsoft SQL Server
10. Використання бібліотеки OpenCV
11. Схема бази даних
12. Блок-схема загального алгоритму
13. Блок-схема алгоритму генерування знижок
14. Тестування системи
15. Апробація та публікація результатів роботи
16. Фінальний слайд

Студент групи 2АКІТ-22м

Підпис Анатолій РИБАК
Ім'я ПРІЗВИЩЕ

Керівник к.т.н., доцент кафедри КСУ

Підпис Тетяна ГРИЩУК
Ім'я ПРІЗВИЩЕ

Вінницький національний технічний університет

Магістерська кваліфікаційна робота

На тему:Система самостійного касового розрахунку з визначенням характеристик покупця

Виконав:
Студент групи 2АКІТ-22м
Рибак А.Ю.

Науковий керівник:
к.т.н., доцент кафедри КСУ
Грищук Т.В.

Рисунок Г.1 – Титульний слайд

Актуальність теми

- Україна переживає збільшений попит на системи самообслуговування, що вказує на рост інтересу до новітніх технологій у роздрібній торгівлі. Впровадження систем самостійного касового розрахунку є стратегічним кроком для підтримки цього технологічного розвитку, сприяючи підвищенню конкурентоспроможності та ефективності роздрібних підприємств в Україні.
- Системи самостійного касового розрахунку стають ключовим елементом самообслуговування, особливо в умовах великого потоку клієнтів, сприяючи уникненню черг на традиційних касах.
- Розпізнавання віку покупців у касах самообслуговування є критично важливою функцією для управління продажами обмежених товарів. Це не лише забезпечує дотримання законодавства, а й підвищує рівень безпеки та контролю над обігом товарів.

Рисунок Г.2 – Актуальність теми

Мета, об'єкт та предмет дослідження

- Метою магістерської кваліфікаційної роботи є підвищення швидкості та зручності процесу купівлі товарів у різних точках роздрібної торгівлі шляхом розробки програмного забезпечення для систем самостійного касового розрахунку, а також підвищення попиту на більш широкий асортимент товарів роздрібної торгівлі за рахунок удосконалення алгоритмів нарахування знижок та розпізнання характеристик користувача, не турбуючись за безпеку продажу обмежених за віком товарів.
- Об'єктом дослідження є процес самостійного касового розрахунку.
- Предметом дослідження є методи та алгоритми додаткової верифікації покупців та рекомендації товарів.

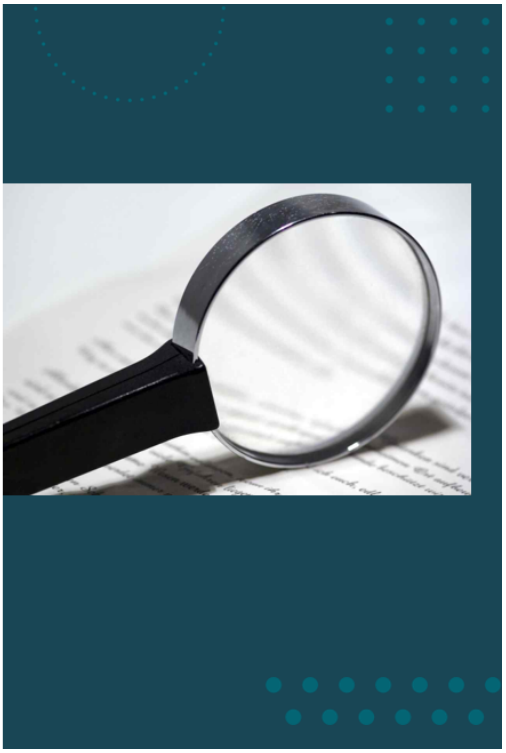

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

Задачі

- Виконати аналіз сучасних систем самостійного касового розрахунку;
- Розробити архітектуру системи;
- Провести варіантний аналіз та розробити базу даних системи;
- Розробити алгоритм додаткової верифікації користувача;
- Розробити метод та алгоритм роботи рекомендаційної системи на основі характеристик покупця;
- Розробити програмне забезпечення для системи самостійного касового розрахунку базуючись на створеній архітектурі та алгоритмах.

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Новизна одержаних результатів

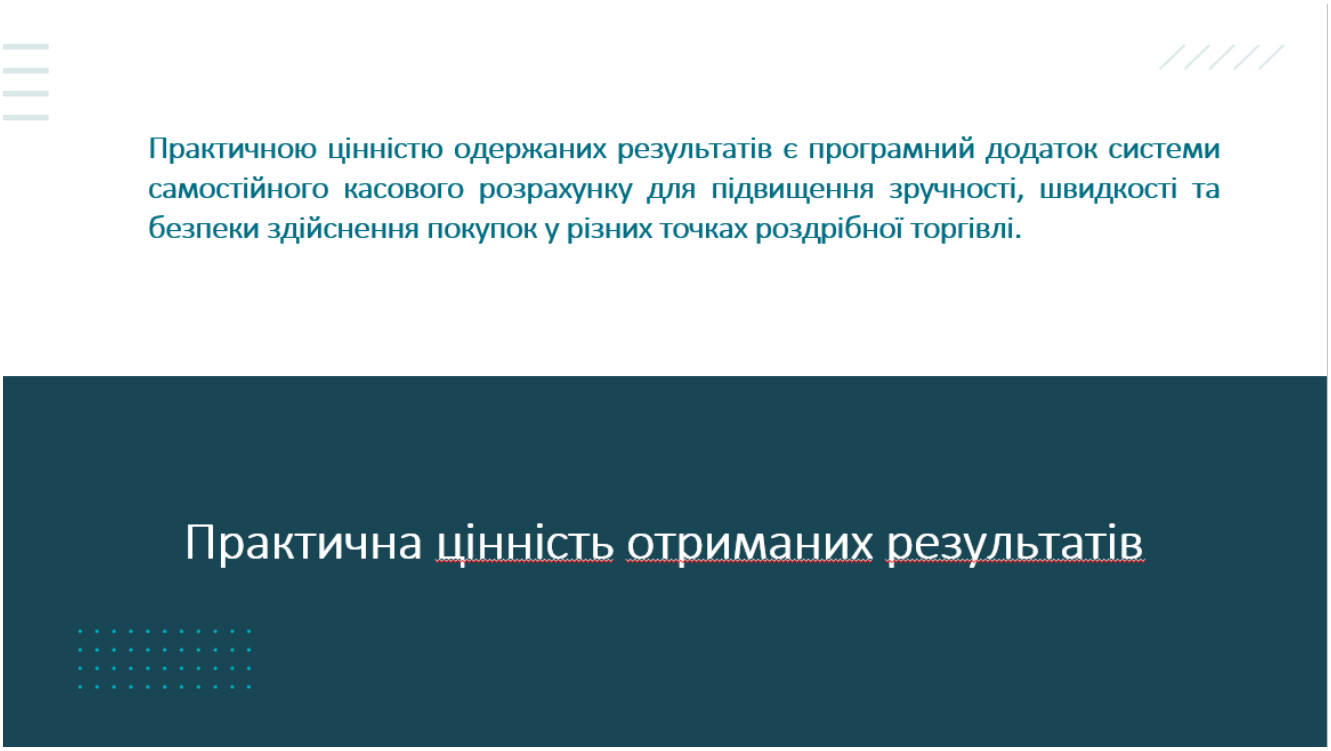

Рисунок Г.6 – Практична цінність одержаних результатів

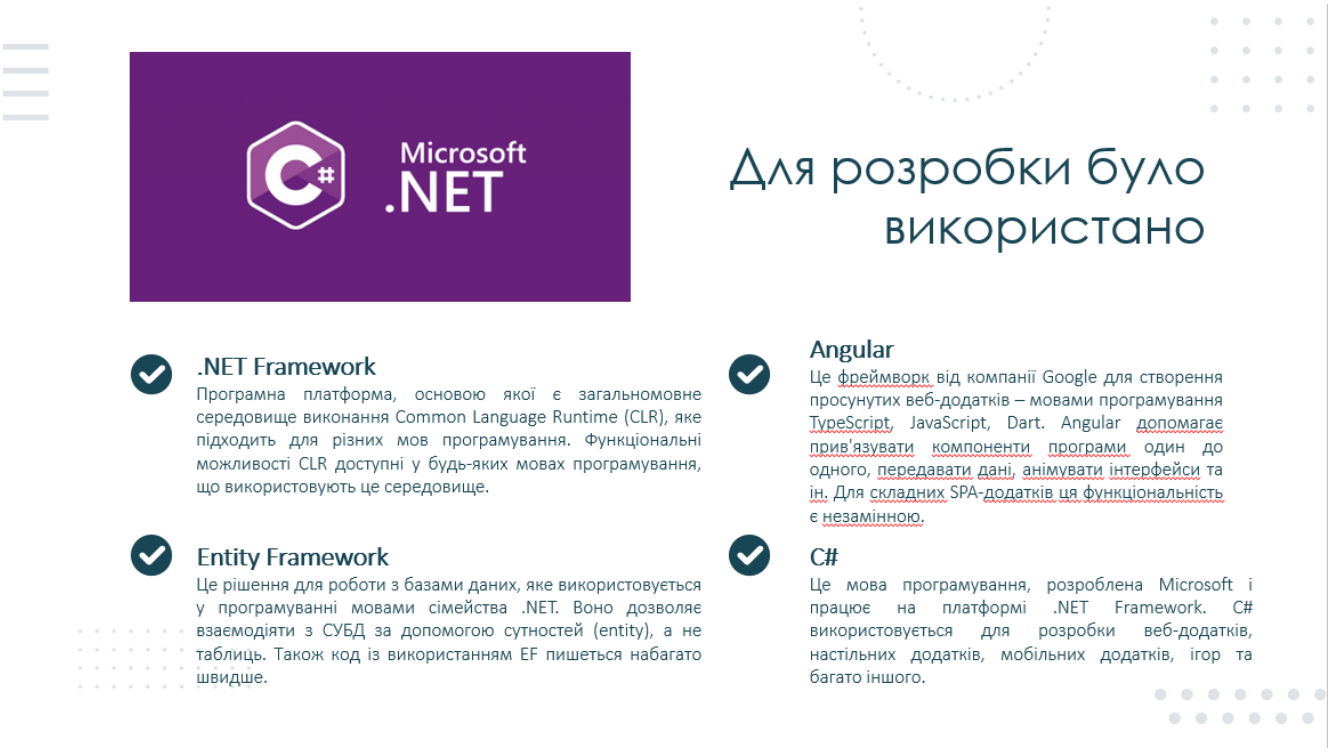

Рисунок Г.7 – Використані фреймворки

Рисунок Г.8 – Використання архітектури Onion

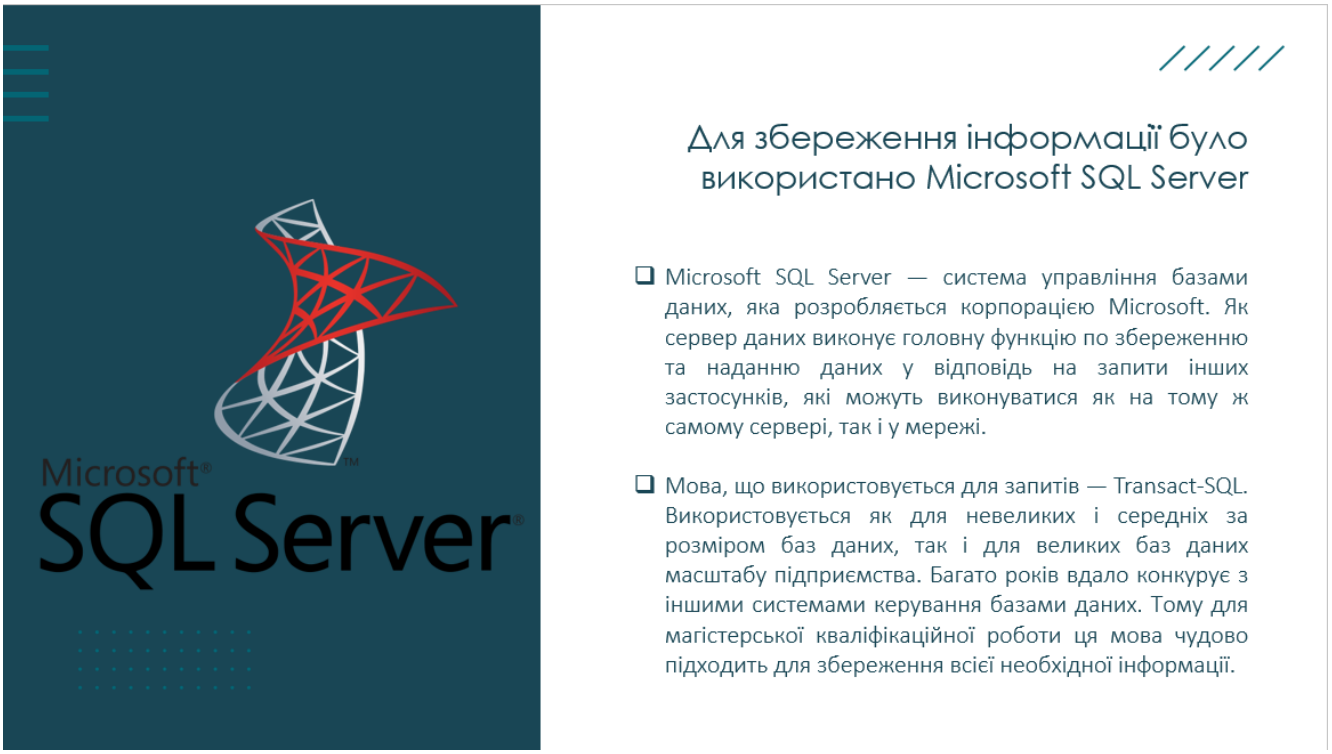

Рисунок Г.9 – Використання Microsoft SQL Server

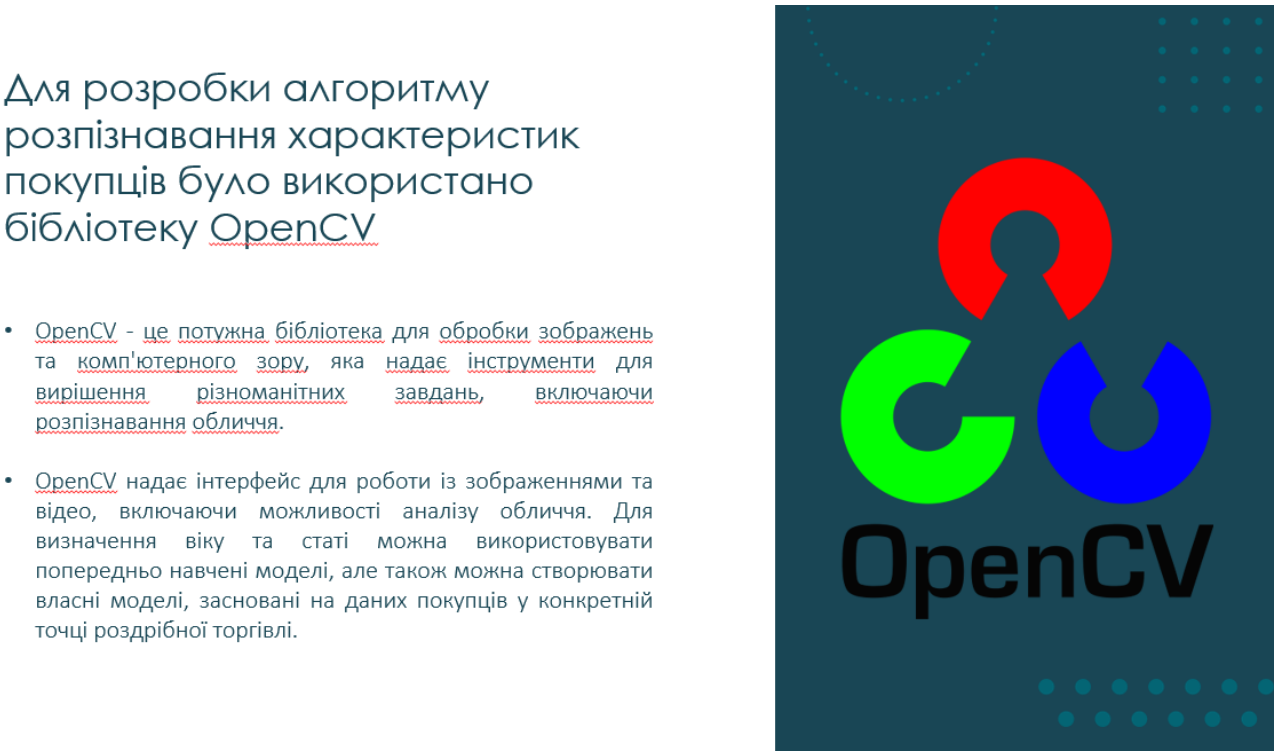

Рисунок Г.10 – Використання бібліотеки OpenCV

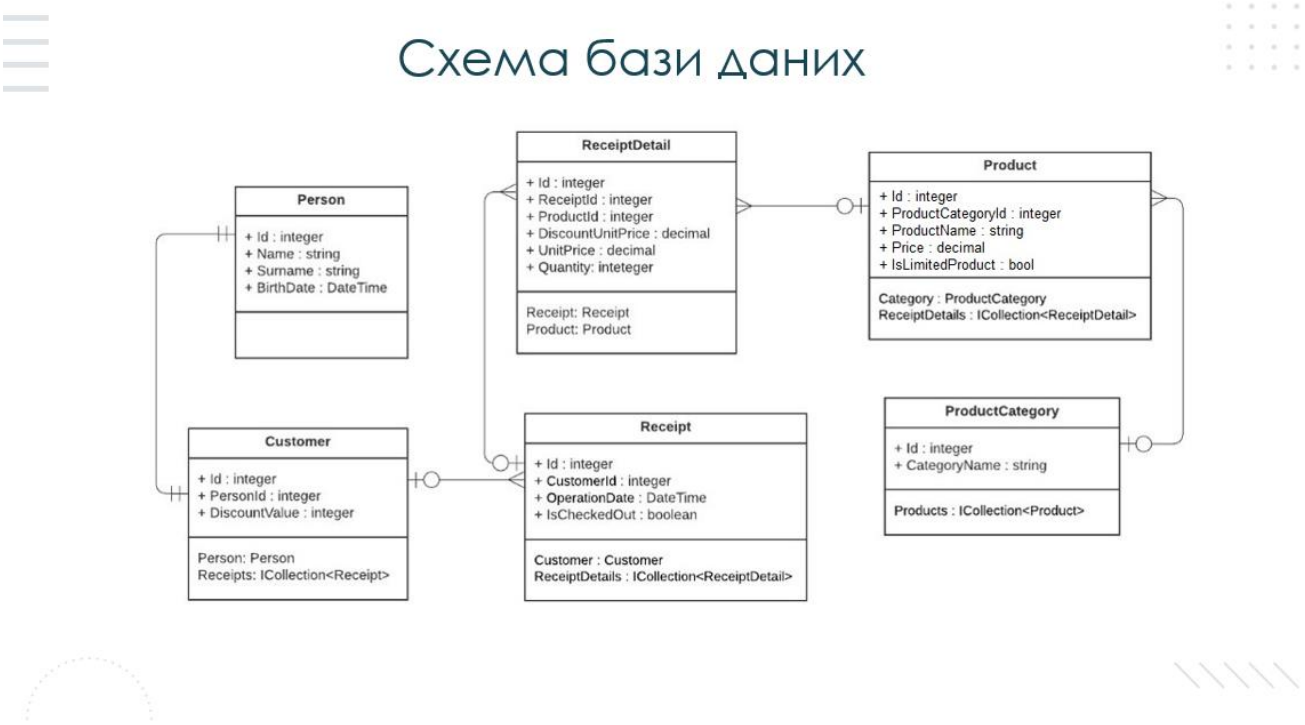

Рисунок Г.11 – Схема бази даних

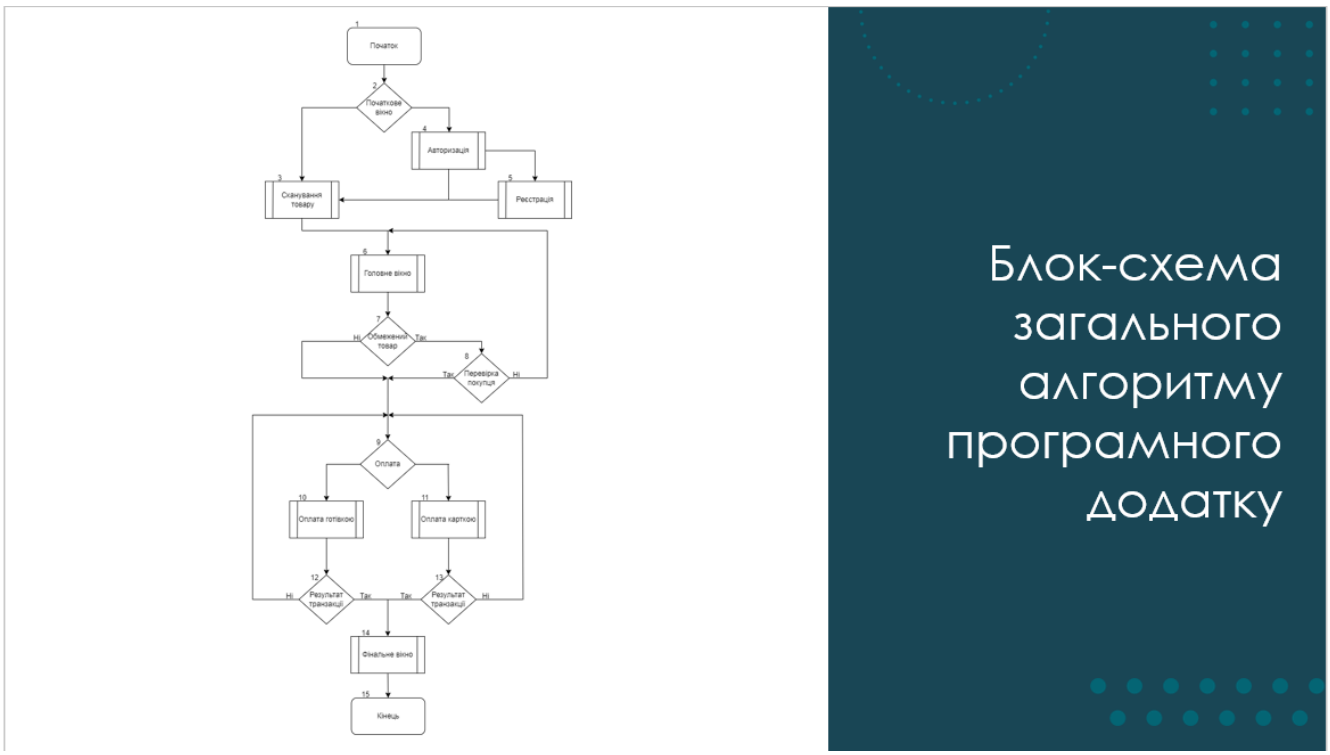

Рисунок Г.12 – Блок-схема загального алгоритму

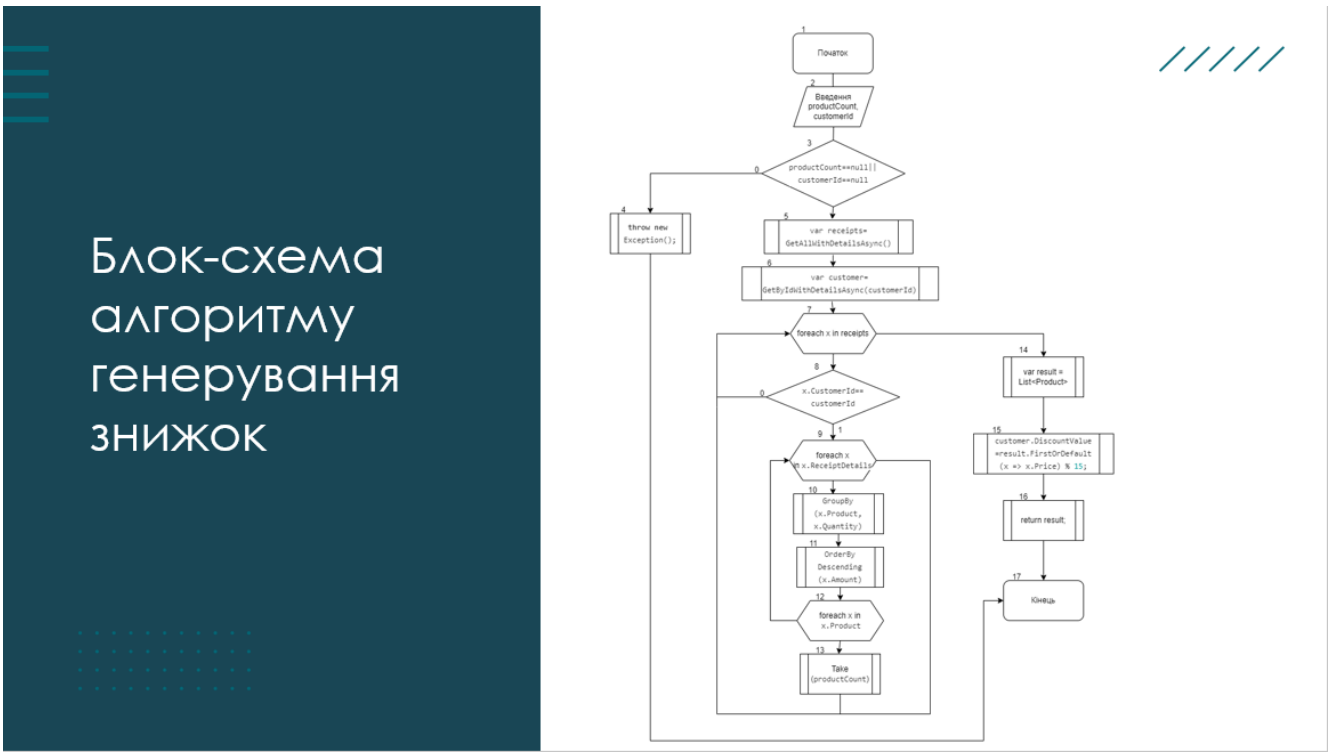

Рисунок Г.13 – Блок-схема алгоритму генерування знижок

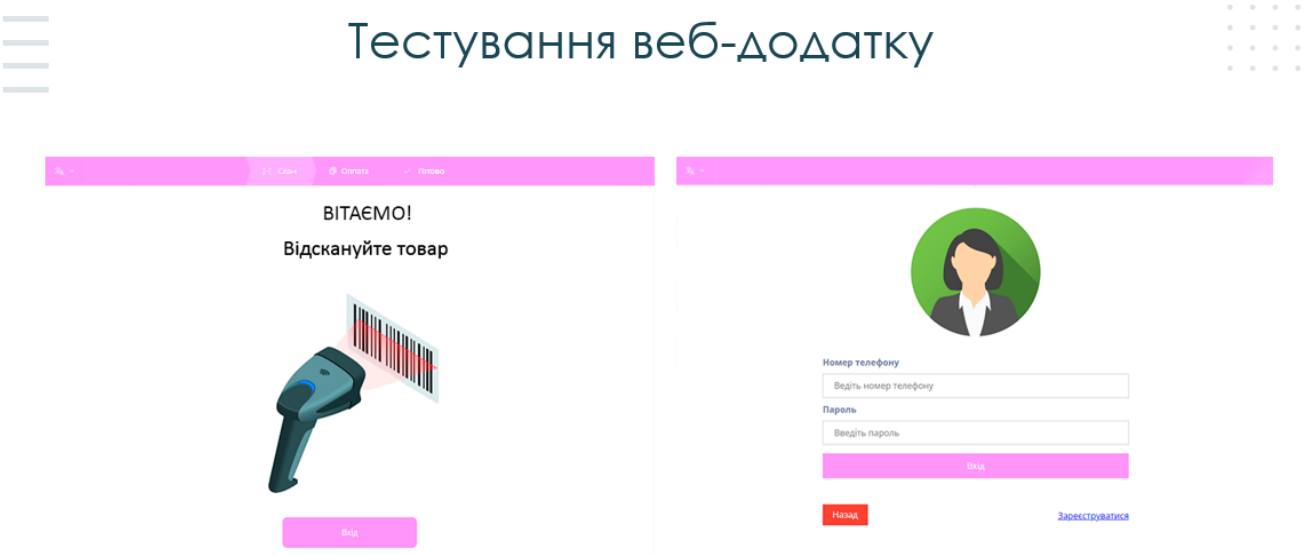

Рисунок Г.14 – Тестування системи

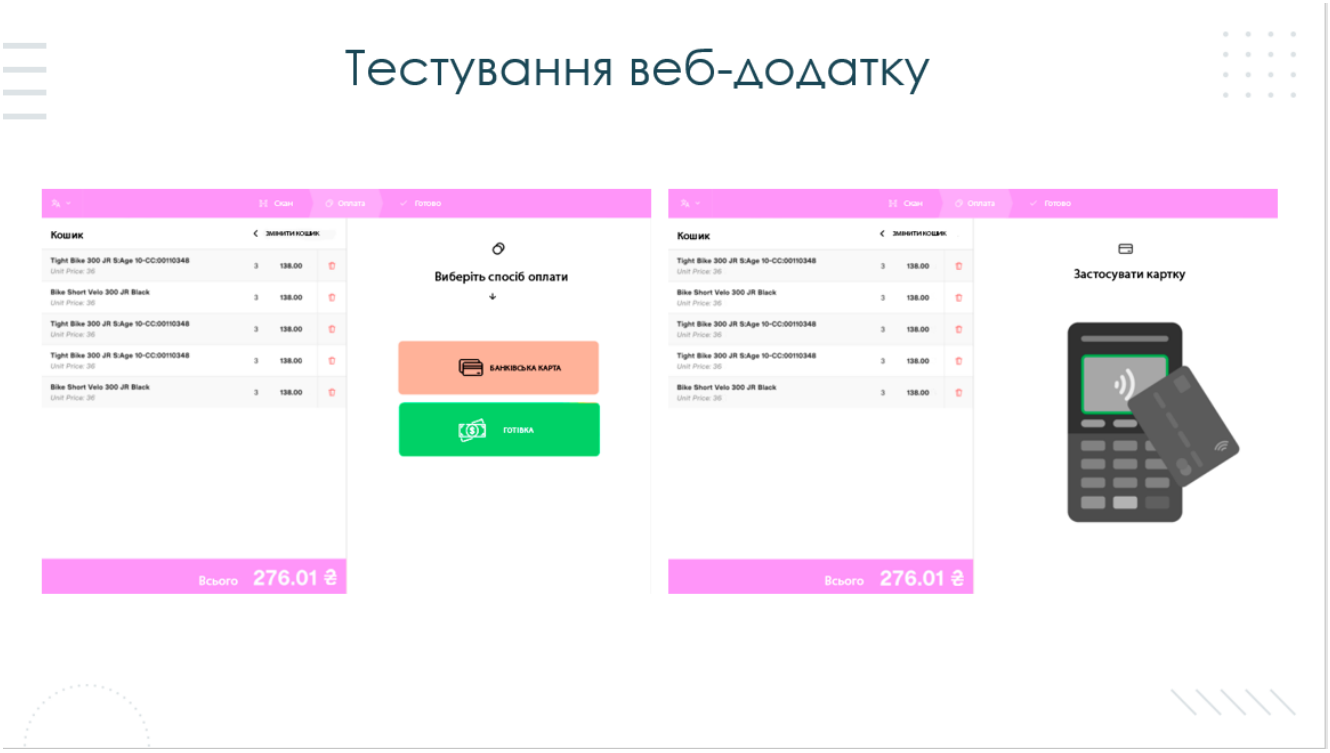

Рисунок Г.15 – Тестування системи(продовження)

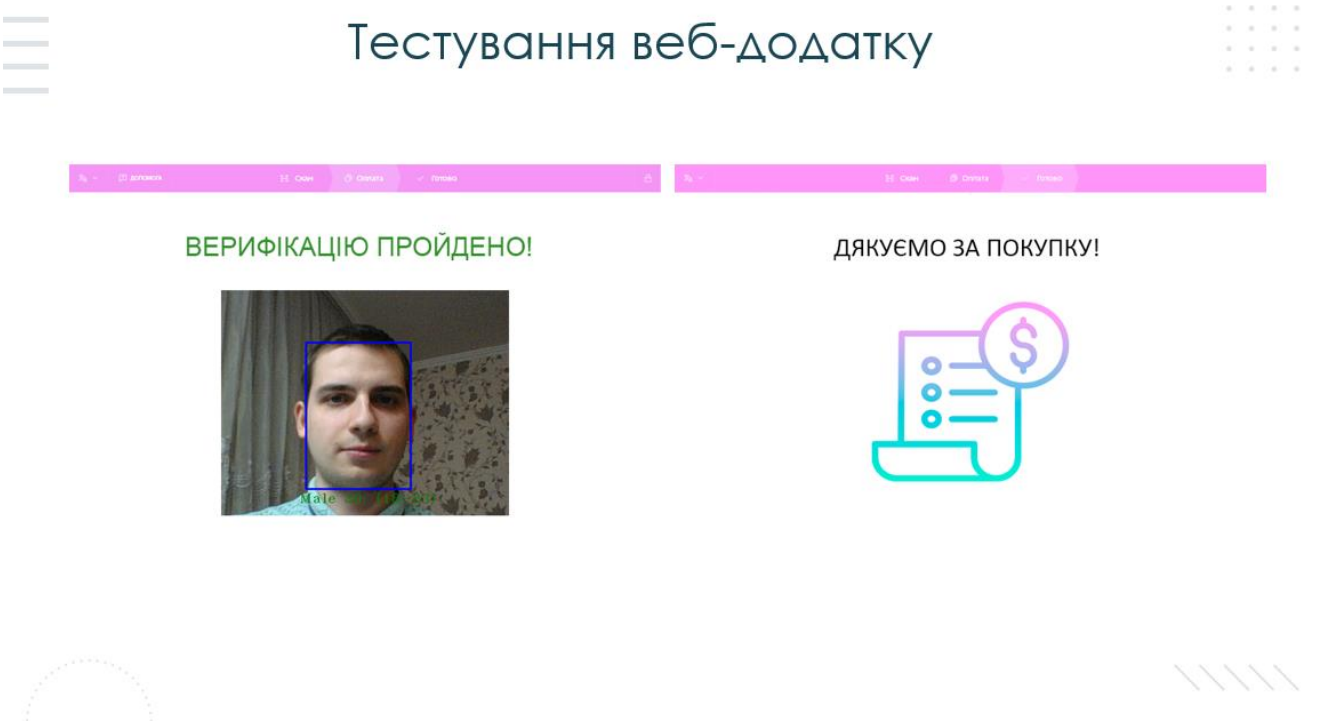

Рисунок Г.16 – Тестування системи(продовження)

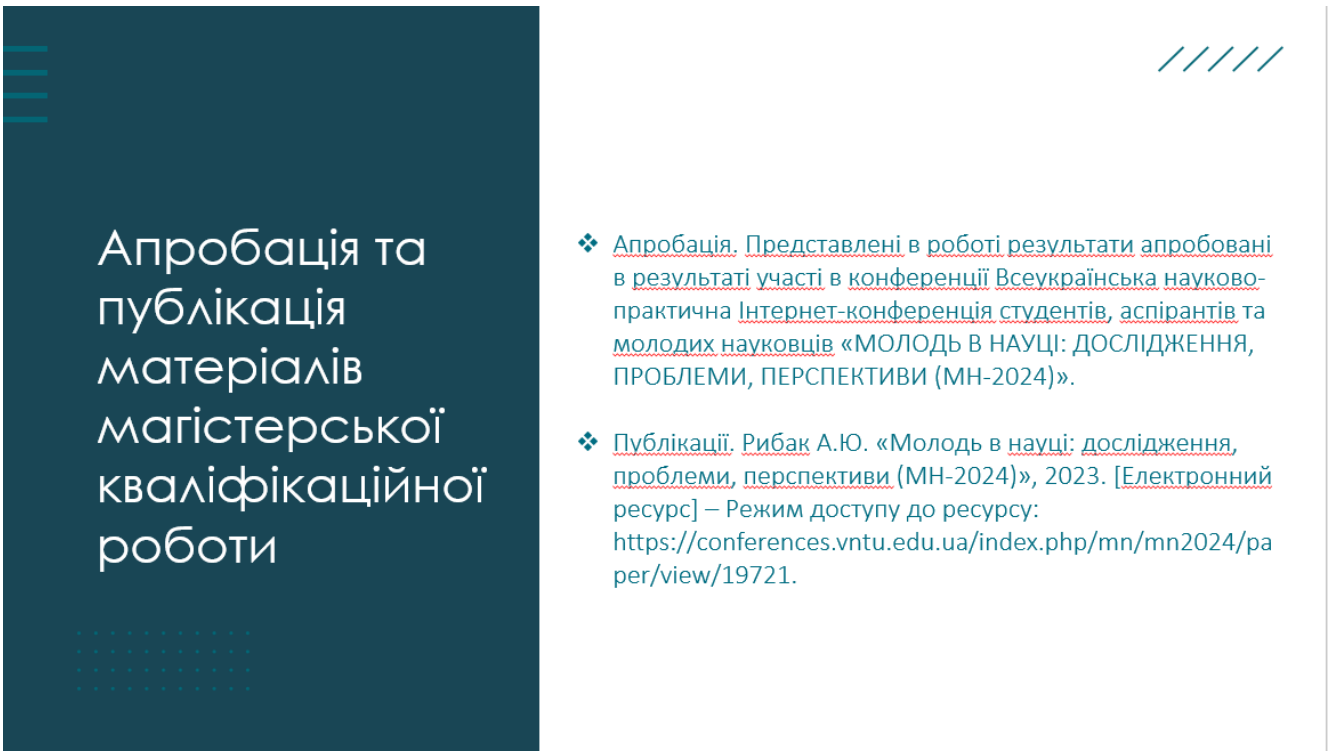

Рисунок Г.17 – Апробація та публікація результатів роботи

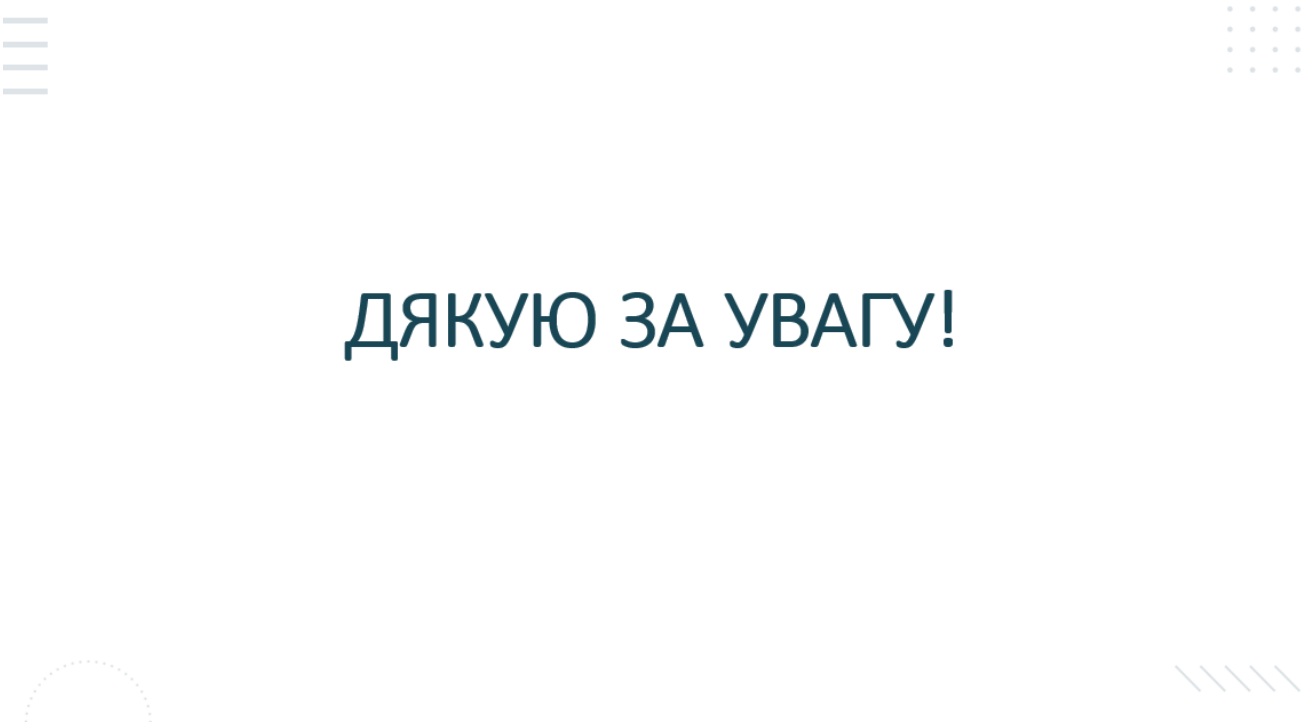

Рисунок Г.18 – Фінальний слайд