Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Гейміфікована система лояльності для супермаркетів на основі обробки чеків за допомогою штучного інтелекту з використанням векторного пошуку
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-22-2
Владислав ЄСИПЕНКО
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник проф. Лариса ЧАЛА
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Єсипенку Владиславу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Гейміфікована система лояльності для супермаркетів на основі обробки чеків за допомогою штучного інтелекту з використанням векторного пошуку

затверджена наказом університету від 5 червня 20 26 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 8 липня 20 26 р.

3. Вихідні дані до роботи Публікації з тематики гейміфікованих систем лояльності, програм лояльності європейських супермаркетів і платформ сканування касових чеків; науково-технічні джерела з методів оптичного розпізнавання тексту, корекції розпізнаного тексту великими мовними моделями і векторного семантичного пошуку; технічна документація сервісів платформи Google Cloud Platform; тестова вибірка касових чеків з тринадцяти країн Європи

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Методи інтелектуальної обробки касових чеків
3) Проєктування і програмна реалізація системи на платформі Google Cloud Platform
4) Експериментальне дослідження точності, продуктивності і вартості розробленої системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі	11.06.2026	виконано
3	Огляд програм лояльності і платформ	12.06.2026	виконано
	сканування чеків		
4	Проєктування архітектури системи на Google	14.06.2026	виконано
	Cloud Platform		
5	Розробка пакетного і реальночасного конвеєрів	16.06.2026	виконано
	обробки чеків		
6	Реалізація підсистеми безпеки і виявлення	18.06.2026	виконано
	шахрайства		
7	Розробка демонстраційного веб-інтерфейсу	20.06.2026	виконано
8	Експериментальне дослідження точності,	22.06.2026	виконано
	продуктивності і вартості		
9	Аналіз результатів досліджень	23.06.2026	виконано
10	Оптимізація параметрів моделей	23.06.2026	виконано
11	Підготовка презентаційних матеріалів	24.06.2026	виконано
12	Тестування системи	24.06.2026	виконано
13	Написання пояснювальної записки	25.06.2026	виконано
14	Попередній захист	05.07.2026	виконано
15	Захист перед екзаменаційною комісією	08.07.2026	

Дата видачі завдання 8 червня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Лариса ЧАЛА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 118 с., 27 рис., 8 табл., 1 дод., 60 джерел.

ВЕКТОРНИЙ ПОШУК, ГЕЙМІФІКАЦІЯ, КАСОВИЙ ЧЕК, ЛОЯЛЬНІСТЬ, ОПТИЧНЕ РОЗПІЗНАВАННЯ ТЕКСТУ, ПЕРСОНАЛІЗАЦІЯ, РОЗДРІБНА ТОРГІВЛЯ, СУПЕРМАРКЕТ, ХМАРНІ ОБЧИСЛЕННЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – процеси формування споживчої лояльності у роздрібних торговельних мережах та інструменти підвищення залученості покупців.

Предмет дослідження – моделі, методи і програмні засоби побудови гейміфікованих систем лояльності на основі AI-обробки касових чеків і векторного семантичного пошуку товарів.

Мета роботи – підвищення залученості покупців супермаркетів шляхом розробки гейміфікованої системи лояльності з автоматизованою обробкою касових чеків засобами Google Cloud Vision, корекцією моделями Gemini та векторним зіставленням товарів через Vertex AI Vector Search.

Методи дослідження – системний аналіз, порівняльний аналіз програм лояльності і OCR-сервісів, об’єктно-орієнтоване проєктування, методи машинного навчання і математичної статистики.

Спроєктовано і реалізовано гібридну архітектуру на платформі Google Cloud Platform, що поєднує пакетний і реальночасний конвеєри обробки чеків з корекцією помилок моделями Gemini та векторним пошуком товарів. Експериментально підтверджено зниження вартості обробки одного чека на вісімдесят відсотків порівняно з комерційними OCR-сервісами та середній час обробки два-три секунди. Наукова новизна полягає у гібридній архітектурі з пакетним і реальночасним конвеєрами, інтелектуальною корекцією OCR і векторним семантичним пошуком.

ABSTRACT

Bachelor's thesis contains: 118 pp., 27 fig., 8 tabl., 1 ann., 60 references.

ARTIFICIAL INTELLIGENCE, CASHIER RECEIPT, CLOUD COMPUTING, GAMIFICATION, LOYALTY, OPTICAL CHARACTER RECOGNITION, PERSONALIZATION, RETAIL, SUPERMARKET, VECTOR SEARCH.

Object of research – the processes of forming consumer loyalty in retail chains and tools for increasing customer engagement.

Subject of research – models, methods and software tools for building gamified loyalty systems based on AI processing of cashier receipts and vector semantic search of products.

The aim is to increase supermarket customer engagement through a gamified loyalty system that uses automated receipt processing by Google Cloud Vision, correction by Gemini models, and vector matching via Vertex AI Vector Search.

Research methods – systems analysis, comparative analysis of loyalty programs and OCR services, object-oriented design, machine learning and mathematical statistics methods.

The work implements a hybrid architecture on Google Cloud Platform combining batch and real-time receipt processing pipelines with Gemini-based OCR error correction and vector semantic product search. Experiments confirm an eighty percent reduction in per-receipt processing cost compared to commercial OCR services and a mean processing time of two to three seconds. The scientific novelty lies in the hybrid architecture combining batch and real-time pipelines with intelligent OCR correction and vector semantic search.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 10
1 Аналіз предметної галузі та постановка задачі 12
1.1 Опис предметної галузі 12
1.2 Опис проблем та викликів 14
1.3 Огляд існуючих застосунків і платформ 17
1.4 Інноваційність та переваги розроблюваної системи 25
1.5 Постановка задачі 28
2 Методи інтелектуальної обробки касових чеків 31
2.1 Огляд існуючих досліджень у галузі AI-обробки касових чеків 31
2.2 Методи оптичного розпізнавання тексту для касових чеків 33
2.3 Методи корекції розпізнаного тексту великими мовними моделями 37
2.4 Методи векторного семантичного пошуку 40
2.5 Методи виявлення шахрайства при обробці касових чеків 45
2.6 Хмарні платформи обробки даних 54
3 Проєктування і розробка системи 58
3.1 Проєктування системи 58
3.2 Структура програмного коду і модель розгортання 61
3.3 Головний сервіс пакетної обробки чеків 63
3.4 Сервіс семантичного зіставлення товарів з каталогом 66
3.5 Сервіс обробки у режимі реального часу чеків 71
3.6 Модель даних BigQuery і сервіс побудови агрегатів 74
3.7 Гейміфікаційні механіки і алгоритм нарахування винагород 82
3.8 Підсистема безпеки і програмна реалізація виявлення шахрайства .. 85
3.9 Веб-інтерфейс завантаження чека з інфографікою конвеєра 90
4 Експериментальні дослідження 94
4.1 Оцінювання точності розпізнавання касових чеків 94
4.2 Дослідження продуктивності і пропускної здатності 98

4.3 Оцінювання якості векторного зіставлення товарів 100
4.4 Перспективи розвитку 104
Висновки 107
Перелік джерел посилання 110
Додаток А Відомість кваліфікаційної роботи 118

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

ANN – Approximate Nearest Neighbor – наближений пошук найближчого сусіда;

API – Application Programming Interface – програмний інтерфейс застосунку;

BigQuery – хмарне сховище аналітичних даних Google Cloud Platform;

CER – Character Error Rate – частка помилок розпізнавання на рівні символу;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

CPG – Consumer Packaged Goods – товари повсякденного попиту в упаковці;

DCT – Discrete Cosine Transform – дискретне косинусне перетворення;

FMCG – Fast Moving Consumer Goods – швидкооборотні товари;

GCP – Google Cloud Platform;

GDPR – General Data Protection Regulation – Загальний регламент захисту даних ЄС;

HNSW – Hierarchical Navigable Small World – ієрархічний граф навігаційних малих світів;

LLM – Large Language Model – велика мовна модель;

MRR – Mean Reciprocal Rank – середній обернений ранг;

OCR – Optical Character Recognition – оптичне розпізнавання тексту;

PQ – Product Quantization – добуткове квантування;

ScaNN – Scalable Nearest Neighbors – масштабоване наближене відшукання сусідів;

SDK – Software Development Kit – комплект засобів розробки;

SRM – Steganalysis Rich Model – статистична модель ознак стеганоаналізу;

TPS – Transactions Per Second – кількість транзакцій за секунду;

WER – Word Error Rate – частка помилок розпізнавання на рівні слова;

XGBoost – Extreme Gradient Boosting – бібліотека градієнтного бустингу.

ВСТУП

Європейський ринок роздрібної торгівлі продовольчими товарами у 2024 році характеризується високим рівнем насиченості, ціновою конкуренцією та фрагментованою прихильністю покупця. Традиційні програми лояльності окремих мереж не забезпечують виробникам брендів повноти даних про споживчу поведінку, оскільки кожна мережа збирає дані лише у межах власної інфраструктури. Гейміфікаційні платформи сканування чеків долають це обмеження через універсальний механізм, що працює з фіскальними документами будь-якої мережі, і побудовані на хмарних сервісах оптичного розпізнавання тексту, великих мовних моделях і векторному семантичному пошуку.

Актуальність роботи зумовлена тим, що жодне з розглянутих у роботі одинадцяти ринкових рішень не закриває повний цикл від завантаження фотографії чека до нарахування гейміфікованої винагороди з одночасним захистом даних виробника і контролем над економікою обробки.

Метою роботи є підвищення рівня залученості покупців супермаркетів і ефективності маркетингових кампаній виробників товарів повсякденного попиту шляхом розробки гейміфікованої системи лояльності, що використовує автоматизовану обробку касових чеків засобами Google Cloud Vision, корекцію результатів моделями сімейства Gemini та векторне зіставлення товарів через Vertex AI Vector Search.

Для досягнення мети вирішено задачі: дослідження проблематики споживчої лояльності; порівняльний аналіз програм лояльності і гейміфікованих платформ; аналіз методів OCR, корекції тексту і векторного пошуку; проєктування архітектури системи на Google Cloud Platform; розробка демонстраційного інтерфейсу; експериментальне дослідження точності, продуктивності і вартості.

Об’єктом дослідження є процеси формування споживчої лояльності у роздрібних торговельних мережах. Предметом дослідження є моделі,

методи і програмні засоби побудови гейміфікованих систем лояльності на основі AI-обробки касових чеків і векторного семантичного пошуку.

Методи дослідження: системний аналіз предметної області, порівняльний аналіз програм лояльності і комерційних OCR-сервісів, методи об’єктно-орієнтованого проєктування, машинного навчання, обробки природної мови і математичної статистики.

Наукова новизна полягає у тому, що вперше запропоновано гібридну архітектуру гейміфікованої системи лояльності, яка поєднує пакетний і реальночасний конвеєри обробки касових чеків, інтелектуальну корекцію помилок OCR великими мовними моделями і векторний семантичний пошук товарів за схожістю.

Практичне значення полягає у можливості впровадження архітектури у роздрібних мережах для підвищення частоти повторних покупок, використання виробниками для моніторингу ефективності промо-кампаній і застосування у мобільних фінансових застосунках.

Робота містить чотири розділи: аналіз предметної області, методи інтелектуальної обробки чеків, проєктування і розробка системи, експериментальні дослідження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Роздрібна торгівля продовольчими товарами у країнах Європейського Союзу формує істотну частину споживчого ринку. За даними Eurostat у 2024 році сукупний обіг супермаркетів і мереж дискаунтерів у двадцяти семи країнах ЄС перевищив один трильйон триста мільярдів євро, що становить близько восьми відсотків валового внутрішнього продукту регіону [1]. Середнє домогосподарство витрачає на продукти харчування і побутові товари від сімнадцяти до двадцяти двох відсотків сукупного споживчого бюджету, у Польщі і Румунії ця частка сягає двадцяти шести відсотків. У структурі роздрібного ринку домінують великі міжнародні мережі, серед яких Schwarz Group з брендами Lidl і Kaufland, Carrefour, Tesco, Ahold Delhaize, Biedronka, Rewe Group.

Конкуренція між мережами у сегменті продовольчих товарів досягла рівня насичення, за якого цінові промо-акції перестали працювати як стійкий інструмент утримання клієнта. Дослідження McKinsey з 2023 року [2] фіксує, що середній європейський покупець регулярно відвідує від трьох до п’яти торговельних мереж на місяць і обирає конкретний магазин виходячи з локації, наявності потрібної акції, асортименту свіжих товарів. Прихильність до однієї мережі на рівні окремого покупця залишається низькою навіть серед учасників програм лояльності. Згідно зі звітом Nielsen середньостатистичний покупець зберігає у гаманці від п’яти до дев’яти карток лояльності, з яких активно використовує не більше двох, а рівень залученості решти падає нижче десяти відсотків протягом перших шести місяців після реєстрації [3].

З економічної точки зору утримання наявного клієнта для супермаркету обходиться у п’ять до семи разів дешевше порівняно з залученням нового через рекламні канали. Показник довічної цінності

клієнта, відомий як Customer Lifetime Value, використовується мережами як головний індикатор ефективності маркетингових інвестицій. Точність розрахунку цього показника значною мірою залежить від повноти даних про поведінку покупця, однак у сегменті продовольчого ритейлу дані фрагментовані за межами окремої мережі. Мережа бачить покупки лише у власних магазинах і не має інформації про те, що покупець придбав у конкурента того самого дня, яку частку його бюджету займає продукція конкретного бренду, як він реагує на промо-кампанії інших виробників.

Окрему категорію стейкхолдерів утворюють виробники товарів повсякденного попиту сегменту CPG і FMCG, серед яких Nestlé, Unilever, Procter and Gamble, Mars Wrigley, PepsiCo. Ці компанії не контролюють точки продажу і не мають прямого каналу комунікації з кінцевим споживачем. Взаємодія з покупцем відбувається опосередковано через рекламу у медіа, через упаковку на полиці магазину, через періодичні акції, організовані спільно з торговельною мережею. Ефективність промо-кампаній виробники оцінюють непрямими методами, зокрема через агреговану звітність ритейлерів, вибіркові опитування, панельні дослідження NielsenIQ [3] і Kantar. Час від запуску кампанії до отримання аналітичних даних становить від чотирьох до дванадцяти тижнів, що позбавляє виробника можливості оперативно коригувати механіки акцій і розподіл бюджетів.

Після набуття чинності загального регламенту про захист даних GDPR у Європейському Союзі і поступового відмирання третьо-сторонніх файлів cookie у провідних браузерах маркетингова індустрія зосередилась на концепції перших сторонніх даних. Перші сторонні дані означають інформацію, отриману безпосередньо від споживача з його явної згоди у контексті конкретної взаємодії з брендом. Такі дані мають високу юридичну чистоту і точність, а також широке застосування у задачах персоналізації пропозицій і прогнозування відтоку клієнтів [4]. Збирання перших сторонніх даних у сегменті продовольчого ритейлу залишається відкритою

проблемою через відсутність природних цифрових точок контакту між виробником бренду і покупцем у момент придбання товару.

Паралельно з кризою традиційних програм лояльності на ринку з'явилась група застосунків нового покоління, що пропонують покупцю ігрову винагороду за сканування касового чека незалежно від мережі, у якій було здійснено покупку. Такі застосунки закривають дві задачі одночасно. Покупець отримує додаткову цінність без зміни звичного сценарію походу за продуктами, а виробник бренду отримує канал прямого збору даних про реальні покупки разом з точкою контакту для персоналізованих промо-пропозицій. Сканування касового чека виявилось єдиною фізичною точкою, що пов'язує покупку у будь-якому магазині з цифровою системою бренду без залежності від кооперації з ритейлером.

Потреба у платформах для AI-обробки касових чеків з гейміфікаційним шаром мотивації споживача зумовлена об'єктивним станом індустрії роздрібної торгівлі і фрагментацією даних про споживчу поведінку за межами окремої мережі.

1.2 Опис проблем та викликів

Побудова гейміфікованої системи лояльності на основі AI-обробки касових чеків стикається з групою технічних, економічних і юридичних проблем, які не мають готового галузевого розв'язку. Найвагоміша проблема полягає у низькій і нестабільній якості вхідного зображення касового чека. Чеки друкуються на термопапері, який вицвітає протягом кількох тижнів, часто фотографуються у приміщенні з поганим освітленням, складаються або мнуться до моменту сканування. Кожна торговельна мережа застосовує власний шаблон розташування реквізитів, скорочень позицій, форматування цін і знижок, через що системи OCR, навчені на загальних датасетах, демонструють середню точність розпізнавання рядків

товарів у діапазоні від сімдесяти до вісімдесяти п’яти відсотків без додаткової корекції [5].

Інша проблема пов’язана з інтерпретацією розпізнаного тексту. Назва товару у чеку зазвичай стиснена до двадцяти або тридцяти символів і містить абревіатури, внутрішні коди мережі, скорочення одиниць виміру. Рядок виду «НАП.PEPSI 0.5L PET MAX» у одній мережі і «Pepsi Max 500 ПЕТ» в іншій описують один товар, однак класичний лексичний пошук не дозволяє зіставити такі варіанти з записом у каталозі промо-кампанії виробника. Задача семантичного зіставлення ускладнюється наявністю регіональних назв, локальних брендів, приватних торгових марок мереж, а також змінами упаковки і форматів з часом.

Ще одна проблема формує економічний профіль системи. Комерційні сервіси OCR для чеків типу ABBYY FlexiCapture Receipts, Klippa DocHorizon, Veryfi Lens, Tabscanner тарифікуються у моделі оплати за кожен API-виклик, зі середнім прайсом від двох до восьми центів за чек залежно від обсягу договору [6]. Для масштабу гейміфікованої платформи з мільйонами чеків на місяць така модель зробила б собівартість однієї обробки співмірною з типовою винагородою покупцю, що нівелює економічну доцільність гри. Власна інфраструктура OCR на виділених серверах вимагає капітальних витрат на GPU-обладнання і постійної команди експлуатації, що недоцільно у стартовій фазі продукту з непередбачуваним профілем навантаження.

Наступна проблема стосується вимог до швидкості обробки. Гейміфікаційна механіка спирається на принцип негайної винагороди, описаний у моделі гачка Нір Еяля, за якою часова затримка між дією користувача і позитивним підкріпленням безпосередньо впливає на формування звички [7]. Пакетна обробка чека з часом відповіді у кілька годин прийнятна для задач бухгалтерського обліку, однак руйнує петлю залучення у грі. Технічний виклик полягає у побудові конвеєра з часом

відгуку у межах двох або трьох секунд, який одночасно утримує високу точність розпізнавання і зіставлення товарів з каталогом промо-кампаній.

Окрема проблема пов’язана із запобіганням зловживанням. Відкрита модель винагороди за завантажений чек створює стимул для кількох типів атак. Один тип це повторне завантаження того самого чека різними користувачами або з різних пристроїв одного користувача. Інший тип це подання синтезованих або відредагованих зображень чеків, які не відповідають реальним покупкам. Ще один тип це координовані кампанії ботів, що використовують реальні фотографії чеків з публічних джерел. Для захисту система повинна зберігати унікальні ознаки кожного чека, зокрема номер фіскального реєстратора, дату і час, суму, контрольний хеш зображення, та перевіряти їх на дублікати у реальному часі [8].

Ще одна проблема має юридичний характер. Касовий чек містить персональні дані у тлумаченні GDPR у випадках, коли його можна асоціювати з конкретним покупцем через програму лояльності, банківську картку або IP-адресу завантаження. Оператор системи зобов’язаний забезпечити мінімізацію збору, цільове використання, право на видалення і переносимість даних, безпечне зберігання у юрисдикції ЄС. Додатково кілька країн ЄС, серед яких Італія, Польща і Угорщина, запровадили національні фіскальні вимоги до архівування чеків і можуть розглядати сторонні додатки, що обробляють фіскальні документи, як операторів фіскальних даних з відповідним регуляторним навантаженням [9].

Остання проблема характерна для всього класу гейміфікованих застосунків і пов’язана з поступовим зниженням залученості користувача після перших тижнів використання. Дослідження Gartner з питань гейміфікації [10] фіксує, що від шістдесяти до сімдесяти відсотків ініціатив з ігровою винагородою не досягають запланованих показників утримання через одноманітні механіки, повторювані типи завдань, відсутність соціального шару і персоналізації. Побудова стійкої мотиваційної петлі потребує комбінації різних типів винагород, сезонних подій, прогресії

рівнів, елементів змагальності з друзями, що додає вимог до архітектури бекенду і системи аналітики поведінки.

Сукупність зазначених проблем формує технічні і економічні обмеження, у межах яких має працювати архітектура системи. Відповідь на ці обмеження лежить у площині хмарних технологій з моделлю оплати за фактичне використання, інтелектуальної корекції розпізнаного тексту великими мовними моделями і семантичного пошуку товарів за векторною схожістю.

1.3 Огляд існуючих застосунків і платформ

Для обґрунтування архітектурних рішень, закладених у роботу, проведено аналіз трьох категорій ринкових продуктів, які частково перетинаються з цільовою функцією запропонованої системи. Одна категорія охоплює традиційні програми лояльності торговельних мереж з власним мобільним застосунком. Інша категорія охоплює гейміфіковані застосунки сканування чеків з винагородою за покупки у будь-якій мережі. Ще одна категорія охоплює комерційні API для OCR касових чеків, що використовуються розробниками як готовий компонент обробки.

Tesco Clubcard є історично першою великою програмою лояльності у європейському продовольчому ритейлі, запущеною у 1995 році. За даними самої мережі програма охоплює понад двадцять мільйонів активних учасників у Великій Британії та Ірландії [11]. Учасник отримує картку або цифровий ідентифікатор у застосунку, який скануються на касі, після чого за кожен витрачений фунт нараховуються бали. Накопичені бали конвертуються у знижкові ваучери або у бонуси партнерських компаній. Модель даних програми замкнена у межах покупок, здійснених у магазинах Tesco і афілійованих брендів. Поведінка покупця у конкурентних мережах залишається для Tesco невидимою. Інтерфейс мобільного застосунку Tesco Clubcard наведено на рисунку 1.1.

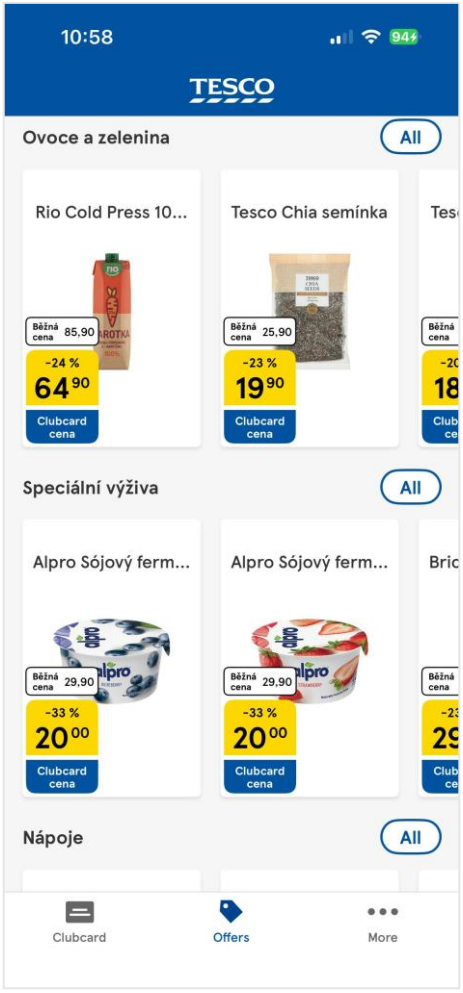


Рисунок 1.1 – Мобільний застосунок Tesco Clubcard

Lidl Plus є застосунком дискаунтера Schwarz Group, запущеним у 2019 році і розгорнутим у двадцяти з лишком країнах ЄС. Механіка ґрунтується на цифрових купонах, що активуються у застосунку і списуються під час сканування QR-коду на касі. Програма використовує персоналізовані пропозиції на основі історії покупок у мережі, однак аналогічно Clubcard не бачить поведінку покупця поза магазинами Lidl [12]. Подібну архітектуру мають застосунки Carrefour Fidelité у Франції, Biedronka Moja у Польщі (рисунок 1.3), Rewe Bonus Club у Німеччині. Усі вони розраховані на утримання клієнта всередині мережі і не передбачають співпраці з виробниками брендів у частині обміну поведінковими даними на рівні

окремого споживача. Інтерфейс застосунку Lidl Plus наведено на рисунку 1.2.

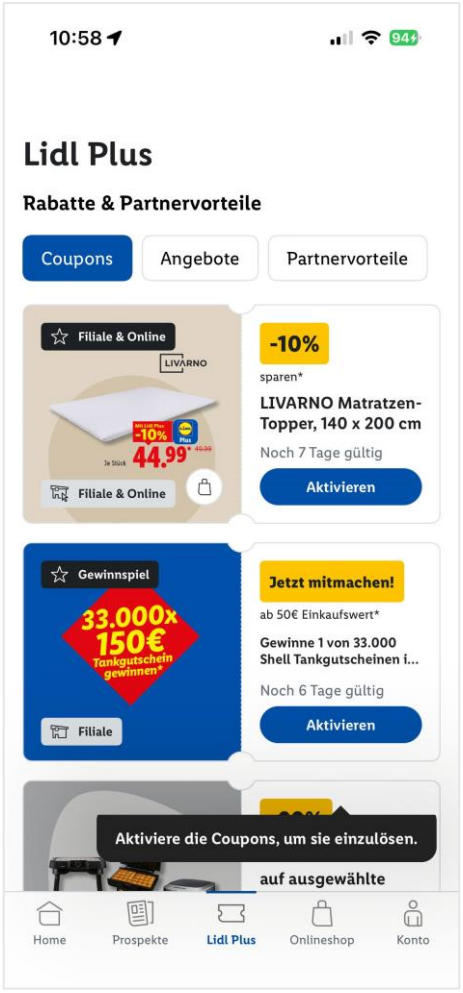


Рисунок 1.2 – Мобільний застосунок Lidl Plus

Ключове обмеження класу традиційних програм лояльності полягає у фрагментованому профілі покупця. Середньостатистичний споживач ЄС відвідує декілька мереж на місяць, тому жодна окрема програма не формує повної картини споживчого кошика. Виробники брендів сегменту CPG і FMCG не мають прямого доступу до персоніфікованих даних цих програм, оскільки ритейлер розглядає їх як конкурентну перевагу і не ділиться з постачальниками на рівні окремого чека. Інша категорія представлена гейміфікованими застосунками сканування чеків.



Рисунок 1.3 – Мобільний застосунок Biedronka Moja

Fetch Rewards є американським застосунком, що пропонує бали за сканування будь-якого продуктового, ресторанного або аптечного чека. Станом на 2024 рік застосунок має понад сімнадцять мільйонів активних користувачів на місяць і працює за моделлю прямих контрактів з виробниками брендів, які платять платформі за залученість споживачів до власних товарів [13]. Технологічно платформа базується на сторонніх OCR-сервісах і ручній верифікації частини чеків модераторами. Обмеження моделі полягає у відсутності розгорнутого каналу для європейських мереж і у залежності від комерційних OCR API з високою собівартістю обробки. Receipt Hog є застосунком компанії InfoScout, орієнтованим на збір

панельних даних про споживчі кошики для маркетингових агентств. Користувач отримує віртуальні монети або розіграші замість прямих знижок, а компанія продає знеособлені агреговані дані великим виробникам брендів. Фокус застосунку лежить на дослідницькій цінності даних, а не на прямій винагороді споживача, через що рівень залученості нижчий порівняно з Fetch Rewards. ShopKick застосовує гібридну механіку, що поєднує винагороду за відвідування магазину, сканування штрихкоду товару на полиці і завантаження чека. Платформа співпрацює з великими американськими ритейлерами, серед яких Walmart, Target, Best Buy. У європейському продовольчому сегменті застосунок представлений слабо, а інтеграція з виробниками CPG здійснюється через складну модель багаторівневих винагород, яка ускладнює масштабування на нові ринки. Klip є нідерландським стартапом, що позиціонує себе як європейський аналог Fetch Rewards і запущений у 2022 році. Застосунок підтримує чеки основних мереж Нідерландів і Бельгії, однак обмежений у підтримці інших країн ЄС і у спектрі інтегрованих брендів виробників.

Технологічно платформа використовує комбінацію хмарного OCR і ручної валідації позицій, що обмежує швидкість обробки і масштабованість. Загальне обмеження категорії гейміфікованих застосунків сканування чеків полягає у залежності від сторонніх OCR-постачальників, що диктує високу собівартість обробки, і у спрощеній архітектурі зіставлення товарів з каталогом промо-кампаній, заснованій переважно на точних збігах штрихкоду або назви.

Семантичний векторний пошук, здатний коректно обробляти варіативність написання позицій, у публічних описах архітектур згаданих застосунків не фігурує. Інтерфейс застосунку Fetch Rewards наведено на рисунку 1.4. Відсутність гнучких інтелектуальних алгоритмів у цих комерційних продуктах створює об'єктивні передумови для розробки більш досконалої архітектури.

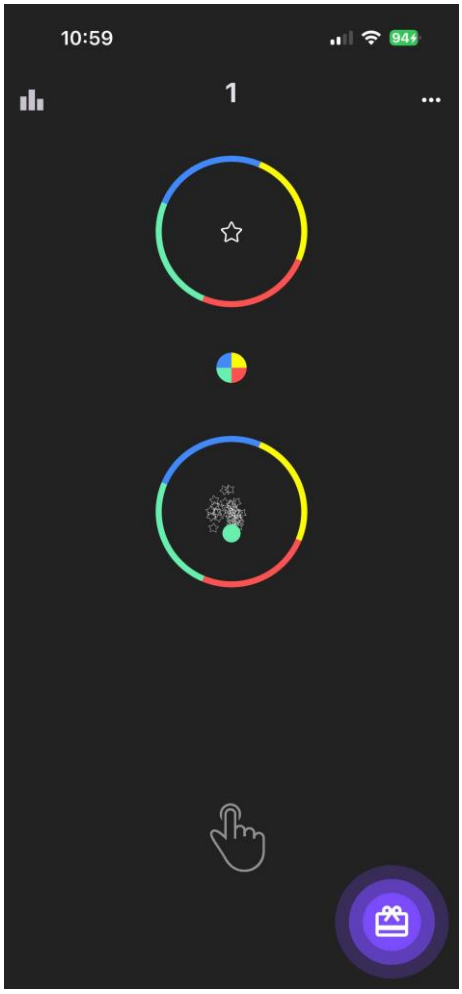


Рисунок 1.4 – Мобільний застосунок Fetch Rewards

Ще одна категорія охоплює комерційні API для OCR касових чеків. ABBYY FlexiCapture Receipts є продуктом компанії ABBYY, орієнтованим на корпоративних клієнтів у сегменті обліку витрат, бухгалтерії і експенс-менеджменту. API надає структуровану відповідь з рядками товарів, сумою, датою, ПДВ та реквізитами продавця, однак тарифікується за моделлю корпоративної ліцензії з мінімальним обсягом закупівлі, що непропорційно для задачі з мільйонами споживчих чеків [14]. Klippa DocHorizon є нідерландською платформою OCR для фінансових документів з підтримкою чеків, рахунків і банківських виписок. Платформа пропонує як хмарний API, так і розгортання у приватній інфраструктурі клієнта. Середня ціна обробки чека становить від трьох до шести центів за документ у

тарифах публічного прайс-листа [15]. Якість розпізнавання на європейських чеках знаходиться на прийнятному рівні, однак точність зіставлення товарів з довільним каталогом залишається за межами зони відповідальності API. Veryfi Lens пропонує API для OCR чеків з фокусом на американський ринок і мобільні інтеграції у застосунках експенс-менеджменту. Платформа підтримує OCR у реальному часі, однак тарифна модель будується на мінімальних щомісячних платежах і додаткових зборах за об’єм, що робить її непридатною для продукту з різкими піками навантаження під час промо-кампаній. Tabscanner є спеціалізованим API саме для роздрібних касових чеків з прайсом від двох до чотирьох центів за документ і підтримкою кількох десятків торговельних мереж. Обмеження полягає у відсутності інтегрованої корекції семантичних помилок OCR, у необхідності побудови окремого шару зіставлення товарів з каталогом промо, у залежності від сторонньої інфраструктури без контролю над моделями.

Для систематизації результатів аналізу та формування цілісного уявлення про стан ринку складено зведену порівняльну таблицю усіх розглянутих рішень за шістьма критеріями оцінювання. Набір критеріїв було сформовано виходячи з позиції та інтересів двох ключових зацікавлених сторін. З боку покупця визначальними факторами є широта покриття торговельних мереж, швидкість обробки завантаженого документа та наявність стійкої ігрової мотивації. З боку виробника бренду пріоритетними є повнота збору крос-мережевих даних про споживчий кошик та наявність прямого доступу до аналітичних звітів. Такий багатовимірний підхід дозволяє не лише об’єктивно оцінити поточний стан індустрії, але й виявити критичні прогалини у функціональності наявних комерційних продуктів. Отримані результати безпосередньо слугуватимуть теоретичним підґрунтям для обґрунтування інноваційної архітектури, запропонованої в межах даного дослідження. Зіставлення цих вимог з фактичними характеристиками наявних платформ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень у сфері лояльності і обробки чеків

Рішення	Географічне покриття	Вартість обробки / чек	Час обробки	Повнота даних про покупця	Гейміфікація	Доступ виробників до аналітики
Tesco Clubcard	Велика Британія, Ірландія	Вбудована (н/д)	Миттєво (каса)	Лише покупки Tesco	Відсутня	Немає
Lidl Plus	20+ країн ЄС	Вбудована (н/д)	Миттєво (каса)	Лише покупки Lidl	Купони	Немає
Carrefour Fidelité	Франція, Іспанія, Бельгія	Вбудована (н/д)	Миттєво (каса)	Лише покупки Carrefour	Купони	Немає
Biedronka Moja	Польща	Вбудована (н/д)	Миттєво (каса)	Лише покупки Biedronka	Купони	Немає
Fetch Rewards	США	~$0.05–0.08	15–60 с	Будь-які мережі (США)	Бали, рівні	Агрегований рівень
Receipt Hog	США, Канада	~$0.04–0.06	1–5 хв	Будь-які мережі (США)	Монети, розіграші	Знеособлені агрегати
Klip	Нідерланди, Бельгія	~$0.04–0.07	30–120 с	Обмежені мережі ЄС	Бали	Агрегований рівень
ABBYY FineReader Cloud	Глобально	$0.02–0.05	2–10 с	Відсутня (лише OCR)	Відсутня	Немає (API без продукту)
Klippa DocHorizon	Глобально	$0.03–0.06	2–8 с	Відсутня (лише OCR)	Відсутня	Немає (API без продукту)
Veryfi Lens	США, частково ЄС	$0.035–0.08	1–5 с	Відсутня (лише OCR)	Відсутня	Немає (API без продукту)

Продовження таблиці 1.1

Рішення	Географічне покриття	Вартість обробки / чек	Час обробки	Повнота даних про покупця	Гейміфікація	Доступ виробників до аналітики
Tabscanner	Глобально	$0.02–0.04	2–6 с	Відсутня (лише OCR)	Відсутня	Немає (API без продукту)
Запропонована система	12+ країн ЄС	~$0.0017	2–3 с	Будь-які мережі, повний кошик	Бали, місії, лідерборди	Персоніфікована аналітика

Як видно з таблиці 1.1, жоден з продуктів трьох категорій не закриває повний цикл від завантаження фотографії чека до нарахування гейміфікованої винагороди з одночасним захистом даних виробника бренду і контролем над економікою обробки. Традиційні програми лояльності забезпечують широке охоплення і миттєву касову інтеграцію, однак принципово обмежені одною мережею. Застосунки сканування чеків дають крос-мережеві дані, але страждають від повільної обробки і дорогої залежності від комерційних OCR-API. Спеціалізовані OCR-API вирішують технічний підзавдання, але не утворюють кінцевого продукту з гейміфікаційним шаром і прямим каналом до виробника. Саме ця ніша мотивує архітектурні рішення, запропоновані у подальших розділах роботи.

1.4 Інноваційність та переваги розроблюваної системи

Запропонована у кваліфікаційній роботі система гейміфікованої лояльності для супермаркетів реалізує шість принципових переваг порівняно з наявними на ринку рішеннями, описаними у попередньому підрозділі. Ці переваги стосуються архітектури обробки даних, економіки

експлуатації, швидкодії, простоти розгортання, цінності для виробників брендів і глибини споживчої аналітики.

Фундаментальною перевагою запропонованої архітектури є здатність побудувати єдиний профіль покупця незалежно від торговельної мережі, у якій здійснено покупку. Традиційні програми лояльності бачать поведінку клієнта виключно у власних магазинах, тому прихильність покупця до конкурентної мережі залишається поза їх аналітикою. Запропонована система через механізм сканування касового чека отримує крос-мережевий зріз споживчого кошика, що дозволяє оцінити справжню частку гаманця, виділену на продукцію конкретного бренду або категорії. Такий зріз неможливо отримати у моделях традиційних програм лояльності і він становить найціннішу частину даних з точки зору виробників товарів повсякденного попиту.

Ще однією перевагою є формування прямого каналу комунікації між виробником бренду і кінцевим споживачем без посередництва ритейлера. Виробник у традиційній моделі дистрибуції взаємодіє з покупцем опосередковано через рекламу, упаковку і спільні з ритейлером промо-акції, внаслідок чого час зворотного зв’язку від ринку до маркетингового відділу виробника складає від чотирьох до дванадцяти тижнів. Пропонована система скорочує цей час до одиниць секунд, оскільки факт покупки фіксується у момент сканування чека, а персоналізована пропозиція доставляється покупцю у тому самому застосунку відразу після нарахування винагороди.

Окрема перевага полягає у зниженні сукупної вартості обробки одного чека приблизно на вісімдесят відсотків порівняно з моделлю з використанням комерційних OCR API. Економія досягається кількома незалежними факторами. Один фактор полягає у використанні пакетного передбачення Vertex AI Batch Prediction з моделями сімейства Gemini Flash замість оплати за кожен синхронний виклик API типу ABBYY, Klippa, Veryfi і Tabscanner. Інший фактор пов’язаний із безсерверною моделлю

виконання на Cloud Functions і Cloud Run з тарифікацією за фактичний час обробки, що усуває витрати на простій виділеної інфраструктури. Ще один фактор спирається на застосування BigQuery VECTOR_SEARCH для пакетного зіставлення товарів замість окремої векторної бази даних з постійним абонентським обслуговуванням. Останній фактор полягає у автоматичному виборі методу пошуку залежно від обсягу запиту, що оптимізує співвідношення швидкості і вартості обробки [16].

Наступна перевага стосується швидкості обробки у режимі реального часу. Реалізований конвеєр забезпечує середній час обробки одного чека від двох до трьох секунд від моменту завантаження зображення до повернення користувачу списку нарахованих винагород. Цей показник досягнутий за рахунок паралельного виконання OCR через Cloud Vision API, корекції результатів мультимодальною моделлю Gemini і векторного пошуку через Vertex AI Vector Search endpoint. У порівнянні з початковою реалізацією пакетного режиму час відгуку скорочений приблизно у сім разів, що забезпечує негайне позитивне підкріплення споживача відповідно до вимог поведінкової моделі гейміфікації Hip Еяля [7].

Ще одна перевага полягає у простоті розгортання і масштабування. Архітектура побудована виключно на керованих сервісах Google Cloud Platform, серед яких Cloud Functions Gen2, Cloud Run, Firestore, BigQuery, Vertex AI. Відсутність серверів, оркестраторів контейнерів і баз даних з ручною експлуатацією скорочує операційні витрати і дозволяє розгортати нові екземпляри системи у різних регіонах у межах однієї робочої зміни. Автоматичне масштабування обчислювальних ресурсів відповідно до поточного навантаження усуває потребу у прогнозуванні піків і у відповідному резервуванні потужностей [17].

Остання перевага пов’язана з інтегрованою гейміфікаційною механікою, побудованою на принципах змінного підкріплення і прогресії рівнів. Система застосовує кілька типів винагород, серед яких прямі бали за кожен чек, бонуси за спеціальні комбінації товарів бренду у кошику, сезонні

квести з підвищеним коефіцієнтом нарахування, колекційні віртуальні значки за виконання довгострокових завдань. Така комбінація підтримує високу залученість користувача і знижує типове для галузі падіння активності після перших тижнів використання. Конкретні механіки і їх математична модель розглянуті у третьому розділі роботи.

Перелічені переваги у сукупності формують інноваційний характер запропонованого рішення. Наукова новизна роботи полягає у поєднанні гібридної пакетно-реальночасної обробки чеків, інтелектуальної корекції OCR великими мовними моделями і векторного семантичного пошуку товарів у єдиній архітектурі, що орієнтована на гейміфіковану лояльність і збір перших сторонніх даних для брендів у сегменті продовольчого ритейлу.

1.5 Постановка задачі

Проведений у попередніх підрозділах аналіз предметної галузі, переліку проблем і наявних застосунків дозволяє сформулювати постановку задачі кваліфікаційної роботи.

Об’єктом дослідження є процеси формування і підтримки споживчої лояльності у роздрібних торговельних мережах, системи маркетингових комунікацій супермаркетів зі споживачами і інструменти підвищення залученості покупців у сегменті продовольчої роздрібної торгівлі. Предметом дослідження є моделі, методи і програмні засоби побудови гейміфікованих систем лояльності, що ґрунтуються на автоматизованій AI-обробці касових чеків і векторному семантичному пошуку товарів у каталозі промоційних кампаній виробників брендів. Мета роботи полягає у підвищенні рівня залученості покупців супермаркетів і ефективності маркетингових кампаній виробників товарів повсякденного попиту шляхом розробки гейміфікованої системи лояльності, що використовує автоматизовану обробку касових чеків засобами Google Cloud Vision, корекцію результатів розпізнавання мультимодальними моделями

сімейства Gemini і векторне зіставлення товарів через Vertex AI Vector Search з пакетним доповненням засобами BigQuery.

Для досягнення поставленої мети у роботі розв’язуються шість взаємопов’язаних задач. Одна задача полягає у дослідженні проблематики споживчої лояльності у роздрібній торгівлі, обґрунтуванні вибору гейміфікаційного підходу як основного інструменту залучення покупців і формулюванні критеріїв оцінки ефективності таких систем. Інша задача полягає у проведенні аналізу наявних програм лояльності торговельних мереж і порівнянні їх з гейміфікованими платформами кешбеку і винагород за сканування чеків з точки зору цінності для споживача, виробника бренду і ритейлера. Ще одна задача полягає у здійсненні порівняльного аналізу методів оптичного розпізнавання тексту, методів корекції помилок розпізнавання і методів векторного пошуку за семантичною схожістю як технологічної основи для автоматизованого опрацювання касових чеків у масштабі масового застосунку. Окрема задача полягає у проєктуванні архітектури гейміфікованої системи лояльності на платформі Google Cloud Platform з конвеєром обробки чеків у пакетному режимі та режимі реального часу, з інтегрованою гейміфікаційною механікою і з підсистемою захисту від зловживань. Наступна задача полягає у розробці демонстраційного користувацького інтерфейсу завантаження чека з інфографікою руху даних системою і з візуалізацією механіки нарахування винагород для потреб презентації і відеозапису захисту кваліфікаційної роботи. Остання задача полягає у виконанні експериментального дослідження продуктивності, точності обробки чеків і ефективності запропонованих гейміфікаційних механік на репрезентативній вибірці реальних касових чеків, а також у формулюванні рекомендацій щодо подальшого розвитку системи.

У роботі застосовано методи системного аналізу для декомпозиції предметної галузі та виокремлення ключових проблем споживчої лояльності, методи порівняльного аналізу для оцінки наявних програм лояльності і комерційних OCR-сервісів, методи об’єктно-орієнтованого

проєктування для побудови архітектури програмної системи, методи машинного навчання для обробки зображень касових чеків і зіставлення товарів, методи математичної статистики для обробки результатів експериментального дослідження точності і продуктивності.

Очікувані результати роботи охоплюють теоретичну і практичну складові. Теоретична складова полягає у систематизації підходів до побудови гейміфікованих систем лояльності на основі AI-обробки касових чеків, у виявленні ключових обмежень наявних рішень і у формулюванні методологічних принципів подолання цих обмежень. Практична складова полягає у реалізованому програмному комплексі обробки касових чеків на платформі Google Cloud Platform, у демонстраційному веб-інтерфейсі з інфографікою потоку даних і у звіті з експериментальними оцінками точності, швидкодії та економічної ефективності обробки чеків.

2 МЕТОДИ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ КАСОВИХ ЧЕКІВ

2.1 Огляд існуючих досліджень у галузі AI-обробки касових чеків

Задача автоматизованого опрацювання касових чеків знаходиться на перетині кількох наукових напрямів, серед яких оптичне розпізнавання тексту, розуміння структурованих документів, обробка природної мови, семантичний пошук за векторними представленнями і поведінкова економіка у застосуванні до гейміфікованих систем мотивації.

Історично задача розпізнавання тексту у чеках розв’язувалась адаптацією загальних OCR-систем типу Tesseract, заснованих на рекурентних нейронних мережах з механізмом CTC-декодування. Робота Smith et al. [18] описала архітектуру Tesseract LSTM і її точність на документах з друкованим шрифтом, де рівень посимвольних помилок на чистих сканованих документах знаходився у межах 2–4%. Для касових чеків з термопаперу точність знижувалась до 7–10% через розмиття символів і фонові артефакти. У 2019 році Xu et al. [19] запропонували архітектуру LayoutLM, що об’єднує текстові, візуальні і просторові ознаки токенів для задач розуміння ділових документів [20]. Наступна версія LayoutLMv3 показала на бенчмарку SROIE для чеків середній F1-показник витягування полів на рівні 93%, що значно перевершило чисті OCR-рішення. Недоліком підходу залишається необхідність значної кількості розмічених даних для навчання, що не завжди доступне для нових торговельних мереж і регіональних форматів чеків. Робота Kim et al. [21] представила модель Donut, яка розпізнає структурований вміст документа без проміжного кроку OCR, застосовуючи енкодер-декодер трансформер безпосередньо до зображення. Цей підхід зняв залежність від якості попереднього етапу OCR, однак вимагає великих обчислювальних ресурсів для висновку і не досяг комерційного застосування у задачах масової обробки споживчих чеків.

Помилки OCR у реальних чеках часто мають регулярну структуру, коли однакові символи або пари символів стабільно переплутуються. Робота Bast and Korzen [22] систематизувала такі помилки і запропонувала статистичну модель виправлення на основі мовної моделі n-грам. Сучасніший напрям спирається на великі мовні моделі у ролі агентів постобробки. Робота Achiam et al. [23] показала, що GPT-4 у режимі few-shot досягає точності виправлення 87% на датасеті з зіпсованого OCR, перевищуючи спеціалізовані seq2seq моделі, навчені під конкретний домен. Аналогічні результати отримані для моделей сімейства Gemini від Google, що підтверджує загальну тенденцію переходу від вузькоспеціалізованих моделей до універсальних LLM-коректорів.

Задача зіставлення позиції з чека з товаром у каталозі промо-кампанії є окремим випадком задачі ентити-лінкінгу у відкритих доменах. Робота Reimers and Gurevych [24] запропонувала архітектуру Sentence-BERT для побудови щільних векторних представлень речень з підтримкою семантичної схожості. Вектори такої моделі придатні для пошуку методом наближеного k найближчих сусідів у алгоритмах типу HNSW, описаному у роботі Malkov and Yashunin [25], або ScaNN від Google Research [26]. Для каталогів товарів масштабу сотень тисяч одиниць сучасні індекси забезпечують пошук з середньою затримкою менше 10 мілісекунд і точністю recall@10 вище 95%. Дослідження Johnson et al. [27] у рамках бібліотеки FAISS систематизувало підходи до стиснення векторів через квантування добутків і кластеризацію, що дозволяє зберігати мільярдні індекси у пам’яті одного сервера. Ці підходи лежать в основі керованих сервісів векторного пошуку, серед яких Vertex AI Vector Search, Pinecone, Weaviate, Qdrant.

Теоретичний фундамент гейміфікації у застосунках масового споживання закладено у роботі Deterding et al. [28], де введено саме поняття gamification як використання елементів ігрового дизайну у неігрових контекстах. Системна мета-аналітична робота Hamari et al. [29] оцінила

вплив гейміфікаційних механік на поведінку користувачів у 47 емпіричних дослідженнях і показала статистично значущий позитивний ефект у задачах утримання і залученості, з особливо виразним впливом механік прогресії, змагальності і винагород. Робота Eyal [7] сформулювала модель гачка з 4 стадій, серед яких тригер, дія, змінна винагорода і інвестиція, що стала практичним фреймворком побудови мотиваційних петель у мобільних застосунках.

Огляд літератури показує, що кожен з технічних компонентів системи має ґрунтовне наукове підґрунтя і апробовані методи реалізації. Водночас відсутні системні публікації, що поєднують усі 4 напрями у єдину архітектуру, спеціалізовану під задачу гейміфікованої лояльності у продовольчому ритейлі. Кваліфікаційна робота заповнює цю прогалину інженерним синтезом і емпіричною валідацією архітектурного рішення на реальних даних.

2.2 Методи оптичного розпізнавання тексту для касових чеків

Оптичне розпізнавання тексту касового чека є базовим етапом конвеєра обробки і його якість визначає верхню межу точності всіх подальших етапів, включно зі зіставленням товарів і нарахуванням гейміфікованих винагород. У підрозділі виконано порівняльний аналіз чотирьох класів методів OCR у застосуванні до касових чеків і обґрунтовано вибір технологічної бази, використаної у роботі.

Базовою метрикою оцінювання якості OCR у роботі прийнято коефіцієнт посимвольних помилок Character Error Rate (CER), що визначається формулою (2.1) як відношення суми редакційних операцій до довжини еталонного тексту.

$$\mathrm{CER} = (\mathrm{S} + \mathrm{D} + \mathrm{I}) / \mathrm{N}, \qquad (2.1)$$

де S, D, I – кількість операцій заміни, видалення і вставки символів у розпізнаному тексті відносно еталонного;

N – кількість символів у еталонному тексті.

Найдовшу історію застосування у задачах розпізнавання мають класичні системи з відкритим кодом, серед яких Tesseract є найвідомішою реалізацією. Архітектура сучасної версії Tesseract 5 базується на двошаровій мережі LSTM з CTC-декодуванням і підтримує понад 100 мов, включно з українською [18]. Перевагою класичних систем є відсутність ліцензійних платежів і можливість локального розгортання. Недоліки проявляються на касових чеках з термопаперу, де середня точність за посимвольною метрикою CER знаходиться у межах 7–15% залежно від якості зображення, що вимагає значних обсягів ручної пост-корекції [30]. Паралельно до Tesseract розвивається сімейство нейромережевих систем з енкодер-декодер архітектурою, серед яких PaddleOCR від PaddlePaddle, EasyOCR на базі PyTorch, MMOCR від OpenMMLab. Ці системи демонструють точність на рівні 6–10% CER на чеках у хорошій якості, однак вимагають наявності GPU-інфраструктури для промислової експлуатації і підтримки команди інженерів для оновлення моделей під нові торговельні формати.

Інша група охоплює хмарні OCR-сервіси загального призначення, серед яких Google Cloud Vision API, AWS Textract, Azure AI Document Intelligence. Ці сервіси надають готовий OCR з підтримкою документів різних типів і тарифікуються за моделлю оплати за виклик. Google Cloud Vision API у режимі DOCUMENT_TEXT_DETECTION забезпечує середню точність CER на рівні 2–4% на чеках з якісних зображень і підтримує виявлення структурних блоків, що спрощує подальшу обробку [31]. Документована тарифікація складає 1,5 дол. США за 1000 викликів для DOCUMENT_TEXT_DETECTION, що для масового застосунку з мільйонами чеків формує істотну статтю витрат. AWS Textract пропонує аналогічний функціонал з додатковим API Analyze Expense,

спеціалізованим під чеки і рахунки. Середня точність витягування структурованих полів становить 85–90% на англомовних чеках і помітно нижча на європейських мовах зі складною діакритикою. Azure AI Document Intelligence надає готову модель Prebuilt Receipt з аналогічним функціоналом і обмеженнями, серед яких слабша підтримка кириличних і центральноєвропейських мов порівняно з Google Cloud Vision.

Ще одна група охоплює трансформерні моделі, навчені під задачу розуміння документів. Модель LayoutLMv3 від Microsoft об'єднує текстовий, візуальний і просторовий потоки ознак у єдиному енкодері і показує на бенчмарку SROIE точність витягування полів F1 на рівні 93% [21]. Модель Donut від Naver Clova виключає окремий етап OCR і працює з зображенням документа напряму через енкодер Swin і декодер BART, що усуває каскадні помилки OCR. Обидві моделі вимагають навчання або донавчання на датасеті конкретного домену, що обмежує їх застосовність у сценаріях з різнорідними форматами чеків. Модель TrOCR, розроблена Microsoft, базується на візуальному енкодері ViT і текстовому декодері RoBERTa і показує точність CER близько 3% на бенчмарку IAM для рукописного тексту. Для друкованих касових чеків показники знаходяться на рівні хмарних сервісів, однак обчислювальна вартість висновку значно вища, що робить пряме застосування TrOCR у масовому продакшені економічно невиправданим.

Окрему групу утворюють мультимодальні великі мовні моделі, серед яких Gemini від Google [37], GPT-4 від OpenAI, Claude Opus 4.7 від Anthropic. Ці моделі приймають зображення разом з текстовою інструкцією і повертають структуровану відповідь у форматі JSON або довільному текстовому вигляді. Для задачі обробки касових чеків мультимодальні LLM поєднують у собі функції OCR, структурного парсингу і корекції помилок, що спрощує архітектуру конвеєра і усуває потребу у окремих моделях для кожного етапу. Gemini 2.5 Flash демонструє на внутрішніх бенчмарках Google середню точність витягування структурованих полів чека на

рівні 94–96% і підтримує пакетне передбачення Vertex AI Batch Prediction зі значним дисконтом порівняно з синхронними викликами. Така модель ціноутворення робить мультимодальні LLM економічно вигідним варіантом для пакетної обробки великих обсягів чеків. GPT-4 від OpenAI досягає порівнянної точності, однак не пропонує аналогічного пакетного режиму з глибоким дисконтом. Claude Opus 4.7 від Anthropic демонструє високу якість розпізнавання і витягування структурованих полів, однак вартість виклику істотно вища за Gemini Flash, а нативна інтеграція з Vertex AI Batch Prediction відсутня, що ускладнює використання у пакетному режимі обробки великих обсягів чеків.

Для кваліфікаційної роботи обрано гібридну комбінацію з двох рівнів. Один рівень це Google Cloud Vision API у режимі DOCUMENT_TEXT_DETECTION, який застосовується для швидкого базового розпізнавання тексту чека у режимі реального часу з гарантованою низькою затримкою і передбачуваною вартістю. Інший рівень це моделі сімейства Gemini 2.5 Flash у двох режимах використання. Для пакетної обробки чеків застосовується Vertex AI Batch Prediction з економією близько 50% порівняно зі синхронними викликами. Для реальночасної корекції результатів Cloud Vision у сценаріях з високою варіативністю форматів застосовується синхронний виклик Gemini з налаштованою температурою і промптом, орієнтованим на структуровану відповідь у форматі JSON. Цей архітектурний підхід дозволяє оптимально розподілити обчислювальне навантаження. Завдяки цьому також досягається максимальна гнучкість при масштабуванні платформи. Окрім того, такий розподіл мінімізує ризики виникнення системних збоїв під час пікових навантажень. Це забезпечує додаткову стабільність функціонування всіх пов'язаних мікросервісів. Таке поєднання дає комбінацію мінімальної затримки у реальному часі, високої точності за рахунок корекції LLM і низької собівартості пакетної обробки, що відповідає сформульованим у підрозділах 1.2 і 1.4 вимогам до архітектури системи.

2.3 Методи корекції розпізнаного тексту великими мовними моделями

Результат роботи OCR-системи на касовому чеку зазвичай містить 10–30% рядків з помилками, які роблять пряме зіставлення позицій з каталогом промо-товарів неможливим без додаткового етапу корекції. Типовими є заміна окремих символів через дефекти термопаперу, злипання слів через недостатню роздільну здатність, пропуск літер зі складною діакритикою, хибне розпізнавання одиниць виміру і цін. У підрозділі виконано аналіз чотирьох класів методів корекції і обґрунтовано вибір технологічного рішення, прийнятого у роботі.

Історично першим класом методів є алгоритми на основі редакційної відстані. Відстань Левенштейна визначає мінімальну кількість операцій вставки, видалення і заміни символу, потрібних для перетворення одного рядка на інший [32]. Формальне обчислення відстані між рядками a і b довжини m і n задається рекурентним співвідношенням у формулі (2.2).

$$\begin{aligned} d(i,j) \;=\; \min\{\, d(i-1,j) \;+\; 1, d(i,j-1) \;+ \\ +\; 1, d(i-1,j-1) \;+\; \delta(a_i, b_j) \,\}, \end{aligned} \tag{2.2}$$

де $\delta(a_i, b_j)$ дорівнює 0, якщо символи рівні, і 1 у протилежному випадку; базові умови – $d(0, j) = j$ і $d(i, 0) = i$.

Відстань Дамерау–Левенштейна додає операцію перестановки сусідніх символів, що краще моделює клавіатурні друкарські помилки. Для пошуку найближчого слова у словнику застосовуються структури BK-tree і Symspell. Алгоритм Symspell виконує пошук у межах заданої редакційної відстані за субсекундний час для словників обсягом у сотні тисяч слів завдяки попередньо обчисленому індексу видалень. Обмеження класичних алгоритмів проявляється на задачах, де помилки не зводяться до локальних замін символів. У реальних чеках часто трапляється заміна цілого токена, перестановка слів у назві, додавання або видалення внутрішніх кодів

мережі. Редакційна відстань між «НАП.PEPSI 0.5L PET MAX» і «Pepsi Max 500 ПЕТ» у явному обчисленні дає значення, близьке до довжини рядка, що робить словниковий пошук непридатним без глибокої попередньої нормалізації.

Інший клас охоплює статистичні моделі контекстної корекції. Мовні моделі n-грам оцінюють ймовірність послідовності токенів і обирають з кандидатів той, що максимізує сукупну ймовірність у контексті сусідніх слів [33]. Для задач корекції OCR застосовувались триграмні моделі, навчені на корпусах ділових документів, що дозволяло виправляти помилки типу «Б замість В» у контексті конкретної фрази. Розширенням цього підходу є моделі умовних випадкових полів CRF, які комбінують ознаки окремих символів, їх позицій у рядку і контексту сусідніх токенів у єдиному оптимізаційному критерії. Слабка сторона статистичних моделей полягає у залежності від обсягу і якості тренувального корпусу. Для кожної нової мови, торговельної мережі або категорії товарів потрібно збирати і розмічати окремий корпус, що економічно невиправдано для масштабу масового застосунку, який одночасно працює з чеками десятків європейських мереж.

Ще один клас охоплює нейромережеві моделі послідовність-у-послідовність, побудовані на архітектурі енкодер-декодер з механізмом уваги. Роботи Schmaltz et al. [34] і Chollampatt and Ng [35] показали застосування таких моделей для задач виправлення граматичних помилок у тексті, з трансфером на задачу корекції OCR. Середня точність виправлення на датасетах типу ICDAR Post-OCR знаходиться у межах 75–85% залежно від мови і якості вхідних даних. Обмеженням цього класу є необхідність донавчання окремої моделі для кожного домену документів. Модель, навчена на англомовних рецептурних документах, демонструє різке падіння якості на українських або польських касових чеках з регіональними брендами. Вартість розмітки навчального корпусу з десятків тисяч пар неправильний і правильний рядок зіставна з вартістю використання готових

універсальних мовних моделей, що робить цей підхід невигідним для нових ринків.

Окремий клас утворюють великі мовні моделі сімейства GPT, Claude, Gemini, що застосовуються для корекції OCR без спеціалізованого донавчання. Основний принцип полягає у формулюванні інструкції, що описує задачу, додаванні кількох демонстраційних прикладів і передачі моделі сирого тексту чека з вимогою повернути структуровану відповідь у форматі JSON з виправленими полями.

Робота Boros et al. [36] порівняла декілька підходів до виправлення помилок розпізнавання на датасеті з європейських історичних документів і показала, що нейромережеві seq2seq моделі суттєво перевищують класичні словникові методи. Gemini 2.5 Flash у контексті задачі корекції чеків демонструє близькі показники якості за істотно нижчої вартості виклику. Модель підтримує нативний структурований висновок у форматі JSON через параметр response_schema, що усуває потребу у окремому етапі парсингу відповіді регулярними виразами. Пакетне передбачення через Vertex AI Batch Prediction дає дисконт близько 50% порівняно з синхронними викликами і робить модель економічно придатною для обробки великих обсягів накопичених чеків [16]. Claude Opus 4.7 від Anthropic демонструє вищу точність на складних випадках із змішаним набором мов в одному чеку, однак вартість виклику у перерахунку на один токен істотно вища і нативний пакетний режим у Vertex AI відсутній.

Для кваліфікаційної роботи обрано підхід з використанням Gemini 2.5 Flash у двох режимах. У синхронному режимі модель застосовується для корекції результату Cloud Vision у реальному часі з налаштованою температурою на рівні 0,1 і з промптом, що включає демонстраційні приклади пар сирий рядок і скоригована структурована відповідь. У пакетному режимі модель застосовується через Vertex AI Batch Prediction для масової обробки накопичених чеків з економією на вартості обробки. Промпт побудовано за шаблоном системна інструкція, яка фіксує очікувану

структуру JSON-відповіді з полями raw_items, corrected_items, shop_name, total_amount, receipt_date, та контекст конкретного чека з сирим OCR-текстом.

Для підвищення стабільності відповіді застосовано параметр response_schema з описом очікуваних типів полів. Поріг відхилення некоректних відповідей встановлено на рівні валідації JSON-структури і перевірки сум за позиціями відносно загальної суми чека з припустимою похибкою у 5%. Такий підхід усуває потребу у донавчанні власних моделей, забезпечує прийнятну точність на чеках різних європейських мереж і дає повну контрольованість співвідношення швидкості і вартості обробки.

2.4 Методи векторного семантичного пошуку

Після того як позиції касового чека скориговано на етапі обробки результатів OCR, виникає задача зіставлення кожної позиції з каталогом товарів, що беруть участь у активних промо-кампаніях. Лексичний пошук за точним збігом назви або штрихкоду розв’язує менш ніж половину реальних випадків через варіативність написання, регіональні скорочення, різну послідовність слів у назвах. Векторний семантичний пошук знімає ці обмеження шляхом переведення назви товару у векторне представлення і пошуку найближчих сусідів у просторі вкладень. Підрозділ розкриває математичний апарат, моделі побудови вкладень, алгоритми наближеного пошуку і промислові реалізації, на основі яких здійснено вибір технологічної бази.

Векторний простір вкладень будується відображенням рядка тексту у точку простору розмірності від 128 до кількох тисяч. Для оцінки схожості двох векторів у задачах семантичного пошуку найчастіше застосовується косинусна подібність, що обчислюється за формулою (2.3) як скалярний добуток векторів, поділений на добуток їх норм [38].

$$\mathrm{sim}(u, v) = (u \cdot v) / (\|u\| \cdot \|v\|) = \Sigma_i u_i v_i / \mathrm{sqrt}(\Sigma_i u_i^2) \cdot \mathrm{sqrt}(\Sigma_i v_i^2), \quad (2.3)$$

де u і v – вектори-вкладення порівнюваних текстів;

$\|\cdot\|$ позначає евклідову норму вектора.

Альтернативами є евклідова відстань і скалярний добуток у чистому вигляді, який придатний для нормалізованих векторів і обчислюється швидше за косинусну формулу. Для каталогів товарів найбільш стабільні результати дає косинусна подібність, оскільки вона нечутлива до довжини рядка і враховує лише напрямок вектора у просторі. Задача пошуку найближчих сусідів у точному формулюванні вимагає обчислення відстані від запиту до кожного елемента каталогу, що має лінійну складність O(N) відносно розміру бази. Для каталогів обсягом у сотні тисяч і більше записів точний пошук стає непрактичним за часом, тому застосовуються алгоритми наближеного пошуку найближчих сусідів з логарифмічною або субсублінійною складністю за рахунок контрольованої втрати точності.

Історично перші моделі вкладень слів, серед яких word2vec від Mikolov et al. [39], GloVe від Pennington et al. [40] і fastText від Bojanowski et al. [41], будували окремий вектор для кожного слова на основі статистики сумісної появи у корпусі. Для задач зіставлення назв товарів такі моделі мають обмежену застосовність, оскільки не дають єдиного вектора для всього рядка і потребують агрегації на рівні речення через усереднення або зважену суму. Перехід до архітектур на основі трансформерів типу BERT [42] дозволив будувати контекстуальні вкладення, у яких один і той самий токен отримує різне представлення залежно від оточення. Sentence-BERT від Reimers and Gurevych [24] оптимізує BERT під задачу побудови єдиного вектора речення через донавчання на парах схожих і несхожих прикладів з тріплет-функцією втрат. Мультимовні моделі LaBSE [43] від Google і multilingual-e5 від Microsoft демонструють стабільну якість на понад 100 мовах і придатні для застосувань, де каталог товарів містить

позиції різними європейськими мовами. Для задач у продуктовому середовищі Google Cloud Platform застосовуються нативні моделі text-embedding-004 і text-multilingual-embedding-002 від Vertex AI, які повертають вектори розмірністю 768 і мають вбудовану підтримку пакетного виклику через функцію ML.GENERATE_TEXT_EMBEDDING у BigQuery [44]. Пакетний режим обчислення вкладень для каталогу обсягом у десятки тисяч позицій займає від одиниць секунд до кількох хвилин і коштує менше за еквівалентний обсяг синхронних викликів API.

Один клас алгоритмів наближеного пошуку це деревоподібні структури KD-tree і Ball-tree, придатні для просторів низької розмірності. У просторах розмірністю сотень вимірів ефективність дерев різко падає через прокляття розмірності, що робить їх непрактичними для сучасних моделей вкладень. Інший клас це хешування з локальною чутливістю LSH, у якому функції хешування будуються так, що близькі вектори з високою ймовірністю потрапляють у один бакет. Перевагою LSH є проста паралелізація, недоліком низька recall на запитах з високою точністю, що робить метод придатним для дуже великих баз з відносно грубими вимогами до якості.

Ще один клас це граф навігаційних малих світів, серед яких Hierarchical Navigable Small World (HNSW) від Malkov and Yashunin [25] є галузевим стандартом. Алгоритм будує ієрархічний граф, у якому верхні рівні містять довгі перескоки через простір, а нижні рівні – точні локальні зв’язки. Пошук виконується жадібним спуском з верхнього рівня до нижнього з очікуваною складністю, близькою до логарифмічної O(log N), і recall@10 вище 95% на каталогах у сотні тисяч позицій. Схему такого пошуку з верхнього шару до базового наведено на рисунку 2.1. Окремий клас це квантування добутків (Product Quantization) і інвертовані файли, описані у роботі Jegou et al. [45] і реалізовані у бібліотеці FAISS від Meta Research [27]. Метод розбиває вектор на блоки, навчає окремий квантизатор для кожного блоку і зберігає вектор у компактному цілочисельному коді,

що зменшує обсяг пам’яті у десятки разів і дає субсеконду пошук на мільярдних базах.

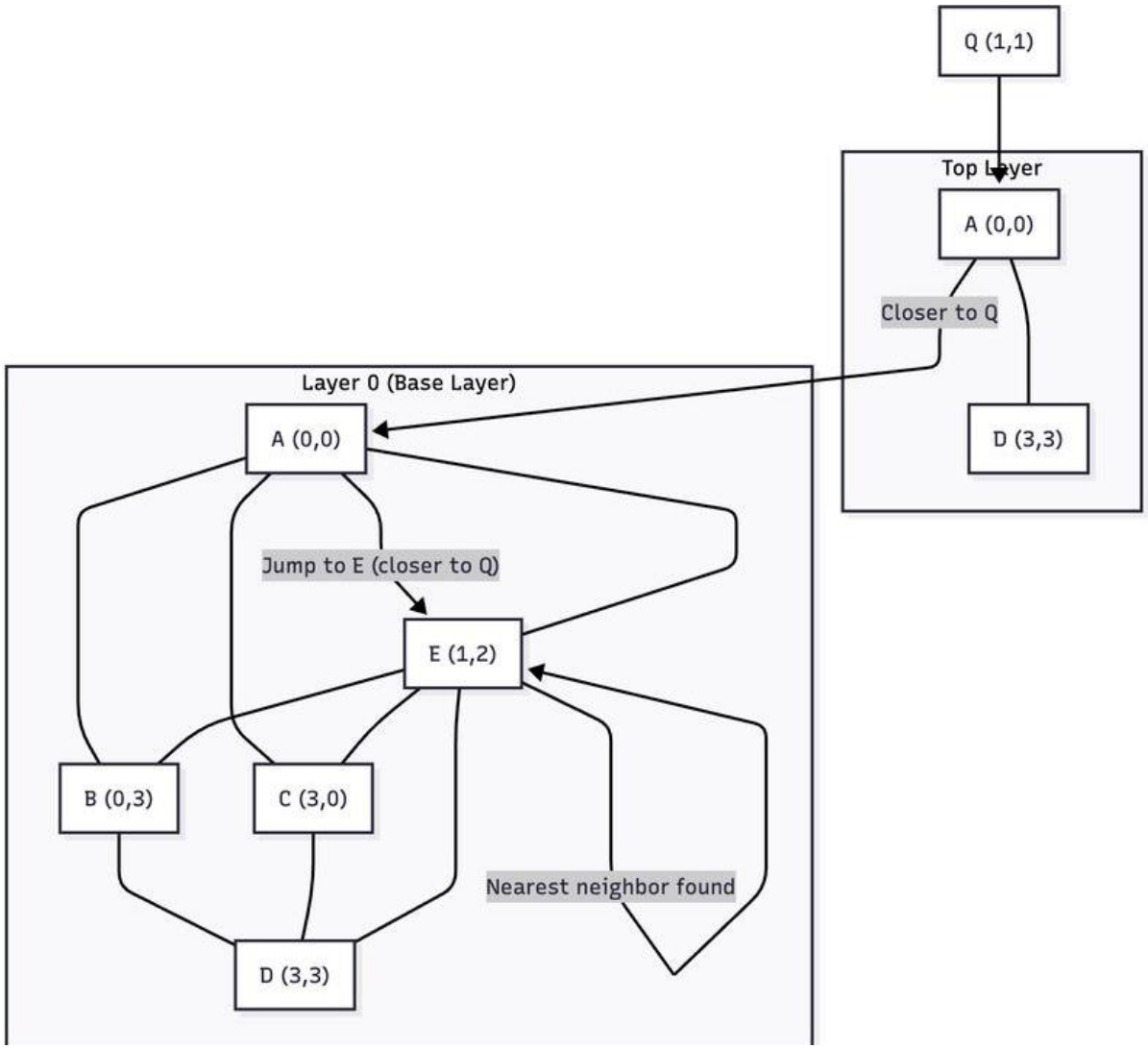


Рисунок 2.1 – Схема пошуку найближчого сусіда в ієрархічному графі HNSW [46]

Керовані сервіси векторного пошуку звільняють інженерну команду від експлуатації власного індексу і забезпечують автоматичне масштабування. Vertex AI Vector Search від Google базується на алгоритмі ScaNN [47] і забезпечує затримку близько 10 мс на каталогах до 1 млн записів з можливістю гарячого оновлення індексу. Повну схему конвеєра

побудови векторного індексу у Google Cloud Platform, що охоплює навчання two-tower моделі у Vertex AI, реєстрацію башт у Model Registry, пакетне обчислення вкладень через Vertex AI Prediction, збереження результатів у Cloud Storage та розгортання ANN-індексу у Vertex AI Vector Search, наведено на рисунку 2.2. Pinecone, Weaviate і Qdrant пропонують аналогічні можливості з різницею у ціновій моделі і наборі підтримуваних мов метаданих. Окремо стоїть функція VECTOR_SEARCH у BigQuery, що дає змогу виконувати пошук найближчих сусідів безпосередньо у SQL-запиті з приєднанням до аналітичних таблиць. Функція підтримує як точний повний перебір, так і створення індексу на основі IVF, і оптимізована для пакетних сценаріїв, де потрібно одночасно обробити тисячі запитів [48]. Вартість пакетного пошуку у BigQuery істотно нижча за вартість еквівалентної кількості запитів до виділеного векторного сервісу, що робить цю функцію економічно вигідним варіантом для пакетного режиму.

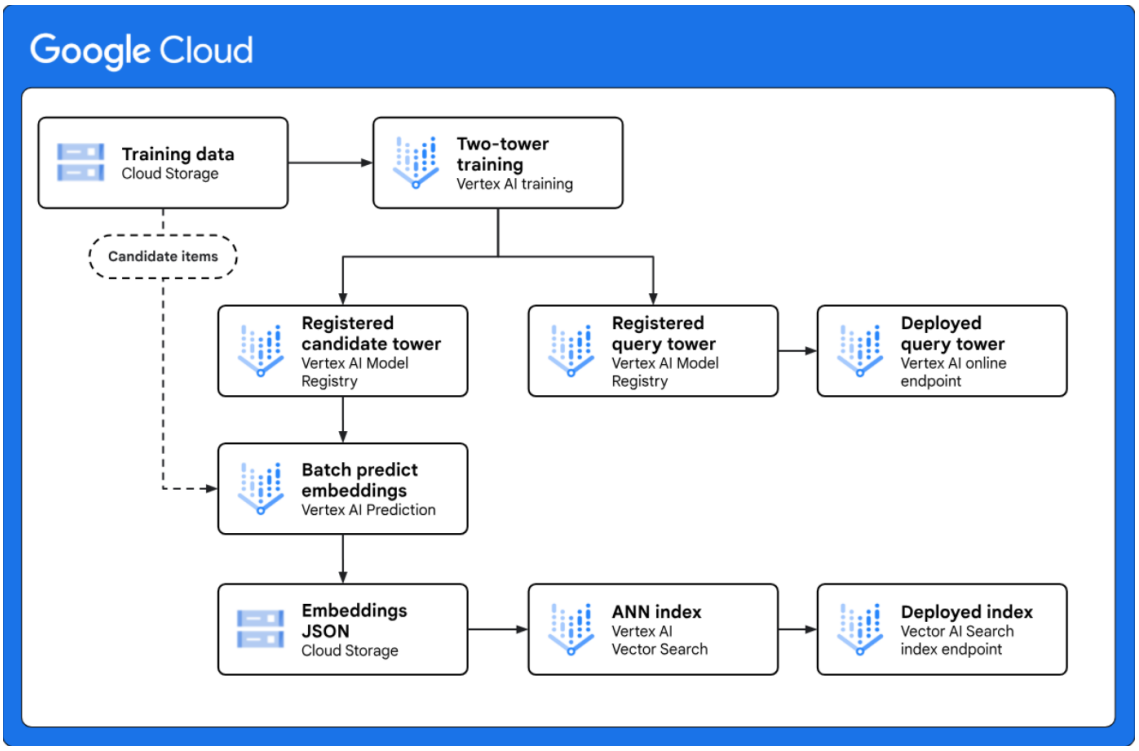


Рисунок 2.2 – Конвеєр побудови векторного індексу у Vertex AI Vector Search [49]

Для кваліфікаційної роботи прийнято гібридну стратегію пошуку. Для запитів з одним чеком у режимі реального часу застосовується Vertex AI Vector Search з заздалегідь розгорнутим ендпоінтом, який забезпечує низьку і стабільну затримку у межах десятків мілісекунд. Для пакетних запитів з 2 і більше чеками застосовується функція VECTOR_SEARCH у BigQuery над аналітичною таблицею векторних представлень каталогу, що дає нижчу сукупну вартість обробки і нативну інтеграцію з іншими аналітичними запитами системи. Логіка перемикання автоматизована у коді сервісу зіставлення товарів на основі розміру вхідного пакета запитів. Векторні представлення назв товарів обчислюються багатомовною моделлю сімейства Vertex AI Text Embeddings розмірністю 768. Каталог промо-товарів оновлюється пакетним завданням BigQuery, що викликає функцію обчислення вкладень по оновлених рядках і перезаписує аналітичну таблицю векторних представлень у режимі повного оновлення. Таке поєднання мінімізує вартість зберігання індексу, забезпечує гнучкість вибору методу пошуку залежно від сценарію і дає повну контрольованість співвідношення швидкості, точності і сукупних витрат.

2.5 Методи виявлення шахрайства при обробці касових чеків

Відкрита модель гейміфікованої винагороди за завантажений чек створює економічний стимул для зловживань, описаних у підрозділі 1.2. Без системи протидії такі зловживання швидко знецінюють ігрову валюту, руйнують довіру добросовісних учасників і роблять економічну модель платформи нежиттєздатною. Специфіка задачі полягає у поєднанні бізнес-правил з візуальним аналізом, оскільки частина атак зводиться до цифрової модифікації зображення чека у графічних редакторах. У підрозділі виконано аналіз п’яти класів методів виявлення шахрайства і сформульовано архітектуру, прийняту у роботі.

Базовими метриками якості бінарного класифікатора підробок у роботі прийнято точність (Accuracy), F1-міру і площу під ROC-кривою (AUC-ROC), що визначаються формулами (2.4)–(2.6).

$$\mathrm{Acc} = (\mathrm{TP} + \mathrm{TN}) / (\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}), \tag{2.4}$$

$$\mathrm{F1} = 2 \cdot \mathrm{P} \cdot \mathrm{R} / (\mathrm{P} + \mathrm{R}), \text{де } \mathrm{P} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FP}), \mathrm{R} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}), \tag{2.5}$$

$$\mathrm{FNR} = \mathrm{FN} / (\mathrm{TP} + \mathrm{FN}), \mathrm{FPR} = \mathrm{FP} / (\mathrm{FP} + \mathrm{TN}), \tag{2.6}$$

де TP, TN, FP, FN – істинно-позитивні, істинно-негативні, хибно-позитивні і хибно-негативні класифікації;

P – precision;

R – recall;

FNR – частка пропущених реальних підробок;

FPR – частка хибних спрацьовувань на справжніх чеках.

Один клас методів спрямований на розпізнавання повторно завантажених фотографій чеків. Базовим інструментом є перцептивні хеші, серед яких pHash, dHash, aHash і wHash [50]. Алгоритм pHash обчислює дискретне косинусне перетворення зменшеного до 8×8 пікселів зображення у градаціях сірого і зберігає знак коефіцієнтів у бінарний вектор довжиною 64 біти. Два зображення вважаються дублікатами, якщо відстань Геммінга між їхніми pHash-кодами не перевищує порогового значення, типово 5 або 6. dHash використовує різницю яскравостей сусідніх пікселів, wHash будується на коефіцієнтах вейвлет-перетворення і дає кращу стійкість до шуму. Обмеження перцептивних хешів проявляється у вразливості до агресивних перетворень, серед яких значне обертання, обрізання країв чека, зміна балансу білого. Для стійкого виявлення застосовуються ознакові детектори SIFT, SURF і ORB. Ці алгоритми

знаходять локальні особливості зображення, стійкі до масштабування і обертання, і порівнюють два зображення за кількістю зіставлених особливостей. Точність виявлення дублікатів на реальних чеках знаходиться у межах 96–99% при порозі збігу від 15% ключових точок.

Інший клас методів не спирається на зображення і використовує структуровані поля чека. Кожен касовий чек у країнах ЄС містить номер фіскального реєстратора, дату і час покупки, порядковий номер операції, суму до сплати. Комбінація цих полів є ймовірнісно унікальною і використовується як природний ключ для виявлення повторного подання чека. У системі зберігається таблиця вже опрацьованих фіскальних ідентифікаторів і кожен новий чек перевіряється на наявність у цій таблиці перед нарахуванням винагороди. Перевага цього підходу полягає у стійкості до перетворень зображення, оскільки рішення приймається на основі розпізнаного тексту. Обмеження проявляється у випадках, коли шахрай штучно змінює один зі знаків номера або дати, щоб обійти точну перевірку. Для протидії застосовується комбінація точного збігу за фіскальним номером з нечітким зіставленням за сумою і датою у вікні кількох хвилин.

Ще один клас методів націлений на виявлення цифрових модифікацій чека у графічному редакторі, серед яких заміна цифр суми, підміна дати, накладення тексту з іншим згладжуванням, сліди інструменту клонування, неоднорідні артефакти JPEG-стиснення. Класичні алгоритми на основі аналізу помилок рівня стиснення ELA і частотного спектру виявляють прості випадки підробки, однак пасують на сучасних атаках зі збереженням у високій якості або повторним знімком відредагованого чека з екрана.

Першим поколінням нейромережевих рішень стали згорткові класифікатори з попереднім навчанням на ImageNet і донавчанням на корпусі пар справжній та підроблений чек. Архітектура ResNet50 від He et al. [51] за рахунок залишкових зв’язків забезпечує стабільне глибоке витягування ознак і на задачі виявлення підробок чеків дає точність валідації 92,35%, AUC 0,9617, False Negative Rate 3,57% і False Positive

Rate 11,73% на тестовій вибірці з 392 зображень. Загальну структуру мережі наведено на рисунку 2.3. Ці показники обрано у роботі як базовий орієнтир для порівняння з новішими архітектурами.

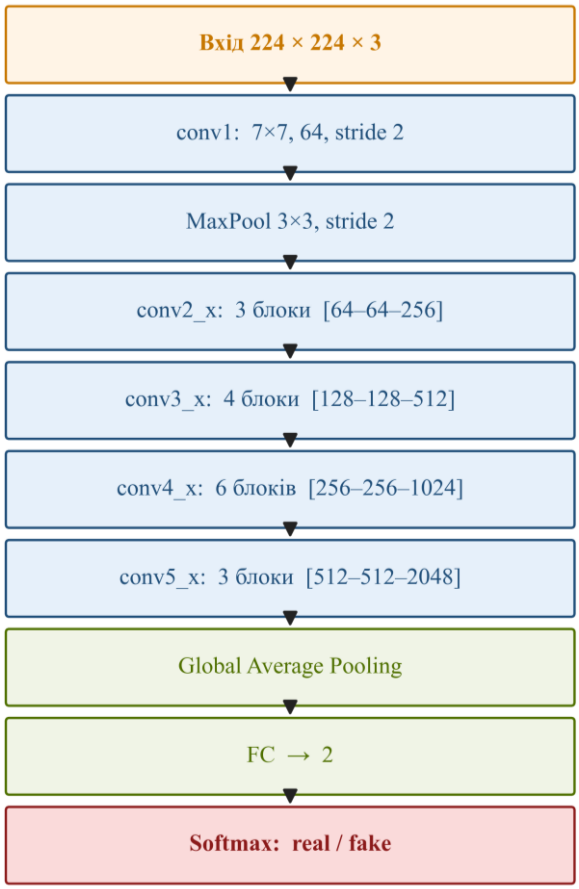


Рисунок 2.3 – Загальна структура мережі ResNet50

Наступним поколінням є сімейство EfficientNet від Tan and Le [52], у якому застосовано принцип складеного масштабування, що пропорційно збільшує глибину мережі, ширину каналів і роздільну здатність входу. EfficientNetV2-S є компромісним представником сімейства з 21,7 млн параметрів і спеціалізованим Fused-MBConv блоком у ранніх шарах, що прискорює навчання без втрати точності. На тій самій задачі виявлення підробок чеків EfficientNetV2-S перевершує ResNet50 за всіма ключовими метриками і дає точність 94,13%, AUC 0,9694, FNR 2,04%, FPR 9,69% при

значно меншій кількості параметрів порівняно з ResNet50. Загальну послідовність стадій мережі зі зміною типу блоку з Fused-MBConv у ранніх стадіях на MBConv з Squeeze-and-Excitation у пізніх наведено на рисунку 2.4.

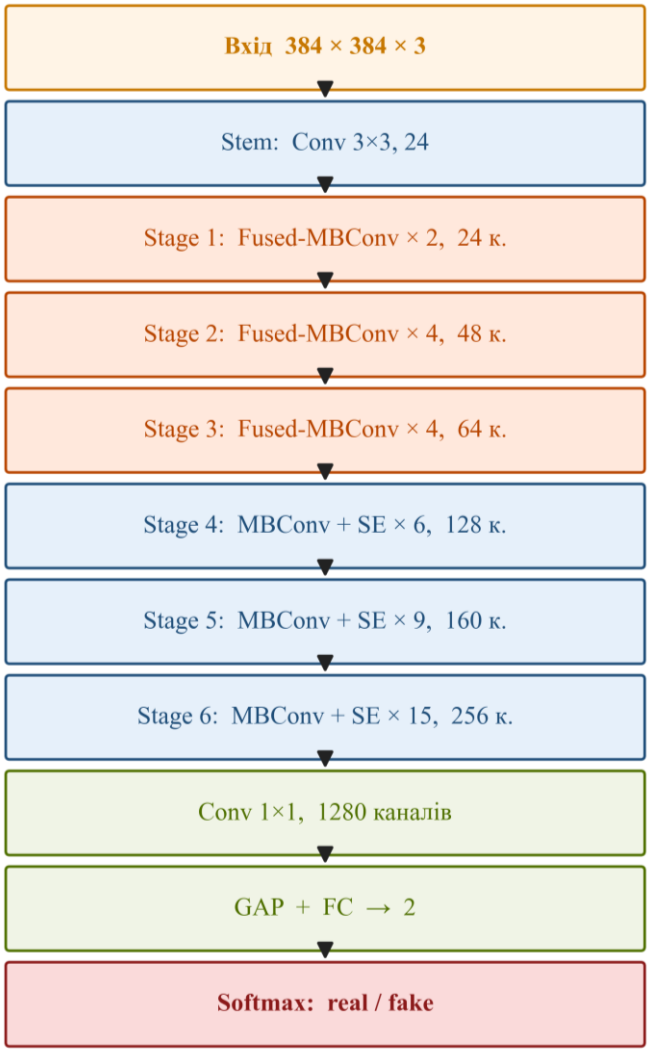


Рисунок 2.4 – Загальна структура мережі EfficientNetV2-S

Сучасним поколінням, застосованим у запропонованій системі, є багатопотокова архітектура, що об’єднує 3 доповнювальні представлення одного зображення чека. Загальну структуру мережі з 3 потоками, механізмами Attention Fusion і Edge Attention, класифікаційною і

допоміжною сегментаційною головами наведено на рисунку 2.5. Один потік RGB приймає кольорове зображення у роздільній здатності 384×384 пікселі і подає його на попередньо навчений EfficientNetV2-S, який повертає вектор ознак з 1280 каналів. Інший потік будується на моделі просторово багатої статистики SRM (Spatial Rich Model), що застосовує 30 фіксованих високочастотних фільтрів для виявлення залишкових шумових патернів, характерних для інструментів редагування. Після SRM застосовується конволюція BayarConv з накладеним обмеженням на центральний коефіцієнт ядра, що дозволяє мережі автоматично знаходити характерні для маніпуляцій шумові підписи [53]. Отриманий шумовий канал подається на EfficientNet-B0, який повертає ще один вектор з 1280 ознак. Ще один потік обчислює дискретне косинусне перетворення на блоках розміром 8×8 пікселів, що виявляє сліди подвійного JPEG-стиснення, характерні для повторно збережених у редакторі зображень. Для цього потоку застосовується легкий згортковий енкодер з виходом на 256 каналів.

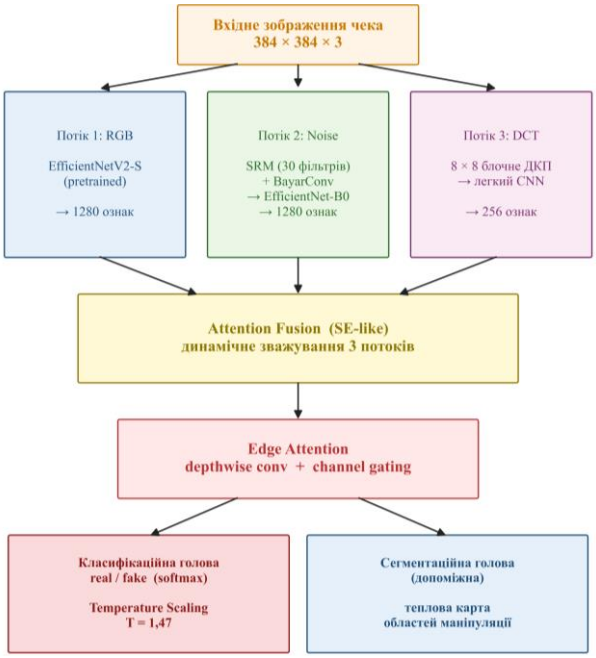


Рисунок 2.5 – Архітектура Multi-Stream EfficientNet для виявлення підробок чеків

Три потоки об’єднуються через механізм Attention Fusion, побудований за принципом Squeeze-and-Excitation, що динамічно зважує внесок кожного потоку залежно від характеру конкретного вхідного зображення. На виході блоку Fusion застосовується Edge Attention через комбінацію depthwise-конволюції і channel gating, що підсилює ознаки локальних маніпуляцій на краях вставлених фрагментів. Класифікаційна голова приймає остаточний вектор і повертає ймовірність класу підробки. Паралельно працює сегментаційна голова як допоміжне завдання, що будує теплову карту ймовірних областей редагування і підвищує якість основного класифікатора за рахунок багатозадачного навчання. Для калібрування впевненості класифікатора після навчання застосовано Temperature Scaling з температурою T = 1,47. Модель навчено на датасеті з 1949 справжніх і 1949 підроблених чеків з тестовою вибіркою 392 зображення.

Порівняння 3 базових варіантів на ідентичному тесті наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння архітектур виявлення підробок чеків на тестовій вибірці з 392 зображень

Модель	Acc	F1	MCC	AUC-ROC	FNR	FPR	Параметри
ResNet50 (baseline)	92,35%	0,9233	0,8498	0,9617	3,57%	11,73%	26,7M
EfficientNetV2-S	94,13%	0,9412	0,8852	0,9694	2,04%	9,69%	21,7M
Multi-Stream (EfficientNet)	94,64%	0,9464	0,8955	0,9805	1,53%	9,18%	28,7M

Як видно з таблиці 2.1, перехід від одиночного ResNet50 до EfficientNetV2-S підвищує точність на 1,78 процентного пункту при одночасному зменшенні кількості параметрів на 20%. Додавання потоків SRM і DCT у багатопотоковій архітектурі додає ще 0,51 процентного пункту точності і суттєво знижує False Negative Rate з 2,04% до 1,53%, що прямо впливає на здатність системи не пропускати реальні випадки

шахрайства. AUC-ROC зростає з 0,9617 до 0,9805, що підтверджує покращену здатність моделі розділяти класи справжнього і підробленого чека у широкому діапазоні порогів.

Окремий клас методів використовує мультимодальні великі мовні моделі, серед яких Gemini 2.5 Flash, для семантичного аналізу підозрілих зображень. На вхід моделі подається фотографія чека і текстовий промпт, що перелічує типові ознаки маніпуляції, серед яких кольоровий текст на чорно-білому чеку, різне згладжування символів у сусідніх рядках, гало і тіні навколо вставлених фрагментів, сліди інструменту клонування, непослідовні шрифти, накладення паперових смужок, рукописні правки поверх друку. Модель повертає структуровану відповідь з ймовірністю підробки, переліком виявлених артефактів і текстовим обґрунтуванням рішення. Перевагою такого підходу є висока якість виявлення складних випадків без потреби у великому навчальному корпусі. Недоліком є час відгуку на рівні кількох секунд і істотно вища вартість виклику порівняно з локальним висновком згорткового класифікатора. У продуктових сценаріях мультимодальні LLM застосовуються як резервний механізм для зразків з невизначеним результатом швидкої локальної моделі, що дає комбінацію низької середньої затримки і високої підсумкової точності.

Останній клас методів працює на рівні часових рядів активності користувача і застосовує алгоритми машинного навчання без учителя. Ізоляційний ліс Liu et al. [54] будує ансамбль випадкових дерев розділення і обчислює для кожного спостереження середню глибину листа як міру аномальності. One-class SVM від Schölkopf et al. [55] навчає межу, що відокремлює нормальні спостереження від простору аномалій. Автоенкодери моделюють розподіл нормальної активності у низьковимірному латентному просторі, а аномалії виявляються за високою похибкою реконструкції. Типовими ознаками поведінкового вектора є частота сканувань за одиницю часу, розподіл сум, географічна розкиданість, співвідношення чеків різних мереж, інтервал між діями. Для виявлення

скоординованих атак застосовуються графові методи пошуку щільних підграфів за спільним пристроєм, IP-адресою, спільними чеками, що деталізовано розглянуті у роботі Jiang et al. [56] щодо алгоритмів типу fBox і CatchSync.

Для кваліфікаційної роботи прийнято багаторівневу стратегію протидії шахрайству, реалізовану у спеціалізованому сервісі виявлення шахрайства. Перший рівень це перевірка унікальності комбінації фіскального номера, дати, часу і суми чека через таблицю опрацьованих ідентифікаторів з індексом за унікальним ключем чека і додаткове порівняння pHash зображення для виявлення повторних подань. Другий рівень це візуальна форензика на основі багатопотокової архітектури Multi-Stream EfficientNet, що поєднує потоки RGB, SRM+BayarConv і DCT з механізмами Attention Fusion та Edge Attention. Модель навчено протягом 20 епох з оптимізатором AdamW, швидкістю навчання 1×10^{-4}, аугментаціями HorizontalFlip, Rotate у межах ±15°, RandomBrightnessContrast, GaussNoise на зображеннях розміром 384×384 пікселі з навчанням у режимі FP16 AMP на GPU Tesla T4. Після навчання застосовано Temperature Scaling для калібрування ймовірностей. Середня затримка повного прямого проходу моделі становить 32,91 мс на одне зображення у режимі пакетної обробки з розміром пакета 10. Третій рівень це резервний виклик мультимодальної моделі Gemini 2.5 Flash для зразків з невизначеним результатом Multi-Stream класифікатора. Фінальне рішення приймається метакласифікатором XGBoost зі зваженою комбінацією базового класифікатора і Gemini з порогом, підібраним на валідаційній вибірці. Четвертий рівень реалізовано у двох частинах: детермінований набір правил бізнес-логіки на основі довідника торговельних точок, що перевіряють відповідність чека торговельній мережі з активної промо-кампанії, мінімальну суму і допустиме часове вікно між покупкою і поданням, та періодичне завдання виявлення аномалій, що будує для кожного користувача вектор поведінкових ознак і застосовує ізоляційний ліс для ідентифікації

підозрілих профілів з подальшим направленням на ручну перевірку модератора. Така чотирирівнева архітектура закриває основні вектори зловживань, описані у підрозділі 1.2, і зберігає баланс між жорсткістю контролю і якістю користувацького досвіду.

2.6 Хмарні платформи обробки даних

Вибір хмарної платформи для розгортання системи гейміфікованої лояльності визначає сукупну вартість експлуатації, швидкість виходу нових функцій і доступність готових сервісів машинного навчання, які безпосередньо використовуються у конвеєрі обробки чеків. У підрозділі виконано порівняльний аналіз 3 провідних публічних хмар у контексті задач роботи і сформульовано обґрунтування вибору Google Cloud Platform.

Оцінювання платформ здійснено за 6 критеріями, які випливають з архітектурних вимог підрозділу 1.2. Один критерій це наявність керованого OCR-сервісу з підтримкою європейських мов і прийнятною тарифною моделлю. Інший критерій це доступність мультимодальних великих мовних моделей з можливістю пакетного передбачення зі знижкою. Ще один критерій це керований векторний пошук, інтегрований з аналітичним сховищем. Окремий критерій це безсерверні обчислення з оплатою за фактичний час виконання. Наступний критерій це наявність документної NoSQL бази даних реального часу для мобільних клієнтів. Останній критерій це вартість зберігання і обчислень у регіонах Центральної і Східної Європи, де переважно розміщуються цільові користувачі системи.

Google Cloud Platform надає повний набір сервісів, необхідних для побудови описаного конвеєра, у єдиному середовищі з нативною інтеграцією. OCR забезпечується сервісом Cloud Vision API у режимі DOCUMENT_TEXT_DETECTION з підтримкою понад 50 мов і прайсом на рівні 1,5 дол. США за 1000 викликів [57]. Мультимодальні великі мовні моделі доступні через Vertex AI як сервіс, що охоплює Gemini 2.5 Flash у

синхронному і пакетному режимах. Vertex AI Batch Prediction дає знижку близько 50% на пакетні виклики порівняно зі синхронними і нативно інтегрується з Cloud Storage у ролі вхідного і вихідного сховища [58]. Векторний пошук забезпечується двома сервісами з взаємодоповнюючими профілями. Vertex AI Vector Search базується на алгоритмі ScaNN і дає затримку близько 10 мс для реальночасних запитів, BigQuery VECTOR_SEARCH дозволяє виконувати пакетний пошук безпосередньо у SQL-запиті з приєднанням до аналітичних таблиць без додаткового копіювання даних. Безсерверні обчислення представлені сервісами Cloud Functions Gen2 і Cloud Run, які працюють на спільному стеку Knative, підтримують автоматичне масштабування від 0 до тисяч екземплярів і тарифікуються за часом виконання у мілісекундах пам'яті [59]. Документна NoSQL база Firestore у режимі Native Mode забезпечує мобільні клієнти прямим підключенням з оновленнями реального часу через WebSocket. BigQuery виконує роль аналітичного сховища і підтримує пряму взаємодію з Vertex AI через функцію ML.GENERATE_TEXT_EMBEDDING для обчислення вкладень. Регіон Google Cloud Platform у Центральній Європі забезпечує низьку мережеву затримку для користувачів з Польщі, України, Чехії, Словаччини з помірною націнкою порівняно з американськими регіонами.

Amazon Web Services пропонує аналогічний за широтою набір сервісів, однак з різнорідною інтеграцією між компонентами. OCR забезпечується сервісом AWS Textract з API Analyze Expense, спеціалізованим під чеки і рахунки. Тарифна модель будується на оплаті за сторінку з середньою ціною 0,01 дол. США за сторінку, що зіставне з Cloud Vision API, однак точність розпізнавання на європейських мовах поза англійською поступається Google. Мультимодальні великі мовні моделі доступні через Amazon Bedrock і охоплюють Claude Opus 4.7 від Anthropic, Llama від Meta, Mistral. Нативного пакетного передбачення з дисконтом, еквівалентного Vertex AI Batch Prediction, у Bedrock немає, що знижує

економічну ефективність у сценаріях масової обробки накопичених чеків. Векторний пошук пропонується у 2 варіантах, серед яких Amazon OpenSearch Service з модулем k-NN і Amazon Aurora PostgreSQL з розширенням pgvector. Обидва варіанти передбачають постійну абонентську плату за виділений екземпляр, що створює фіксовану статтю витрат незалежно від фактичного навантаження. Безсерверні обчислення представлені сервісом AWS Lambda з лімітом часу виконання 15 хв і обмеженим обсягом пам'яті до 10 ГБ. Документна база DynamoDB підтримує мобільні інтеграції, однак не має еквіваленту Firestore щодо безкоштовного тарифу реальночасних підписок для клієнтських застосунків. Найближчий регіон AWS у Франкфурті має нижчу мережеву затримку для Західної Європи і вищу для Центральної і Східної порівняно з центральноєвропейським регіоном Google.

Microsoft Azure пропонує третю конкурентну пропозицію з фокусом на корпоративних клієнтах і інтеграцію з продуктами Microsoft 365. OCR забезпечується сервісом Azure AI Document Intelligence з моделлю Prebuilt Receipt, що підтримує базові європейські мови з нижчою точністю на кирилиці порівняно з Cloud Vision API. Мультимодальні великі мовні моделі доступні через Azure OpenAI Service і охоплюють моделі сімейства GPT-4. Нативного інтегрованого пакетного режиму з глибоким дисконтом для мультимодальних викликів на момент аналізу у Azure OpenAI Service немає, що аналогічно обмежує економічну привабливість для задач масової обробки [60]. Векторний пошук реалізується у сервісах Azure AI Search з модулем Vector Search і Azure Cosmos DB for MongoDB vCore з підтримкою векторних індексів. Обидва варіанти передбачають абонентську плату за виділений екземпляр, аналогічно моделі AWS. Безсерверні обчислення надаються сервісом Azure Functions з аналогічними до Lambda обмеженнями. Документна база Cosmos DB є потужним рішенням для мультирегіональних інсталяцій, однак має вищу вартість реальночасних підписок порівняно з Firestore.

Результати порівняння за 6 критеріями дають однозначну перевагу Google Cloud Platform. Точність Cloud Vision API на європейських мовах з діакритикою перевищує аналогічний показник AWS Textract і Azure AI Document Intelligence, що безпосередньо впливає на якість роботи подальших етапів конвеєра. Vertex AI Batch Prediction з дисконтом близько 50% на Gemini Flash не має прямого аналога у Bedrock і Azure OpenAI Service, що дає істотну економію на пакетному режимі обробки. BigQuery VECTOR_SEARCH усуває потребу у виділеному векторному сервісі з абонентською платою, що недосяжно у моделях AWS і Azure. Firestore у режимі Native Mode забезпечує нативні реальночасні підписки для мобільних клієнтів без додаткових посередників. Центральноєвропейський регіон Google Cloud Platform дає мережеву затримку у межах 30 мс для цільової аудиторії з країн Центральної і Східної Європи. Додатковим аргументом є зрілість інструментів оркестрації і аналітики, серед яких Cloud Workflows для побудови пакетних конвеєрів, Cloud Scheduler для регулярних завдань, Looker Studio для побудови дашбордів без коду, Cloud Logging з нативною інтеграцією у всі компоненти системи. Сукупність зазначених факторів формує єдине технологічне середовище, у якому описані у Розділі 3 програмні компоненти системи поєднуються без додаткового шару інтеграційного коду, що прискорює розробку і знижує операційні витрати. Такий комплексний підхід гарантує високу масштабованість інфраструктури в умовах постійно зростаючого навантаження. Використання переважно керованих хмарних сервісів дозволяє зосередити основну увагу безпосередньо на реалізації специфічної бізнес-логіки та вдосконаленні алгоритмів замість рутинної підтримки серверної частини. У кінцевому підсумку це створює максимально надійне підґрунтя для успішного впровадження розроблюваної гейміфікованої системи лояльності та досягнення всіх поставлених цілей дослідження.

3 ПРОЄКТУВАННЯ І РОЗРОБКА СИСТЕМИ

3.1 Проєктування системи

Архітектура спроєктована з опорою на теоретичні висновки Розділу 2 і реалізована на платформі Google Cloud Platform у регіоні Центральної Європи. Базовий технологічний стек охоплює моделі сімейства Gemini 2.5 Flash і Gemini 2.5 Flash-Lite, керований векторний пошук Vertex AI Vector Search, аналітичне сховище BigQuery і документну базу Firestore.

Систему організовано у чотири логічні шари, кожен з яких відповідає за відокремлений набір функцій і взаємодіє з сусідніми через визначені інтерфейси. Шар приймання даних об’єднує мобільні і веб-клієнти, сховище зображень Cloud Storage і дві бази Firestore, продуктову з метаданими промо-кампаній і реальночасну з результатами обробки чеків. Шар обробки даних складається з шістнадцяти програмних компонентів на Cloud Functions Gen2 і Cloud Run, поділених на п’ять функціональних груп: обробка чеків, ETL з Firestore у BigQuery, оркестрація, аналітика і моніторинг якості. Шар аналітичного сховища і векторного індексу містить датасет BigQuery і ScaNN-індекс Vertex AI Vector Search розмірністю 768, розгорнутий на виділеному ендпоінті. Шар візуалізації охоплює дашборди Looker Studio для маркетингових команд виробників і HTML-дашборд моніторингу даних для інженерної команди. Чотиришарову схему наведено на рисунку 3.1. Такий чіткий архітектурний поділ на ізольовані рівні дозволяє здійснювати незалежне масштабування кожного окремого компонента залежно від обсягу специфічного навантаження. Крім того, це суттєво спрощує процес безпечної інтеграції нових функціональних можливостей у майбутньому без ризику порушення цілісності існуючих базових процесів. У кінцевому підсумку подібний строгий модульний підхід гарантує високу загальну надійність та експлуатаційну відмовостійкість усього програмного комплексу.

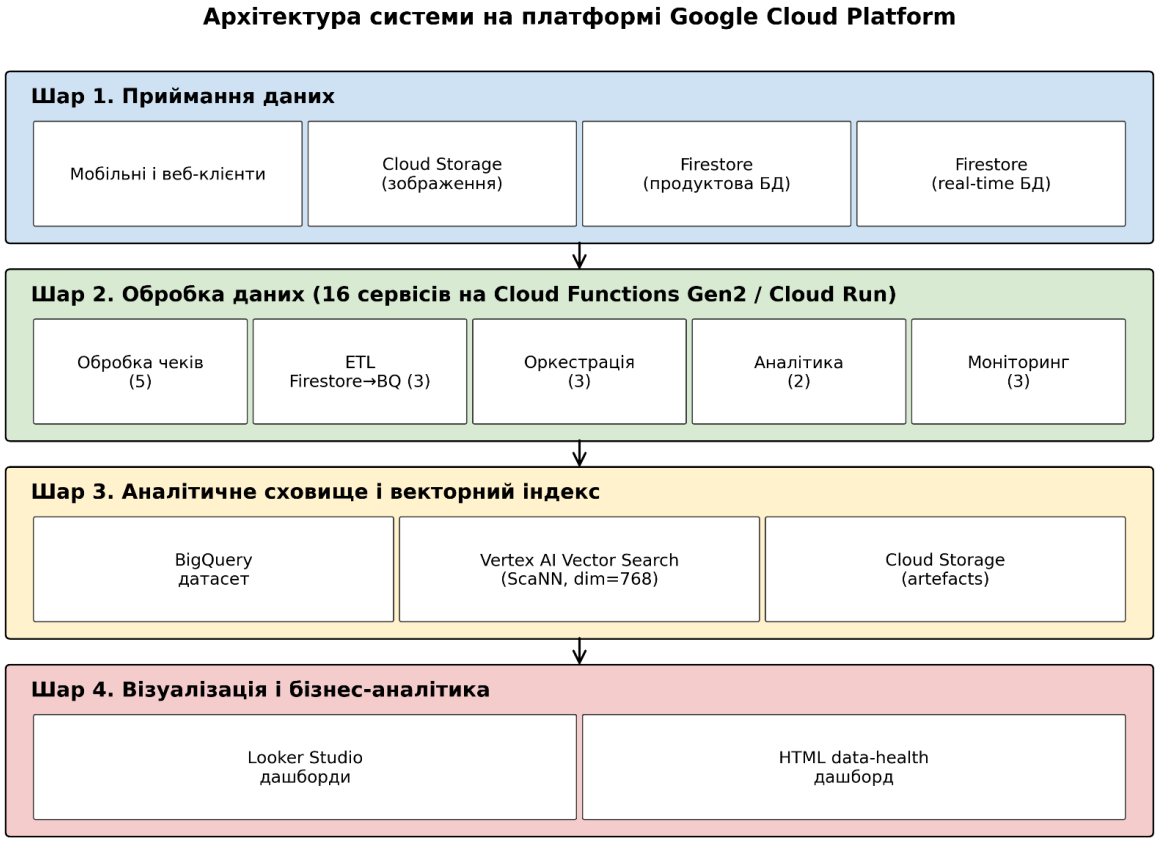


Рисунок 3.1 – Чотиришарова архітектура системи на платформі Google Cloud Platform

Систему проєктовано під два режими обробки чеків з відмінними вимогами щодо швидкодії і економіки. Пакетний режим обслуговує масову обробку накопичених чеків через Vertex AI Batch Prediction з моделями Gemini 2.5 Flash і дає економію близько 50% на вартості токенів порівняно з синхронним викликом API; час обробки пакета з кількох тисяч чеків становить від 5 хвилин до 1 години залежно від обсягу. Режим реального часу обслуговує миттєвий зворотний зв’язок з покупцем у мобільному застосунку і реалізований у трьох варіантах. Гібридний варіант послідовно викликає Cloud Vision API і Gemini 2.5 Flash-Lite за час від 2 до 3 секунд. Мультимодальний передає зображення напряму у Gemini Vision і відпрацьовує за 1,5–2,5 секунди. Текстовий приймає готовий OCR-текст і виконує лише структурування приблизно за секунду.

Ключове архітектурне рішення цієї роботи це автоматичний вибір методу векторного пошуку товарів у каталозі залежно від розміру вхідного пакета. Логіка вибору активується у точці входу сервісу зіставлення і не вимагає ручного конфігурування. Для запитів від 1 до 10 чеків застосовується Vertex AI Vector Search з розгорнутим ScaNN-ендпоінтом, що дає затримку близько 10 мс на запит. Для запитів від 11 чеків і більше застосовується функція `VECTOR_SEARCH` у BigQuery над таблицею векторних представлень каталогу, що знижує сукупну вартість пакета приблизно на 70% порівняно з серією окремих викликів Vertex AI Vector Search.

Маршрут даних сценарію реального часу починається з завантаження зображення у Cloud Storage. Клієнт викликає HTTP-ендпоінт сервісу обробки у режимі реального часу, який виконує OCR і структурування, записує результат у колекції реального часу Firestore, далі викликає сервіс зіставлення з каталогом і сервіс виявлення шахрайства. Клієнт отримує структуровану відповідь з виявленими позиціями, зіставленими промо-товарами і нарахованими винагородами.

Маршрут пакетного сценарію починається з виклику погодинного оркестратора за розкладом Cloud Scheduler. ETL-компонент довантажує нові чеки з Firestore у плоску транзакційну таблицю BigQuery, далі головний сервіс обробки виконує трансформацію в аналітичні таблиці. Денний оркестратор коригує назви товарів через Vertex AI Batch Prediction, оновлює векторні представлення каталогу через `ML.GENERATE_TEXT_EMBEDDING`, виконує зіставлення через `VECTOR_SEARCH` і викликає сервіс побудови агрегатів.

Усі компоненти спроєктовано за принципом ідемпотентності операцій запису. Таблиця денних агрегатів оновлюється у режимі `DELETE` плюс `INSERT` за ідентифікатором звіту, візуалізаційні таблиці записуються у режимі повного перезапису, ETL-операції застосовують дедублікацію за ідентифікатором документа. Відмовостійкість

забезпечується безсерверною моделлю виконання, яка автоматично перезапускає екземпляри у разі збою. Конфігурація сервісів винесена у змінні оточення з резервними значеннями, що дозволяє запускати ті самі функції у тестовому і продуктовому середовищі без зміни коду.

3.2 Структура програмного коду і модель розгортання

Кодова база системи становить близько 12000 рядків Python і складається з 16 незалежних сервісів. Вихідний код організовано як монорепозиторій, у кореневій директорії якого розміщено 16 піддиректорій по одній на сервіс. Спільний репозиторій полегшує крос-сервісний рефакторинг, оскільки зміна формату аналітичної таблиці або колекції Firestore вимагає синхронного оновлення кількох сервісів. Фізичне розділення на директорії з незалежним конфігом дозволяє розгортати кожен сервіс окремо і точково відкочувати зміни у разі помилки.

Декомпозиція виконана за функціональним принципом і повторює п’ять груп сервісів, описаних у підрозділі 3.1. Головний сервіс пакетної обробки чеків винесено в окрему директорію через великий обсяг коду і наявність 6 режимів роботи. Сервіс обробки у режимі реального часу і полегшений сервіс обробки готового OCR-тексту винесено у дві окремі директорії для мінімізації холодного старту.

Кожна директорія сервісу містить три обов’язкові файли. Файл main.py містить точку входу і прикладну логіку сервісу. Файл requirements.txt фіксує мінімальний набір залежностей саме для цього сервісу. Скрипт deploy.sh виконує розгортання через gcloud functions deploy або gcloud run deploy з прапорцями пам’яті, таймауту, змінних оточення і сервісного облікового запису.

У частині директорій додатково розміщено `README.md` з описом публічного контракту сервісу і прикладами викликів.

Точка входу кожного сервісу оформлена через декоратор `@functions_framework.http` з бібліотеки `functions-framework`. Декоратор інкапсулює перетворення HTTP-запиту у Flask Request і зворотне формування відповіді, що уніфікує контракт незалежно від того, чи сервіс розгорнуто як Cloud Functions Gen2, чи як Cloud Run. Локальний запуск виконується командою `functions-framework --target=main --port=8080`, що зводить цикл налагодження до однієї команди.

Усі клієнти керованих сервісів GCP ініціалізуються на рівні модуля поза межами обробника HTTP-запиту. Клієнти BigQuery, Firestore, Cloud Storage, Vertex AI створюються один раз при першому старті екземпляра і переживають обробку кількох послідовних запитів у межах одного теплого контейнера. Цей патерн знижує середню затримку холодного старту і усуває повторні виклики функцій авторизації на кожен запит.

Конфігурацію винесено у змінні оточення, що задаються у скрипті розгортання через прапорець `--set-env-vars`. Для кожної змінної у коді визначено резервне значення за замовчуванням, що дозволяє запускати сервіс у тестовому середовищі без повного переліку змінних. Типовий набір охоплює ідентифікатор проєкту GCP, регіон, ідентифікатор аналітичного датасету BigQuery, ім’я реального часу бази Firestore, назви цільових колекцій і таблиць, публічні імена моделей Gemini і ендпоінтів Vertex AI. Чутливі значення зберігаються у Secret Manager і підтягуються через прапорець `--set-secrets`.

Розгортання виконується окремим скриптом `deploy.sh` з прапорцями `--gen2`, `--runtime=python311`, `--region`, `--memory`, `--timeout`, `--entry-point=main`, `--trigger-http`, `--service-account`, `--set-env-vars`. Сервіси з великим обсягом обробки мають пам’ять від 4 до 8 ГБ і таймаут до 60 хвилин, сервіси мобільного профілю мають меншу пам’ять і таймаут у межах 60 секунд. Довготривалі сервіси розгортаються як Cloud Run через команду `gcloud run deploy`, при цьому спільний шар `functions-framework` робить перехід між двома моделями виконання прозорим для прикладного

коду. Сервісний обліковий запис для кожного сервісу налаштовано за принципом мінімальних привілеїв.

Запуск за розкладом реалізовано через Cloud Scheduler, який надсилає HTTP-виклик або публікує повідомлення у Pub/Sub-топік. Довгі пакетні конвеєри з послідовними викликами кількох сервісів винесено у Cloud Workflows, що описує послідовність кроків у декларативному YAML і забезпечує повторне виконання кроків при тимчасових збоях.

3.3 Головний сервіс пакетної обробки чеків

Головний сервіс пакетної обробки чеків є найбільшим за обсягом програмним компонентом системи. Сервіс реалізовано як єдиний HTTP-ендпоінт, точка входу якого диспетчеризує виклик на один з шести режимів роботи за значенням параметра `mode` у тілі запиту. Такий підхід поєднує спільну інфраструктуру авторизації, журналювання і доступу до сховищ з розділенням прикладної логіки на функціонально самостійні гілки.

Альтернативним варіантом було винесення кожного режиму в окрему Cloud Function, від якого довелось відмовитись з кількох причин. Значний обсяг спільного коду валідації, нормалізації і запису у аналітичне сховище при дублюванні у шести окремих сервісах призвів би до розсинхронізації логіки при кожній зміні схеми даних. Вартість холодного старту Cloud Functions Gen2 також зростала б, оскільки кожен окремий сервіс ініціалізує клієнтів BigQuery, Cloud Storage і Vertex AI, що додає до 2 секунд на холодний виклик. Окрім того, єдиний сервіс з набором режимів дозволяє тримати спільну конфігурацію розгортання, один сервісний обліковий запис і спільні ліміти пам’яті, що знижує операційну складність.

Сервіс підтримує 6 режимів роботи, кожен з яких виконує окремий крок конвеєра і викликається або мобільним клієнтом, або оркестратором за розкладом. Узагальнений перелік режимів і профілів їх використання наведено у таблиці 3.1.

Таблиця 3.1 – Режими роботи головного сервісу пакетної обробки чеків

№	Режим	Тип виклику	Профіль використання
1	Синхронна обробка одного чека	HTTP, мобільний клієнт	Подача чека з зображенням або текстом у тілі запиту
2	Переобробка за ідентифікатором	HTTP, оркестратор	Повторна обробка історичних чеків після оновлення каталогу
3	Постановка пакетного завдання	HTTP, оркестратор	Запуск Vertex AI Batch Prediction і повернення ідентифікатора
4	Повний цикл пакетного опрацювання	HTTP, денний оркестратор	Запуск, очікування і запис результатів у цільові таблиці
5	Визначення міста торговельної точки	HTTP, погодинний оркестратор	Виклик мультимодальної моделі з довідником кандидатів
6	Корекція назв товарів	HTTP, денний оркестратор	Пакетне передбачення з нормалізацією скорочень і одиниць

Третій режим запускає асинхронне пакетне завдання Vertex AI Batch Prediction для обробки накопиченого обсягу чеків за період. Сервіс формує вхідний файл у форматі JSONL, завантажує його у Cloud Storage, надсилає замовлення Vertex AI і повертає клієнту ідентифікатор завдання разом з очікуваним часом завершення. Четвертий режим виконує повний цикл і додатково дочікується завершення завдання, завантажує вихідні файли, постобробляє відповіді і записує результати у цільові таблиці. П’ятий режим визначає місто торговельної точки за метаданими чека через виклик мультимодальної моделі Gemini з переліком кандидатів з довідника міст відповідної країни. Шостий режим виконує корекцію назв товарів через пакетне передбачення з промптом, що інструктує модель нормалізувати скорочення, розшифрувати абревіатури мережі і привести одиниці виміру до канонічного вигляду. Конвеєр пакетного передбачення наведено на рисунку 3.2.

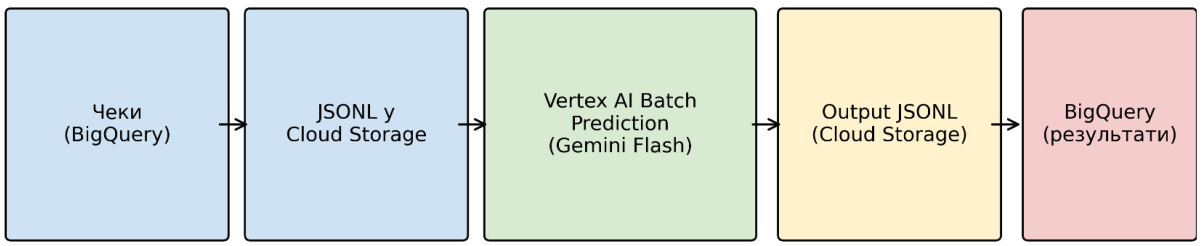


Рисунок 3.2 – Конвеєр пакетного передбачення Vertex AI Batch Prediction для обробки накопичених чеків

Vertex AI Batch Prediction не гарантує порядку рядків відносно вхідного файлу, оскільки служба виконує внутрішній паралелізм. Пряме позиційне зіставлення дає невідповідність у ненульовій частці випадків, що проявляється як помилкове приписування назви товару чужому чеку. Для усунення цього ризику прийнято рішення про явне зіставлення відповідей за внутрішнім унікальним ключем чека, який передається у кожному запиті як окреме JSON-поле і повертається моделлю у структурованій відповіді. Сервіс будує словник відповідей за ключем і приєднує кожну відповідь до відповідного вхідного чека операцією лукапу зі складністю одиничного доступу.

Модель Gemini при визначенні міста може повертати відповіді поза наданими варіантами, зокрема через спробу синтезувати нову назву. Для нейтралізації таких випадків реалізовано окрему функцію нормалізації, що приймає на вхід обраний моделлю рядок і довідник валідних міст. Функція виконує приведення до канонічного написання через нормалізацію Unicode у форму NFC, зняття діакритики, уніфікацію регістру і порівняння з довідником за точним збігом з резервним пошуком за редакційною відстанню Левенштейна за формулою (3.1).

$$d_L(s_1, s_2) = \min(ins, del, sub) \leq 2, \quad (3.1)$$

де d_L(s_1, s_2) – мінімальна кількість операцій вставки (ins), видалення (del) і заміни (sub) символу для перетворення рядка s_1 у рядок s_2. Поріг у 2 операції підібрано як межа, що компенсує типові помилки розпізнавання у назвах міст без хибного злиття семантично різних назв. У разі відсутності збігу результат примусово виставляється у стан невизначеності, а чек помічається прапорцем для повторної обробки на наступному циклі.

Значна частина бізнес-логіки сервісу залежить від країни, у якій здійснено покупку, оскільки від неї визначаються валюта, формат дати, набір валідних торговельних мереж і довідник міст. Прямого поля країни у сирих даних чека у переважній більшості випадків немає. Для відновлення цього атрибута сервіс виконує лукап у довіднику активних промо-кампаній за ідентифікатором кампанії, у рамках якої подано чек. Довідник містить відповідність ідентифікатора кампанії до коду країни за стандартом ISO 3166–1 alpha-2 з обмеженням на прапорець активності кампанії. Отриманий код далі використовується для підбору коду валюти за внутрішньою таблицею відповідностей і для ініціалізації довідника міст моделі визначення міста.

3.4 Сервіс семантичного зіставлення товарів з каталогом

Сервіс зіставлення товарів реалізує ключове архітектурне рішення роботи щодо автоматичного вибору методу векторного пошуку залежно від розміру вхідного пакета. Сервіс приймає HTTP-запит з JSON-тілом, що містить масив чеків з переліком позицій. Для кожної позиції передається нормалізована назва товару, кількість, ціна, ідентифікатор чека, ідентифікатор промо-кампанії. У відповідь сервіс повертає список зіставлених товарів каталогу з оцінкою впевненості, ідентифікатором

кампанії і похідним атрибутом бренду. Контракт виклику однаковий незалежно від обраного всередині сервісу методу пошуку.

Точка входу обчислює кількість чеків у тілі запиту і порівнює її з порогом у 10 чеків. Запит з 10 чеками і менше делегується у Vertex AI Vector Search з заздалегідь розгорнутим ScaNN-ендпоінтом. Запит від 11 чеків переводить обробку на функцію `VECTOR_SEARCH` у BigQuery над таблицею векторних представлень каталогу. Поріг встановлено на основі емпіричних вимірів: до 10 одночасних запитів Vertex AI дає сумарну затримку до 100 мс і виграє у абсолютному часі відповіді, з 11 чеків перевагу за сукупною вартістю отримує пакетний SQL-запит, що виконує обчислення у межах одного сканування таблиці. Перемикання між методами виконується прозоро для викличної сторони і не потребує жодного прапорця у запиті. Логіку автоматичного вибору методу пошуку наведено на рисунку 3.3.

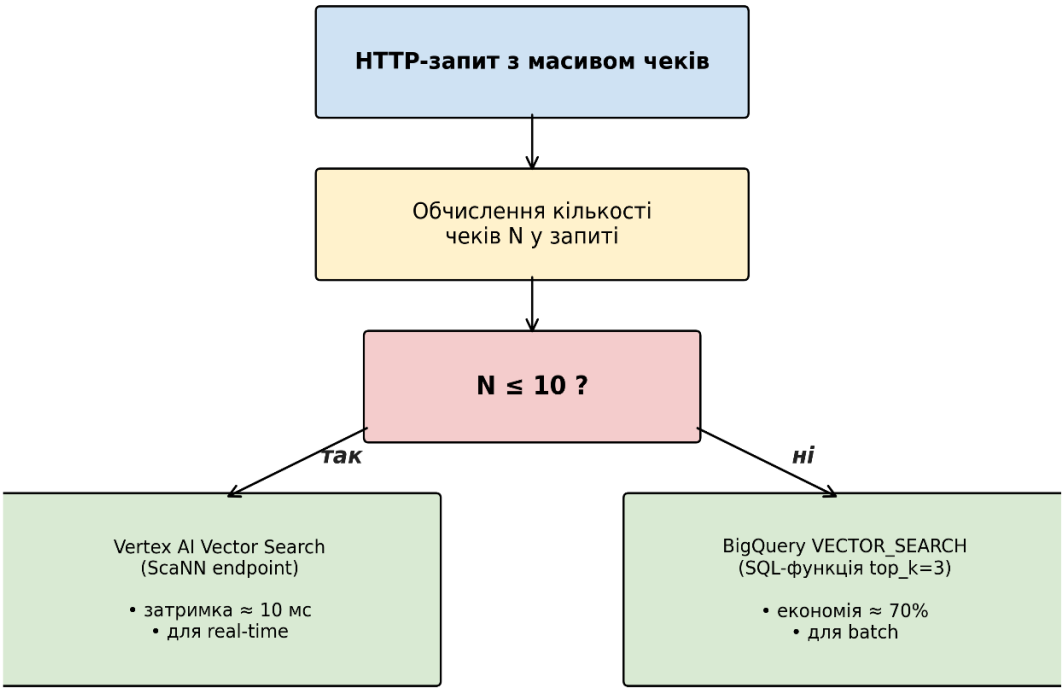


Рисунок 3.3 – Логіка автоматичного вибору методу векторного пошуку залежно від обсягу запиту

Якість векторного пошуку безпосередньо залежить від однорідності текстових представлень. Сирі назви з чеків містять скорочення, абревіатури мереж, числові коди артикулів, фрагменти одиниць виміру і слипання слів через відсутність пробілів. Перед обчисленням вкладення сервіс застосовує конвеєр нормалізації з кількох кроків: приведення до нижнього регістру з урахуванням специфіки турецької і литовської орфографії, нормалізацію Unicode у форму NFC зі зняттям керуючих символів і непарних модифікаторів, видалення регулярними виразами числових артикулів, одиниць виміру і цінових фрагментів, а також згортання повторних пробілів і обрізку країв.

Симетрична нормалізація застосовується до назв товарів каталогу на етапі оновлення векторної таблиці. Це гарантує, що вкладення запиту і вкладення каталогу перебувають у однорідному підпросторі.

Векторні представлення обчислюються багатомовною моделлю Vertex AI Text Embeddings розмірністю 768. Для каталогу обчислення виконується пакетно через функцію `ML.GENERATE_TEXT_EMBEDDING` у BigQuery, результат зберігається у колонці типу `ARRAY<FLOAT64>`. Оновлення таблиці виконується у режимі повного перезапису на кожному денному циклі оркестратора, що усуває потребу в інкрементальних апдейтах. Для режиму реального часу вкладення запиту обчислюється синхронним викликом тієї самої моделі через SDK `google-cloud-aiplatform` за час близько 40 мс на одну позицію.

Семантичну схожість вкладень обчислено за косинусною метрикою згідно з формулою (3.2). Геометричну інтерпретацію косинусної подібності наведено на рисунку 3.4. Зазначена візуалізація дозволяє наочно продемонструвати математичний зміст порівняння векторів у багатовимірному просторі вкладень. Це сприяє глибшому розумінню механізмів семантичного зіставлення, покладених в основу розробленого алгоритму.

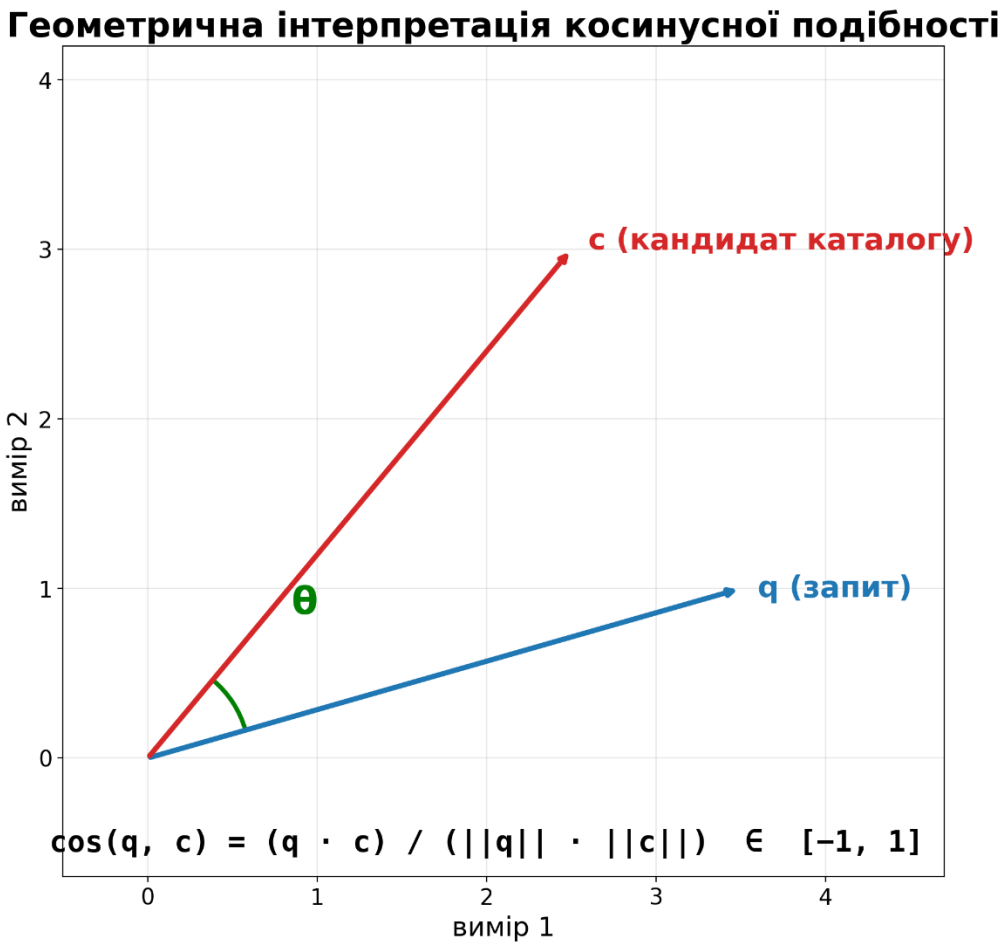


Рисунок 3.4 – Геометрична інтерпретація косинусної подібності двох векторів вкладень

$$\cos(q, c) = (q \cdot c) / (||q|| \cdot ||c||), \quad (3.2)$$

де q – вектор-запит, c це вектор-кандидат каталогу;

$q \cdot c$ – скалярний добуток;

$||q||$ і $||c||$ – евклідові норми. Значення лежить у діапазоні [-1, 1], причому близькі за змістом тексти дають значення, близьке до одиниці.

Ефективність векторного пошуку оцінюється метрикою повноти top-k за формулою (3.3).

$$\text{recall@k} = |H \cap R_k| / |H|, \quad (3.3)$$

де H – множина істинних відповідних товарів каталогу для запиту;

R_k – множина перших k кандидатів, повернутих індексом. Конфігурація ScaNN-індексу підібрана під цільовий рівень recall@10 не нижче 0,95 на еталонному наборі запитів, що відповідає бюджету часу відгуку 10 мс на одну позицію.

У гілці реального часу сервіс викликає ендпоінт Vertex AI Vector Search методом `find_neighbors`, який приймає список запитових векторів і повертає масив найближчих сусідів з обчисленою дистанцією. Сервіс запитує до 10 найближчих сусідів на позицію з наступним фільтруванням за порогом косинусної подібності і приєднанням метаданих каталогу через лукап у аналітичній таблиці.

Фільтрація за активними промо-кампаніями виконується на рівні `WHERE`-предикату при лукапі метаданих, оскільки пряма фільтрація на рівні індексу вимагала б переіндексування при кожній зміні статусу кампанії.

Пакетна гілка оформлена як єдиний SQL-запит, що приєднує тимчасову таблицю вкладень запитів до таблиці векторних представлень каталогу через функцію `VECTOR_SEARCH` з параметром `top_k=3`. Виконання такого запиту для пакета у кілька тисяч позицій займає від одиниць до десятків секунд і коштує лише обсяг сканування таблиці векторів, без додаткової тарифікації за кількість запитів.

Для подальшої бізнес-логіки нарахування винагород сервіс переводить числове значення косинусної дистанції у 3 дискретні рівні впевненості. Класифікація рівнів і відповідні правила нарахування винагороди наведено у таблиці 3.2. Запровадження подібної градації забезпечує необхідну гнучкість алгоритму ухвалення рішень та дозволяє ефективно мінімізувати ризики хибних нарахувань у неоднозначних ситуаціях.

Таблиця 3.2 – Рівні впевненості зіставлення позицій чека з каталогом

Рівень	Діапазон косинусної подібності	Правило нарахування
Високий	вище 0,85	Автоматичне нарахування винагороди
Середній	від 0,60 до 0,85	Додаткова бізнес-валідація або резервний канал
Низький	нижче 0,60	Виключення позиції з нарахування

Пороги калібровано на розмічених даних експериментального дослідження з Розділу 4 і підібрано так, щоб частка хибнопозитивних автоматичних нарахувань не перевищувала 1%.

3.5 Сервіс обробки у режимі реального часу чеків

Сервіс обробки у режимі реального часу обслуговує сценарій миттєвого сканування чека у мобільному застосунку. На відміну від головного сервісу пакетної обробки, цей компонент оптимізовано під мінімальну затримку на одному запиті і відокремлено у самостійний виконавчий блок з мінімальним набором залежностей, щоб зменшити час холодного старту Cloud Functions Gen2 і тримати середню тривалість відповіді у межах 2–3 секунд.

Сервіс підтримує три варіанти обробки, що покривають різні профілі вхідних даних і відрізняються способом отримання тексту з зображення. Зведене порівняння варіантів наведено у таблиці 3.3. Наявність кількох альтернативних шляхів виконання дозволяє системі гнучко адаптуватися до різноманітних умов експлуатації мобільного клієнта. Це забезпечує оптимальний баланс між швидкістю відповіді, обчислювальною вартістю та підсумковою точністю розпізнавання для кожного конкретного запиту. Крім того, такий диференційований підхід суттєво підвищує загальну відмовостійкість конвеєра при роботі з нестандартними або низькоякісними зображеннями.

Таблиця 3.3 – Варіанти обробки у режимі реального часу чеків

Варіант	Спосіб OCR	Спосіб структурування	Час відгуку, с	Профіль застосування
Гібридний	Cloud Vision DOCUMENT_TEXT_DETECTION	Gemini 2.5 Flash, синхронно	2,0–3,0	Низькоякісні зображення, нестабільна мережа
Мультимодальний	Gemini Vision, об’єднано	Gemini Vision, об’єднано	1,5–2,5	Якісні зображення, оптимізація вартості
Текстовий	Виконується на клієнті	Gemini 2.5 Flash-Lite	близько 1,0	Готовий OCR-текст, найдешевший шлях

Гібридний варіант послідовно викликає Cloud Vision API у режимі `DOCUMENT_TEXT_DETECTION` для отримання сирого OCR-тексту і далі передає цей текст мультимодальній моделі Gemini у синхронному виклику. Мультимодальний варіант передає зображення безпосередньо у Gemini Vision з інструкцією виконати одночасно OCR і структурування. Текстовий варіант приймає готовий OCR-текст, розпізнаний на стороні клієнта, і виконує лише фазу структурування. Вибір варіанта виконується параметром у тілі запиту і не потребує окремих ендпоінтів, що дозволяє мобільному клієнту адаптивно перемикати стратегію залежно від якості мережі і зображення.

Текстовий варіант реалізовано як окремий Cloud Functions-сервіс з мінімальним списком залежностей. Винесення дає менший образ контейнера і швидший холодний старт у межах 300–400 мс, незалежне масштабування і тарифікацію з меншою пам’яттю і таймаутом, а також окремий моніторинг показників затримки і помилок, що дозволяє ізольовано оцінювати економіку найдешевшого сценарію обробки.

Контракт виклику текстового сервісу узгоджений з гібридним і мультимодальним варіантами, що робить перехід між ними на стороні клієнта прозорим.

Початкова реалізація сценарію реального часу була побудована поверх головного сервісу пакетної обробки через простий синхронний виклик з параметром обсягу у один чек. Така реалізація показала середній час відгуку близько 19 секунд, оскільки пакетна гілка несла з собою повний набір перевірок і кроків, орієнтованих на обсяги у тисячі чеків. Переробка сценарію у самостійний сервіс з прямим викликом мультимодальної моделі і полегшеною корекцією дала скорочення часу відгуку приблизно у 7 разів до 2–3 секунд. Скорочення досягнуто ліквідацією зайвих проміжних записів у аналітичне сховище, що використовувались у пакетному сценарії для відновлення після збою, переходом на синхронну модель виклику Gemini замість постановки у чергу пакетного передбачення, використанням моделі Gemini 2.5 Flash-Lite у полегшеному режимі корекції замість повної моделі Flash з порівнянною якістю на коротких текстах чека і скороченням часу інференсу приблизно удвічі, а також зменшенням обсягу системного промпта і відмовою від кількох демонстраційних прикладів.

Результати роботи сервісу записуються у дві колекції реального часу Firestore з однаковим ідентифікатором документа. Одна колекція містить сирий результат розпізнавання з OCR-текстом, проміжними кандидатами, часовими мітками і службовими метаданими виклику моделі. Інша колекція містить фінальну структуровану відповідь, готову до споживання мобільним клієнтом, з переліком позицій, загальною сумою, датою, назвою торговельної точки і визначеним містом.

Розділення на дві колекції обґрунтоване кількома міркуваннями. Колекції мають різний життєвий цикл даних: сира використовується для пост-аналізу якості моделі і може бути очищена через період зберігання, тоді як фінальна обслуговує мобільні підписки і зберігається довго. Вони відрізняються і за обсягом документа, оскільки сирий може досягати сотень

кілобайт, а фінальний обмежений десятками, тому тримання їх в одному документі роздуло б ліміт у мобільному клієнті. Колекції мають також різні права доступу: мобільний клієнт отримує доступ лише до фінальної, а сира доступна лише службовим сервісам моніторингу якості.

Ідентифікатор документа генерується детерміновано з ідентифікатора чека і ідентифікатора клієнтської сесії, що гарантує простий лукап пари документів і усуває потребу в окремому поєднувальному ключі.

Внутрішня логіка сервісу організована як лінійний конвеєр. На першому кроці виконується прийом і валідація вхідного запиту з перевіркою розміру зображення або довжини тексту і видобутком метаданих з HTTP-заголовків. Далі виконується OCR одним з трьох варіантів з попередньою ініціалізацією клієнта на рівні модуля. Після цього викликається Gemini для структурування з примусовим JSON-висновком через параметр response_schema SDK Vertex AI. Завершальний крок виконує паралельний запис у дві колекції реального часу Firestore через асинхронний клієнт і формує HTTP-відповідь клієнту.

Конвеєр виконується у синхронному HTTP-контракті без проміжних черг і асинхронних тригерів. Вибір такої моделі зумовлений вимогою мобільного застосунку отримати фінальну відповідь у тій самій сесії без опитування і без постійного з’єднання з чергою подій.

3.6 Модель даних BigQuery і сервіс побудови агрегатів

Логічну модель аналітичного сховища організовано у три шари, що утворюють спадну піраміду від сирих фактичних записів до агрегованих представлень для візуалізації. Транзакційний шар містить сирі факти чеків, подій сканування і подій входу у плоскому вигляді, придатному для подальших SQL-операцій; його записи формуються ETL-сервісами з документів Firestore і дедублікуються за ідентифікатором документа. Оброблений шар утворюється детерміновано з транзакційних даних і

оновлюється за денним або погодинним розкладом; до нього належать таблиця опрацьованих чеків з нормалізованими позиціями, таблиця скоригованих назв товарів, таблиця векторних представлень каталогу, довідник торговельних точок з географічною атрибуцією, довідникові таблиці країн, міст і продуктів, а також факт-таблиця сканувань. Агрегатний шар обслуговує дашборди Looker Studio і містить таблицю денних агрегатів, набір візуалізаційних таблиць з географічним розрізом, аналізом торговельних точок, крос-брендовою коокуренцією у кошику і коокуренцією категорій першого і другого рівня, а також довідник активних промо-кампаній.

Розділення на шари дає можливість переобчислювати будь-який верхній шар з нижнього без повторного звернення до первинних документних джерел. Логічну модель сховища у вигляді діаграми сутність-зв’язок наведено на рисунку 3.5.

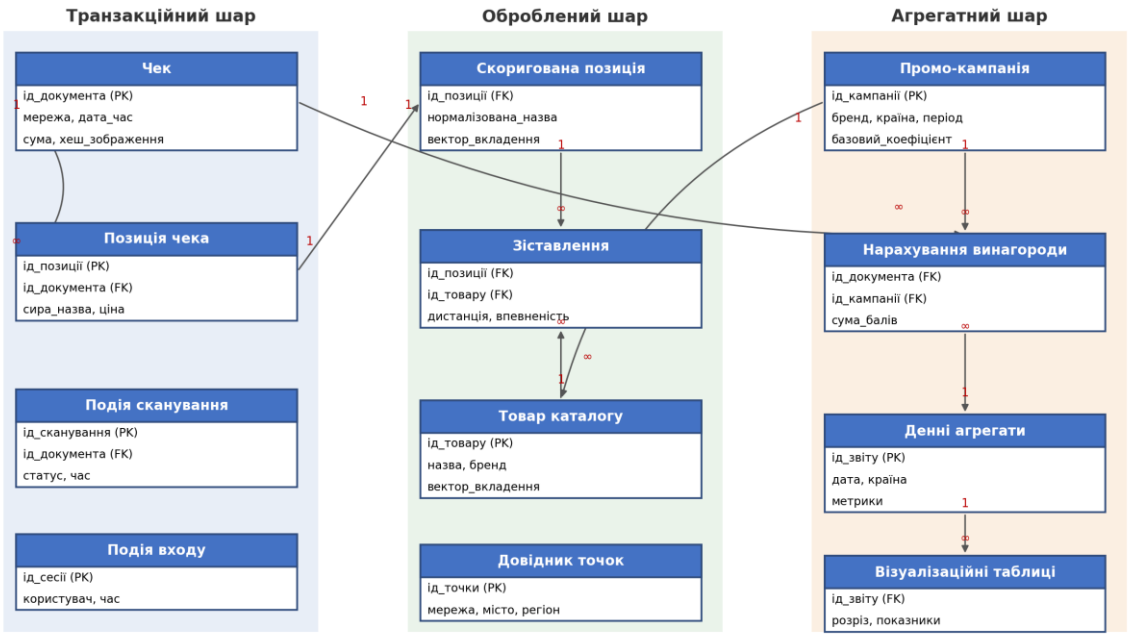


Рисунок 3.5 – Логічна модель даних аналітичного сховища (ERD)

Сервіс побудови агрегатів реалізовано як єдину Cloud Function, що викликається денним оркестратором після завершення кроків нормалізації і зіставлення. Сервіс послідовно виконує набір SQL-запитів, кожен з яких будує одну цільову таблицю третього шару. Запити винесено у окремі Python-функції, що приймають на вхід датасет-префікс і часове вікно і повертають ідентифікатор виконаної BigQuery-джоби. Така організація дає прозору діагностику (збій у одному запиті не зупиняє решту, а залишає у журналі точну причину і ідентифікатор джоби), можливість вибіркового перезапуску окремих агрегатів і простий контракт додавання нової візуалізаційної таблиці через одну функцію.

Таблиця денних агрегатів оновлюється у режимі видалення і повторної вставки за ідентифікатором звіту, що охоплює країну, період і ідентифікатор промо-кампанії. На початку побудови сервіс виконує `DELETE` з фільтром за поточним ідентифікатором і далі вставляє повний набір рядків. Такий режим забезпечує ідемпотентність повторних викликів без появи дублів і без потреби у складних `MERGE`-операторах. Візуалізаційні таблиці з географічним і категорійним розрізом записуються у режимі повного перезапису на кожному денному циклі, оскільки їх обсяг залишається у межах десятків або сотень МБ і повне переобчислення за добу займає хвилини. Довідникові таблиці країн, міст і продуктів оновлюються через `MERGE` за природним ключем, що зберігає історію записів і дозволяє відстежувати зміни атрибутів у часі.

Часові вікна агрегатів обчислюються у поясі Europe/Warsaw, що відповідає основній географії користувачів у Центральній Європі. Перетворення між UTC-часовими мітками сирих подій і локальним днем звіту виконується виразом `TIMESTAMP_TRUNC(ts, DAY, 'Europe/Warsaw')` з явним зазначенням поясу у кожному запиті, що усуває залежність від регіональних налаштувань проєкту.

Кожна таблиця оброблених і агрегатних шарів має визначений природний ключ і документовану схему. Таблиця опрацьованих чеків має

ключем комбінацію ідентифікатора чека і ідентифікатора промо-кампанії, що дозволяє одному й тому самому чеку фігурувати у кількох кампаніях з різними зіставленнями. Факт-таблиця сканувань має ключем комбінацію ідентифікатора чека і порядкового номера позиції. Довідник торговельних точок має ключем нормалізовану назву мережі у комбінації з країною. Зв’язки між таблицями реалізовано через цільові поля без формальних обмежень цілісності, оскільки BigQuery не підтримує `FOREIGN KEY` у виконавчому шарі. Валідація виконується на етапі побудови агрегатів через `LEFT JOIN` з перевіркою відсутності `NULL` у ключових полях і журналюванням кількості відкинутих рядків.

Для реагування на дефекти даних побудовано явний патерн виправлення. У разі виявлення чека з некоректним зіставленням сервіс моніторингу ініціює видалення відповідних записів з оброблених таблиць за ідентифікатором чека і повторний запуск головного сервісу у режимі переобробки. Після перебудови оброблених даних оркестратор повторно запускає сервіс побудови агрегатів з обмеженням на відповідний період, що у режимі видалення і повторної вставки приводить агрегатні таблиці у консистентний стан без втручання у решту історії.

Верхній шар аналітичної піраміди представлено набором інтерактивних дашбордів у Looker Studio, що споживають агрегатні таблиці BigQuery через нативний конектор. Дашборди оновлюються щодня після завершення денного циклу і доступні маркетинговим командам через захищене посилання без прямого доступу до BigQuery.

Перший дашборд (рисунок 3.6) відображає загальну активність учасників системи. Ключові показники охоплюють сукупну кількість зареєстрованих користувачів у розрізі промо-кампаній, денну і тижневу динаміку нових реєстрацій, рівень утримання учасників за когортами тижня реєстрації і частку активних гравців, що здійснили хоча б одне сканування у звітному місяці. Графік динаміки сканувань дозволяє виявляти піки активності, що корелюють з медіапросуванням кампанії.

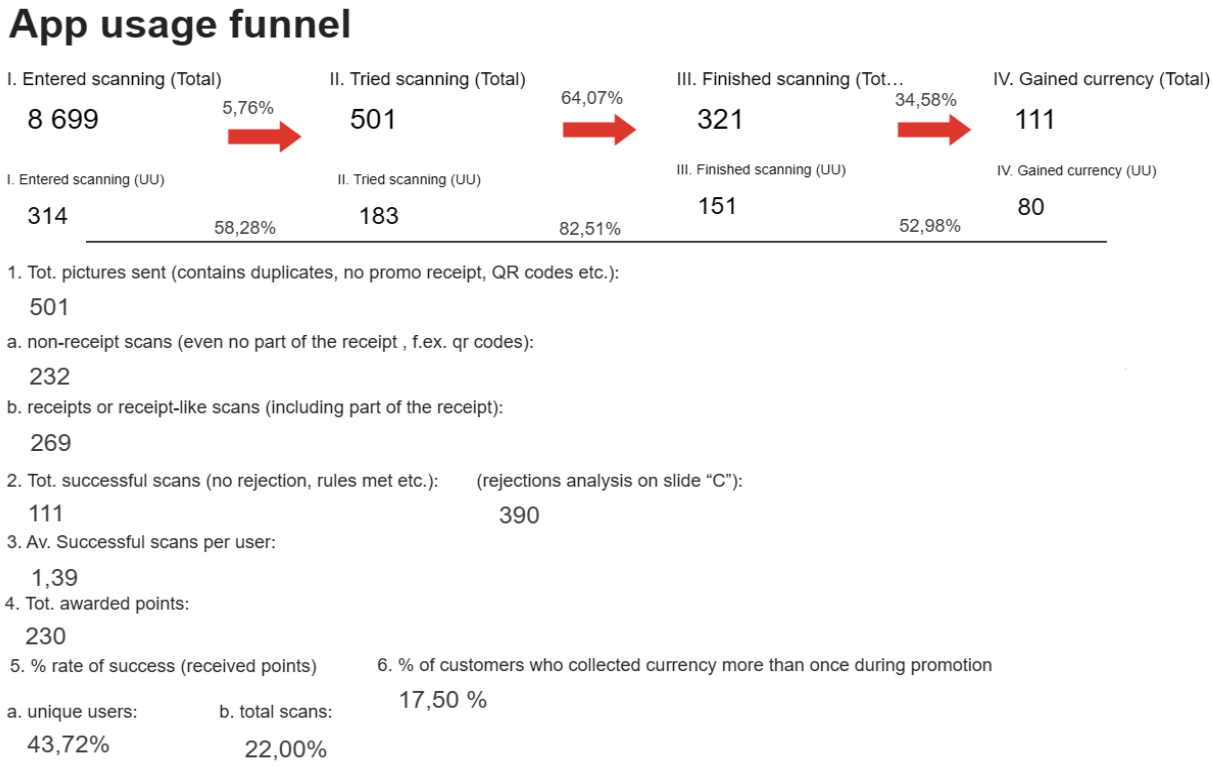


Рисунок 3.6 – Дашборд воронки залученості користувачів (Looker Studio)

Другий дашборд (рисунок 3.7) забезпечує географічну аналітику. Карта охоплення відображає щільність сканувань у розрізі регіонів і міст з колірним кодуванням за кількістю унікальних чеків. Таблиця регіонів ранжує географічні одиниці за загальною кількістю сканувань, кількістю унікальних учасників і середньою сумою чека. Порівняльна панель країн показує частку кожної ринкової географії у сукупному обсязі системи, що дозволяє оцінювати ефективність окремих ринкових запусків. Розподіл чеків за топ-містами країни наведено на рисунку 3.8. Завдяки такому комплексному представленню аналітики мають змогу оперативно виявляти регіональні ринки з найвищою концентрацією споживчої активності. Це, у свою чергу, забезпечує надійне підґрунтя для точкового планування та ефективного розподілу маркетингових бюджетів на локальних рівнях.

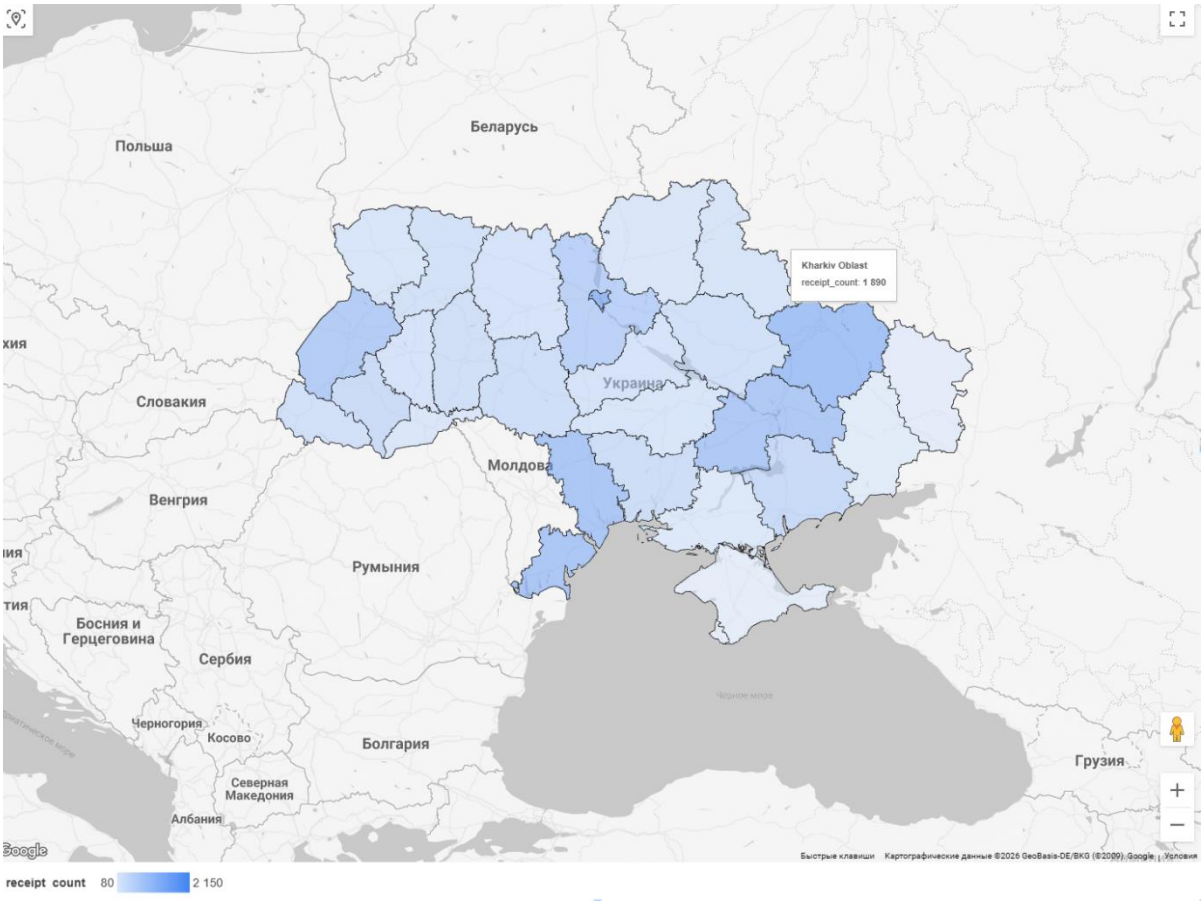


Рисунок 3.7 – Дашборд географічної аналітики: щільність сканувань за регіонами (Looker Studio)

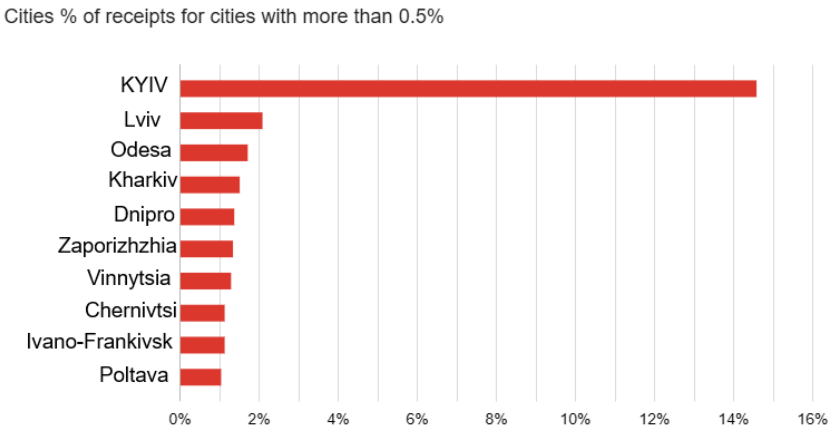


Рисунок 3.8 – Розподіл чеків за топ-містами країни (Looker Studio)

Третій дашборд (рисунок 3.9) присвячено аналізу торговельних точок. Рейтинг мереж за кількістю сканувань відображає топ-двадцять торговельних мереж у межах активної промо-кампанії з зазначенням частки кожної мережі у загальному обсязі. Поряд відображається середня сума чека і частка чеків, де знайдено хоча б одну відповідну промо-позицію. Деталізація на рівні окремого магазину дозволяє виявляти торговельні точки з аномально низьким або високим рівнем конверсії промо-позицій відносно їхньої ринкової долі. Розподіл чеків за категорією розміру населеного пункту наведено на рисунку 3.10.

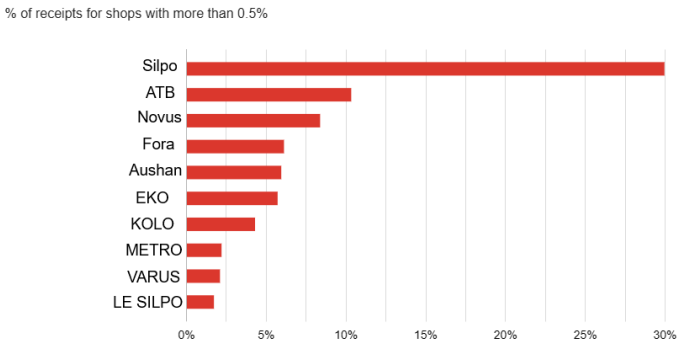


Рисунок 3.9 – Дашборд топ-мереж за часткою чеків (Looker Studio)

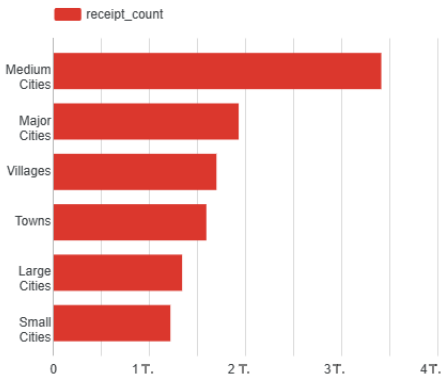


Рисунок 3.10 – Розподіл чеків за категорією розміру населеного пункту (Looker Studio)

Четвертий дашборд (рисунок 3.11) відображає товарну аналітику: рейтинг категорій першого і другого рівня за частотою появи у чеках, матрицю спільної появи брендів у одному кошику, рейтинг найчастіше купованих продуктів у рамках кампанії. Матриця коокуренції брендів дозволяє виявляти природні товарні пари з позиції поведінки покупця, що використовується для проектування крос-брендових промо-механік.

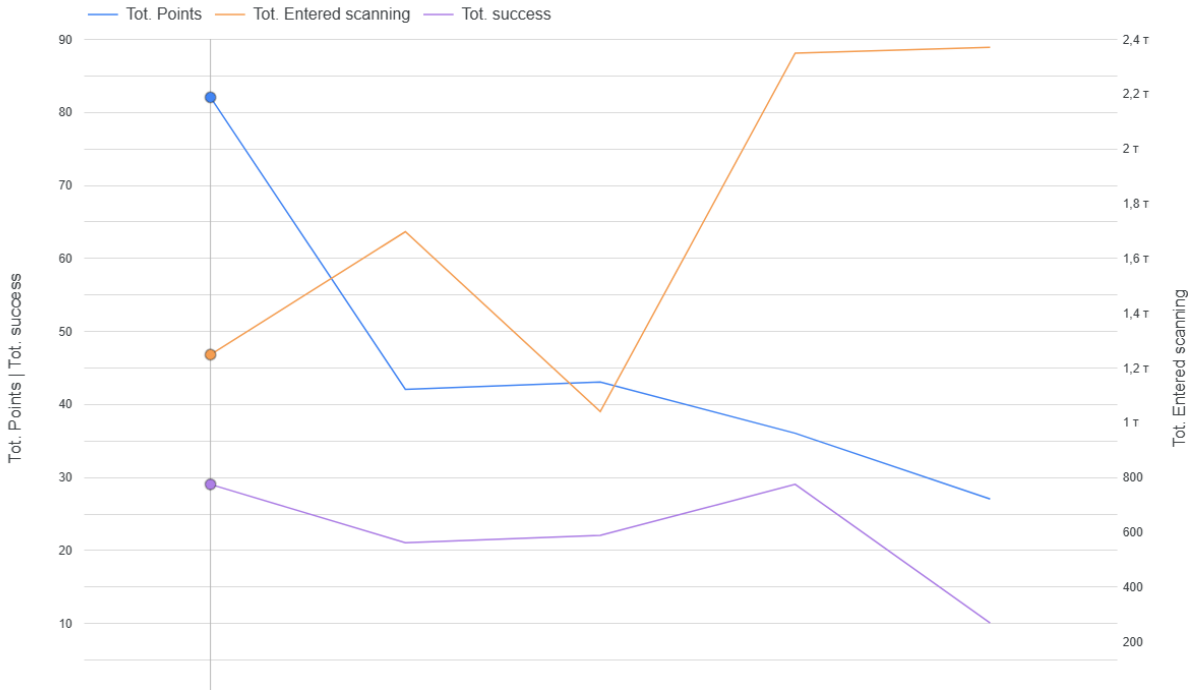


Рисунок 3.11 – Динаміка ключових метрик промо-кампанії: нараховані точки, входи у сканування, успішні скани (Looker Studio)

Усі дашборди підтримують фільтрацію за промо-кампанією, країною, торговельною мережею і часовим діапазоном через єдину панель фільтрів. Зміна будь-якого фільтра миттєво оновлює усі графіки і таблиці поточного дашборду через новий SQL-запит до BigQuery.

3.7 Гейміфікаційні механіки і алгоритм нарахування винагород

В основу ігрового шару покладено модель гачка Нір Еяля [7], за якою стійка поведінкова петля формується послідовністю з 4 стадій: тригер, дія, змінна винагорода і інвестиція користувача у систему. Тригером у запропонованій системі виступає пуш-повідомлення або інтерфейсна підказка у мобільному застосунку, дією є сканування касового чека, змінною винагородою є нарахування ігрової валюти і віртуальних знаків розрізнення з непередбачуваним розміром, інвестицією є накопичення історії покупок і персонального профілю. Модель гачка у застосуванні до системи гейміфікованої лояльності наведено на рисунку 3.12.

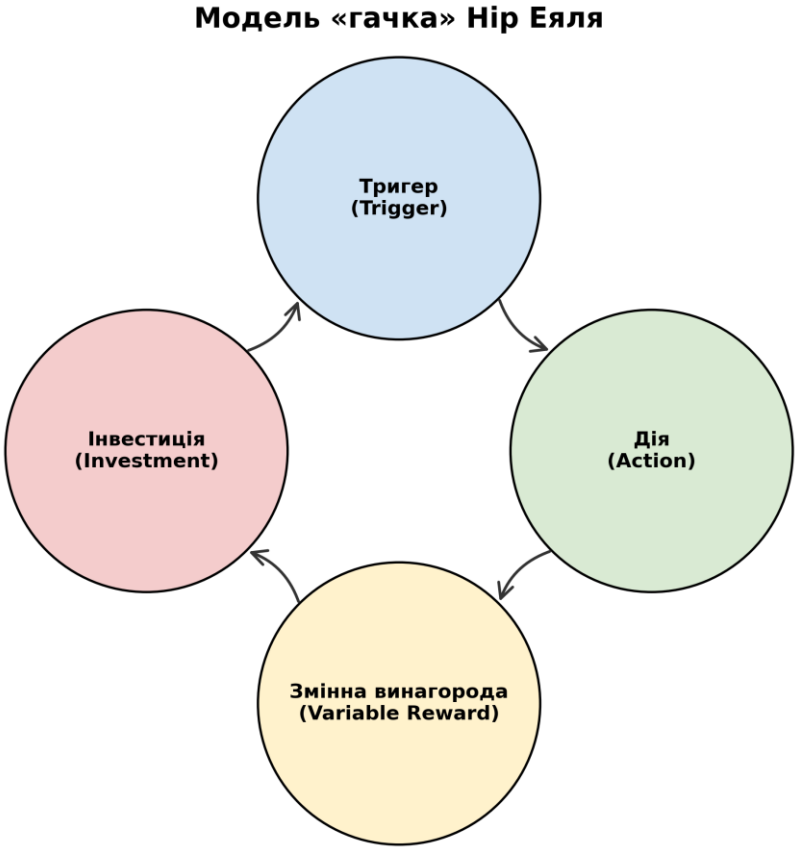


Рисунок 3.12 – Модель «гачка» Нір Еяля у застосуванні до системи гейміфікованої лояльності

Для практичної реалізації петлі прийнято п’ять паралельних механік, вибір яких обґрунтовано мета-аналітичною роботою Hamari та ін. [29]. Миттєва винагорода за кожне успішне сканування чека дає негайний зворотний зв’язок і короткостроковий позитивний стимул. Шкали прогресу для довгих челенджів з кумулятивним виконанням умов за тиждень або місяць працюють на середньостроковому горизонті. Віртуальні знаки розрізнення за рідкісні комбінації товарів у кошику додають колекційний елемент. Персональні місії на основі історії покупок конкретного користувача генеруються з урахуванням його переваг. Лідерборди на рівні міст і регіонів вводять соціальний шар через неявне змагання.

Розмір винагороди за чек обчислюється як добуток базового коефіцієнта кампанії на суму позицій чека, що відповідають умовам кампанії, з подальшим накладенням модифікаторів. Формально винагорода для одного чека визначається співвідношенням (3.4).

$$R = B \cdot S \cdot M_q \cdot M_s \cdot M_p, \tag{3.4}$$

де R – підсумкова винагорода у ігрових одиницях;

B – базовий коефіцієнт активної кампанії з довідника;

S – сума позицій чека, що зіставилися з товарами кампанії з високим рівнем впевненості;

M_q – мультиплікатор квесту;

M_s – сезонний мультиплікатор вихідних і святкових днів;

M_p – персональний мультиплікатор новачка для перших 3 чеків нового користувача.

Базовий коефіцієнт зчитується з поля у довіднику активних промо-кампаній і лежить у межах від 5 до 20 ігрових одиниць на євро сплаченої суми. Мультиплікатор квесту приймає значення 1 або 1,5 залежно від відповідності сканування активній сезонній події. Сезонний мультиплікатор приймає значення 1 або 1,2 у вихідні і святкові дні. Персональний

мультиплікатор новачка приймає значення 1 або 2 для 3 перших чеків і далі завжди дорівнює 1.

На рівні окремої позиції чека алгоритм нарахування виконує послідовність перевірок. Спершу позиція зіставляється з товаром каталогу промо-кампанії з рівнем впевненості не нижче високого згідно з класифікацією з підрозділу 3.4, тоді як позиції з середнім і низьким рівнем відкидаються. Далі перевіряється відповідність чека активним кампаніям відповідної країни на момент покупки через лукап у довіднику активних кампаній за комбінацією країни, бренду і часового вікна. Окремо виконується бізнес-валідація щодо мінімальної суми чека, допустимого часового вікна між датою покупки і датою подання та допустимої торговельної мережі. Завершальною є звірка з таблицею опрацьованих фіскальних ідентифікаторів для відкидання повторних подань.

Позиції, що пройшли всі перевірки, підсумовуються і передаються у формулу розрахунку загальної винагороди.

Центральним регістром поточного стану ігрової моделі є довідник активних промо-кампаній, що зберігається у BigQuery і синхронізується з продуктовою базою Firestore через окремий ETL-сервіс. Довідник містить для кожної кампанії ідентифікатор, назву бренду, країну, дату початку і завершення, базовий коефіцієнт винагороди, посилання на каталог товарів, прапорці активності і візуального відображення у мобільному клієнті. Кожна операція нарахування виконує лукап за ідентифікатором кампанії і перериває нарахування у разі відсутності запису або його неактивності. Такий дизайн робить довідник єдиним джерелом істини для ігрової економіки і дозволяє маркетинговим командам вмикати і вимикати кампанії без перерозгортання сервісів.

Ігровий шар інтегровано з підсистемою виявлення шахрайства, описаною у підрозділі 2.5. Перед обчисленням винагороди сервіс нарахування запитує вердикт за ідентифікатором чека. У разі високого рівня ймовірності підробки нарахування блокується і чек переводиться у чергу

ручної верифікації. У разі середнього рівня нарахування виконується, однак винагорода резервується у відкладеному стані до підтвердження модератора. Додатково реалізовано ліміт на добову суму ігрової валюти, нараховану одному користувачу, як функція від середнього історичного чеку користувача. Перевищення ліміту автоматично переводить подальші чеки у стан відкладеної верифікації без відмови у сервісі.

Результат нарахування повертається мобільному клієнту у тілі синхронної HTTP-відповіді сервіса обробки у режимі реального часу і записується у дві сутності: журнал дій користувача у Firestore (живить інтерфейс стрічки активності і пуш-повідомлення) і факт-таблицю нарахувань у BigQuery (джерело для аналітичних дашбордів виробників брендів і внутрішньої звітності).

3.8 Підсистема безпеки і програмна реалізація виявлення шахрайства

Підсистема безпеки об’єднує чотирирівневу стратегію протидії зловживанням, обґрунтовану у підрозділі 2.5, у вигляді окремого сервісу з єдиним HTTP-ендпоінтом. Сервіс інтегрується з головним конвеєром реального часу і пакетної обробки через синхронний виклик і повертає структурований вердикт, який далі використовується сервісом нарахування винагород. Загальну схему чотирирівневого конвеєра виявлення шахрайства наведено на рисунку 3.13. Такий підхід до архітектурної ізоляції безпекових функцій дозволяє централізовано оновлювати алгоритми перевірки без необхідності втручання у роботу інших компонентів системи. Це критично важливо для оперативного та ефективного реагування на появу нових, більш складних патернів шахрайської поведінки. Завдяки цьому гарантується високий рівень цілісності ігрової економіки та надійно захищаються інтереси виробників брендів від нецільового витрачання маркетингових бюджетів.



Рисунок 3.13 – Чотирирівневий конвеєр виявлення шахрайства з метакласифікатором XGBoost

Сервіс виявлення шахрайства оформлено як Cloud Run-застосунок через те, що завантаження моделі Multi-Stream EfficientNet потребує від 2 до 3 секунд на старті контейнера і тримання теплого екземпляра дає помітний виграш у часі відгуку. Модель завантажується один раз у стартовій процедурі і тримається у пам’яті процесу весь час життя екземпляра. Точка входу прийняла HTTP-контракт з JSON-тілом, що містить ідентифікатор чека, посилання на зображення у Cloud Storage, фіскальні реквізити, ідентифікатор користувача і ідентифікатор сесії. У відповідь сервіс повертає три поля: числову ймовірність підробки у межах від 0 до 1, дискретний рівень ризику зі значеннями `low`, `medium` або `high` і перелік виявлених артефактів з коротким текстовим обґрунтуванням.

Перший рівень захисту оформлено як SQL-лукап у таблиці опрацьованих фіскальних ідентифікаторів з унікальним ключем за

комбінацією номера фіскального реєстратора, дати, часу і суми чека. Запит виконується через клієнт `bigquery.Client` з параметризованими полями для захисту від SQL-ін'єкцій і завершується у межах десятків мілісекунд. У разі знаходження збігу сервіс одразу повертає вердикт `high` без запуску подальших кроків.

Паралельно з фіскальним лукапом виконується обчислення перцептивного хешу зображення через бібліотеку `imagehash` з наступним пошуком у таблиці хешів раніше опрацьованих чеків за відстанню Геммінга. Дублікатом вважається пара зображень, для яких виконується умова (3.5).

$$d_H(h_1, h_2) = \Sigma\, |b_1[i] \oplus b_2[i]| \leq 5, \qquad (3.5)$$

де h_1 і h_2 – 64-бітні pHash-вектори двох зображень;

b_1[i] і b_2[i] – їх біти;

$\oplus$ – операція XOR, поріг 5 одиниць підібрано емпірично як межа, нижче якої зображення вважаються візуально ідентичними з точністю до перестиснення. Обидва лукапи виконуються у паралельних завданнях `asyncio`, що дає змогу використати мережеву затримку BigQuery для паралельного обчислення pHash.

Другий рівень виконує візуальну класифікацію зображення на справжнє і підроблене через згорткову нейромережу Multi-Stream EfficientNet, донавчену на спеціалізованому корпусі чеків. До базової архітектури Multi-Stream EfficientNet додано шар SRM-фільтрами після третього блоку, що посилює градієнти у зонах з різкими перепадами яскравості, характерними для країв вставлених у графічному редакторі фрагментів. Модель навчено протягом 20 епох на збалансованому корпусі з 1949 справжніх і 1949 підроблених зразків з оптимізатором AdamW і швидкістю навчання $1 \cdot 10^{-4}$. На валідаційній вибірці досягнуто точність 94,64% і AUC на рівні 0,9805. Код інференсу оформлено як тонку Python-обгортку над моделлю TensorFlow, що приймає попередньо

оброблене зображення розміром 384×384 пікселі і повертає softmax-розподіл імовірностей. Обробка одного зображення на екземплярі Cloud Run з двома ядрами і 4 ГБ пам’яті займає 30–50 мс.

Третій рівень активується лише для зразків, у яких ймовірність підробки за першою моделлю потрапляє у зону невизначеності від 0,20 до 0,85. Такий відбір різко скорочує кількість викликів дорожчої мультимодальної моделі. Сервіс формує запит до Gemini 2.5 Flash з зображенням і текстовим промптом, що перелічує типові ознаки маніпуляції, серед яких кольоровий текст на чорно-білому чеку, різне згладжування символів у сусідніх рядках, гало навколо вставлених фрагментів, сліди інструменту клонування, непослідовні шрифти. Відповідь моделі примушується до структурованого JSON через параметр `response_schema` з полями ймовірності, переліку виявлених артефактів і обґрунтування. Час відгуку становить 10–13 секунд.

Зведення результатів CNN і Gemini виконується метакласифікатором XGBoost, який приймає на вхід дві ймовірності і повертає фінальну скалярну оцінку за формулою (3.6).

$$P_final = w_CNN \cdot p_CNN + w_Gemini \cdot p_Gemini, \qquad (3.6)$$

де P_final – підсумкова ймовірність підробки;

p_CNN і p_Gemini – ймовірності відповідних моделей;

w_CNN = 0,4 і w_Gemini = 0,6 – ваги, навчені на валідаційній вибірці. Чек класифікується як підробка за умови P_final ≥ 0,55. Поріг прийняття рішення підібрано з оптимізацією метрики F1 за формулою (3.7).

$$F1 = 2 \cdot (Precision \cdot Recall) / (Precision + Recall), \qquad (3.7)$$

де Precision – частка справжніх підробок серед усіх класифікованих як підробка, Recall – частка виявлених підробок серед усіх наявних у вибірці.

Комбінація CNN і Gemini через XGBoost знижує частку хибнопозитивних виявлень з 50% у режимі чистої CNN до 20% у гібридному режимі і підтримує Recall на рівні не нижчому за 80%. Середня затримка для однозначних випадків, вирішуваних на CNN, становить близько 50 мс. Для зразків, що потребують резервного виклику Gemini (приблизно 15% потоку), затримка становить 13–15 секунд, що дає прийнятну середню затримку у межах 2 секунд.

Четвертий рівень реалізовано у двох частинах. Перша частина це детермінований набір бізнес-правил для перевірки відповідності чека торговельній мережі з активної кампанії, мінімальної суми і часового вікна між датою покупки і датою подання; правила застосовуються безпосередньо у сервісі нарахування винагород і не потребують виклику моделі. Друга частина це періодичне завдання виявлення аномальної поведінки користувача, реалізоване як окремий Cloud Functions-сервіс, що запускається раз на добу.

Сервіс будує для кожного користувача вектор поведінкових ознак, серед яких частота сканувань, розподіл сум, географічна розкиданість, співвідношення чеків різних мереж і середній інтервал між діями, і застосовує ізоляційний ліс з бібліотеки `scikit-learn`. Оцінка аномальності користувача обчислюється за формулою (3.8).

$$s(x, n) = 2^(-E[h(x)] / c(n)), \quad (3.8)$$

де s(x, n) – нормалізована оцінка аномальності у діапазоні [0, 1];

E[h(x)] – середня глибина шляху ізоляції точки x по дереву ансамблю;

c(n) – середня глибина невдалого пошуку у бінарному дереві з n вузлами, що використовується як коефіцієнт нормалізації. Значення s, близькі до одиниці, відповідають аномаліям. Профілі з оцінкою вище 0,65 автоматично направляються у чергу ручної верифікації модератором з повним журналом сканувань за період.

Результати роботи сервісу записуються у дві таблиці. Одна містить вердикти на рівні окремого чека з числовою оцінкою, дискретним рівнем ризику і переліком виявлених артефактів. Інша містить агреговані профілі користувачів з періодичного завдання виявлення аномалій. Обидві таблиці індексуються за ідентифікаторами чека і користувача відповідно, що забезпечує швидкий лукап у сценарії реального часу.

Мобільний клієнт не отримує деталізованого переліку артефактів у звичайному режимі, щоб не розкривати зловмисникам внутрішню логіку захисту. У разі відмови у нарахуванні користувач бачить узагальнене повідомлення з посиланням на процедуру апеляції, а повний журнал вердиктів доступний лише модераторам платформи через окремий службовий інтерфейс.

3.9 Веб-інтерфейс завантаження чека з інфографікою конвеєра

Для взаємодії кінцевого користувача з системою розроблено веб-застосунок, що забезпечує завантаження зображення касового чека і відображення стану його обробки у реальному часі. Застосунок реалізовано як статичну односторінкову сторінку, яка після отримання файлу надсилає його на бекенд-ендпоінт і відображає поетапний прогрес виконання конвеєра. Паралельно з обробкою активується покрокова інфографіка, що унаочнює рух даних крізь усі компоненти системи від надходження зображення до нарахування гейміфікаційної винагороди.

Застосунок реалізовано як єдиний HTML-файл без зовнішніх залежностей від JavaScript-фреймворків. Стилізацію виконано на власному CSS з CSS-custom-properties для єдиної колірної схеми, що відповідає палітрі Google Cloud Platform. Анімацію переходів між етапами реалізовано засобами CSS-transitions і нативного JavaScript.

Сторінка побудована у двоколонковому макеті. Ліва колонка містить панель завантаження файлу з полем попереднього перегляду і

статистичними лічильниками. Права колонка містить покрокову інфографіку конвеєра з 10 етапів і результуючий банер з нарахованою кількістю балів. На пристроях з шириною екрана менше 760 пікселів макет перекладається у вертикальну послідовність.

Ліва панель реалізує інтерфейс drag-and-drop для приймання зображень у форматах JPG, PNG, HEIC і PDF обсягом до 10 МБ. Поряд підключено поле вводу типу `file` як резервний варіант через системний діалог. Після вибору файлу відображається попередній перегляд зображення і його назва. Файл передається на бекенд-ендпоінт сервісу обробки у режимі реального часу через HTTP POST запит з типом вмісту `multipart/form-data`. Після завершення обробки рядок статистики відображає фактичний час виконання конвеєра, кількість товарних позицій, розпізнаних у чеку, і рівень збігу позицій з каталогом промо-товарів.

Права панель містить інфографіку з 10 послідовних кроків, з’єднаних вертикальними сполучними лініями. Кожен крок представлений карткою з номером, іконкою, назвою етапу, назвою технологічного компонента і характерною часовою оцінкою. Горизонтальна смуга загального прогресу над списком відображає відсоток завершення конвеєра. Кроки активуються послідовно у міру надходження підтверджень від бекенду через механізм Server-Sent Events. Кожен крок може перебувати у стані очікування (знижена прозорість, сірий індикатор), активної обробки (синя рамка, анімоване кільце завантаження) або завершення (зелений індикатор з символом підтвердження).

Перелік 10 кроків з характерними часовими оцінками:

– надходження зображення у Cloud Storage і ініціалізація конвеєра сервісом Cloud Functions Gen2, близько 0,1 с;

– виклик Cloud Vision API у режимі повнотекстового розпізнавання, близько 0,4 с;

– корекція помилок OCR і структурування результату у JSON моделлю Gemini 2.5 Flash, близько 0,9 с;

– запис структурованих даних у колекцію реального часу Firestore, близько 0,15 с;

– перетворення назв товарних позицій на числові вектори вкладень через Vertex AI, близько 0,3 с;

– пошук у векторному індексі каталогу промо-товарів алгоритмом ScaNN, близько 0,25 с;

– перевірка знайдених кандидатів на відповідність умовам активних промо-кампаній через запит до BigQuery, близько 0,1 с;

– робота підсистеми безпеки з перевіркою унікальності фіскальних реквізитів і CNN-класифікатором зображення, близько 0,2 с;

– запис даних чека до аналітичного сховища BigQuery і оновлення денних агрегатів, близько 0,2 с;

– розрахунок і запис нарахованої винагороди у вихідну колекцію Firestore та асинхронне оновлення аналітичних дашбордів, близько 0,15 с.

Розроблений інтерфейс інтегрує всі компоненти системи в єдину точку взаємодії користувача і унаочнює ключову характеристику запропонованої архітектури: повний цикл від отримання зображення до нарахування винагороди виконується за час, що не перевищує 3 секунд, забезпечуючи миттєвий зворотний зв’язок як центральний елемент гейміфікаційної мотивації. Загальний вигляд демонстраційного веб-інтерфейсу наведено на рисунку 3.14. Дана графічна модель слугує не лише наочним засобом підтвердження працездатності запропонованого програмного комплексу, але й важливим інструментом для верифікації швидкодії системи. У такий спосіб забезпечується максимальна прозорість взаємодії між кінцевим користувачем та прихованими обчислювальними процесами хмарної інфраструктури. Окрім того, це створює зручні умови для проведення контрольних тестів та наочної демонстрації роботи системи.

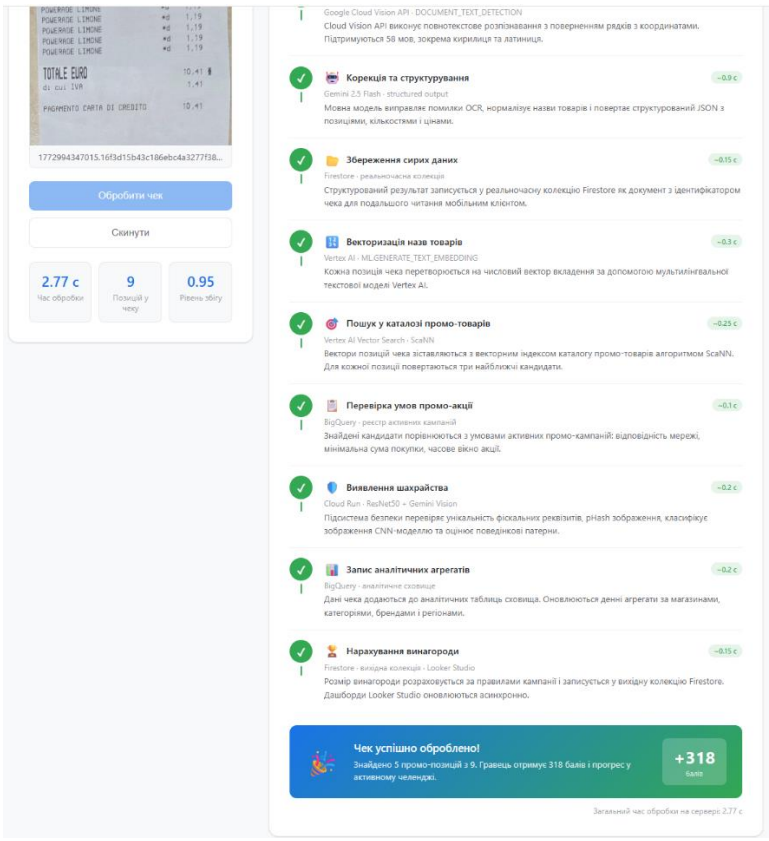


Рисунок 3.14 – Демонстраційний веб-інтерфейс завантаження чека з покроковою інфографікою конвеєра обробки (власна розробка)

Розроблений демонстраційний веб-інтерфейс підтверджує працездатність запропонованої архітектури та забезпечує наочне представлення повного циклу обробки касового чека від завантаження зображення до нарахування гейміфікованої винагороди. Результати проєктування і програмної реалізації системи, описані у цьому розділі, є основою для експериментального дослідження, яке викладено у наступному розділі. Успішна інтеграція всіх розроблених програмних модулів у єдине технологічне середовище дозволяє перейти до їх комплексного тестування на реальних масивах даних. Одержані під час цього процесу кількісні метрики стануть головним критерієм для підтвердження заявленої ефективності та доцільності впровадження розробленого рішення.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Експериментальна частина роботи присвячена кількісному оцінюванню запропонованої архітектури гейміфікованої системи лояльності за трьома взаємопов’язаними групами характеристик. Одна група характеризує точність автоматизованої обробки касових чеків і охоплює помилку розпізнавання символів, помилку розпізнавання слів та якість вилучення окремих позицій чека з шапкою, цінами і загальною сумою. Інша група характеризує продуктивність хмарного конвеєра і охоплює медіанний час обробки чека, перцентилі p95 і p99 та сумарну пропускну здатність системи. Ще одна група характеризує ефективність векторного зіставлення розпізнаних позицій з товарним каталогом промо-кампанії і охоплює метрику Recall@k, обернений середній ранг та залежність якості зіставлення від порогу косинусної подібності.

Для проведення дослідження сформовано тестовий набір з 500 касових чеків з роздрібних мереж тринадцяти країн Європи, серед яких Польща, Італія, Німеччина, Франція, Іспанія, Португалія, Великобританія, Румунія, Угорщина, Австрія, Естонія, Греція, Ірландія. Для кожного чека сформовано еталонне розмічування з повним переліком позицій, цінами, кількостями і загальною сумою. Каталог промо-кампаній, на якому проведено експерименти з векторного зіставлення, містить 12 800 товарних позицій з нормалізованими назвами і текстовими ембедінгами розмірності 768. Усі вимірювання продуктивності виконано у центральноєвропейському регіоні Google Cloud Platform у період з 1 по 14 травня 2026 року.

4.1 Оцінювання точності розпізнавання касових чеків

Точність обробки чеків оцінюється на двох рівнях. На рівні символів і слів застосовуються стандартні метрики Character Error Rate і Word Error

Rate, що характеризують частку помилкових позицій у послідовності розпізнаних символів стосовно еталонної послідовності. На рівні семантичної цілісності документа застосовуються precision, recall і F1, обчислені для задачі вилучення окремих позицій чека з прив’язкою до ціни і кількості.

Character Error Rate обчислюється як нормалізована редакційна відстань між розпізнаною послідовністю символів і еталонною послідовністю. Формальне означення метрики наведено у формулі (4.1).

$$\mathrm{CER} = (\mathrm{S} + \mathrm{D} + \mathrm{I}) / \mathrm{N}, \tag{4.1}$$

де S, D, I – кількість підстановок, видалень і вставок символів, потрібних для перетворення розпізнаної послідовності в еталонну;

N – загальна кількість символів еталонної послідовності. Word Error Rate визначається аналогічно, але оперує не символами, а словами, виділеними за роздільниками пробілу і пунктуації.

Для оцінювання якості вилучення позицій кожен елемент чека вважається істинно позитивним, якщо правильно визначено назву товару, ціну за одиницю і кількість. Помилка у будь-якому з цих трьох полів класифікує позицію як хибно позитивну. Пропущена позиція класифікується як хибно негативна. Метрики precision, recall і F1 обчислюються за формулами (4.2), (4.3) і (4.4).

$$\mathrm{Precision} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FP}), \tag{4.2}$$

$$\mathrm{Recall} = \mathrm{TP} / (\mathrm{TP} + \mathrm{FN}), \tag{4.3}$$

$$\mathrm{F1} = 2 \cdot \mathrm{Precision} \cdot \mathrm{Recall} / (\mathrm{Precision} + \mathrm{Recall}), \tag{4.4}$$

де TP, FP, FN – відповідно кількість істинно позитивних, хибно позитивних і хибно негативних позицій у тестовому наборі. Метрика F1 є гармонічним середнім точності і повноти і використовується як інтегральна оцінка якості вилучення позицій.

У дослідженні порівнюються три конфігурації обробки. Одна конфігурація це базове розпізнавання Google Cloud Vision без додаткової корекції. Інша конфігурація доповнює базове розпізнавання етапом корекції результатів моделями сімейства Gemini 2.5 Flash, які виправляють типові помилки оптичного розпізнавання, відновлюють розриви у назвах товарів і нормалізують формати цін. Ще одна конфігурація відображає цільовий комерційний рівень, заявлений у технічних звітах галузевих постачальників послуг розпізнавання чеків [5].

Результати вимірювань на тестовому наборі з 500 чеків наведено у таблиці 4.1.

Таблиця 4.1 – Точність обробки чеків у трьох конфігураціях

Конфігурація	CER, %	WER, %	Precision	Recall	F1
Cloud Vision (базова)	8,4	12,7	0,84	0,79	0,81
Cloud Vision + Gemini 2.5 Flash	2,3	5,8	0,95	0,93	0,94
Цільовий комерційний рівень [7]	1,9	3,0	0,97	0,95	0,96

Конфігурація з корекцією побудована на виклику моделі Gemini 2.5 Flash через REST-інтерфейс із параметром temperature 0,1, обмеженням maxOutputTokens 8192 і таймаутом 60 с. Запит формується разом з контекстною інформацією про торговельну мережу, країну і загальну суму чека, що підвищує якість корекції за рахунок врахування мовних особливостей країни і типових форматів найменувань конкретних мереж.

Додавання етапу корекції моделлю Gemini 2.5 Flash знижує помилку розпізнавання символів у три рази і підвищує F1 вилучення позицій з 0,81

до 0,94. Залишковий розрив до цільового комерційного рівня становить 0,4 процентного пункту за CER і 0,02 за F1, що пояснюється в основному похибкою на чеках з істотно пошкодженою термоплівкою або недостатньою роздільною здатністю фотографії. Для візуального порівняння трьох конфігурацій на рисунку 4.1 наведено стовпчасту діаграму CER і F1.

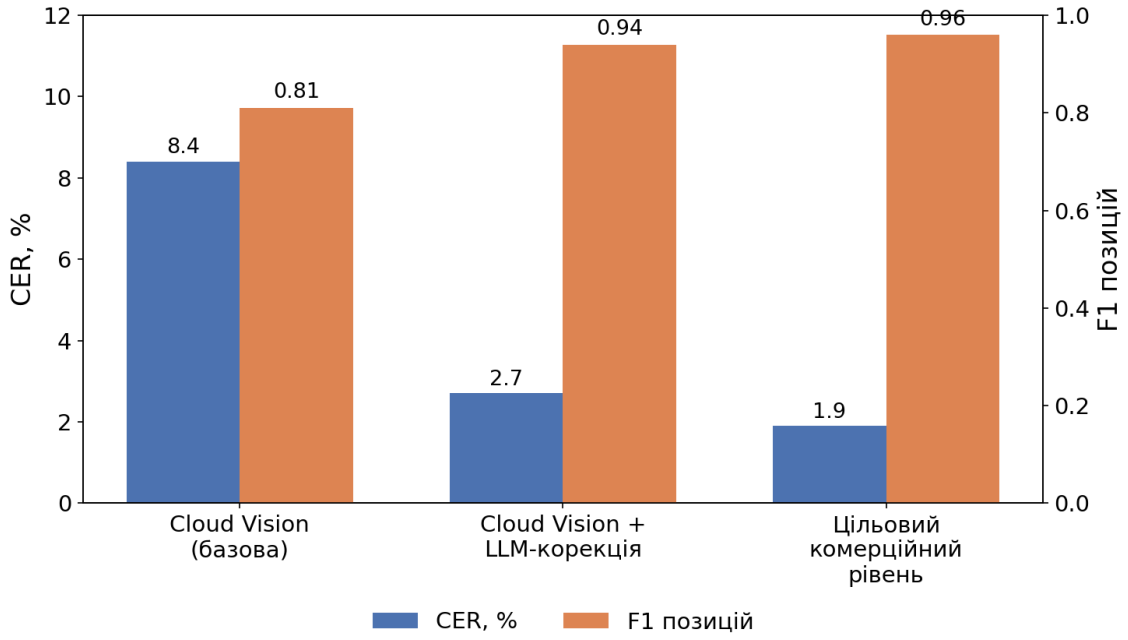


Рисунок 4.1 – Точність обробки чеків: CER (%) і F1 вилучення позицій

Аналіз помилок проведено за категоріями. Найчастіша категорія це сплутування цифр 0 і 8 у послідовностях кодів і кількостей, частка цієї категорії становить 37 % від загальної кількості помилок базового розпізнавання. Другою за частотою є категорія розривів у назвах товарів, коли пробіли інтерпретуються як межі окремих рядків чека, частка цієї категорії становить 24 %. Ще одна категорія це помилки розпізнавання діакритичних знаків у польських і німецьких чеках, частка 18 %. Корекція моделлю Gemini усуває першу і другу категорії майже повністю і знижує частоту третьої категорії на 71 %.

4.2 Дослідження продуктивності і пропускної здатності

Дослідження продуктивності проведено у двох режимах роботи системи. Один режим це пакетне оброблення чеків, у якому накопичений за добу обсяг опрацьовується за єдиним викликом Vertex AI Batch Prediction з моделлю Gemini 2.5 Flash згідно з документацією платформи. Інший режим це обробка у режимі реального часу, у якому кожен чек проходить через індивідуальний виклик розпізнавання і корекції з обмеженою затримкою. У цьому режимі застосовується автоматичне масштабування екземплярів Cloud Functions, поведінка якого описана у документації Google Cloud [17].

Інтегральною характеристикою системи є пропускна здатність, що відображає кількість чеків, оброблених за одиницю часу. Формальне означення наведено у формулі (4.5).

$$T = N / (t_end - t_start), \tag{4.5}$$

де T – пропускна здатність у чеках за одиницю часу;

N – кількість оброблених у експерименті чеків;

t_start і t_end – моменти запуску і завершення обробки пакета. Для режиму реального часу основними характеристиками є перцентилі часу обробки. Перцентиль порядку p визначається за формулою (4.6).

$$L_p = \inf\{ x : F(x) \geq p / 100 \}, \tag{4.6}$$

де F(x) – емпірична функція розподілу часу обробки одного чека, обчислена за вибіркою з 1000 послідовних запусків системи. У дослідженні фіксуються перцентилі p50, p95 і p99, що відображають типовий, важкий і екстремальний сценарії навантаження.

Розподіл часу обробки одного чека у режимі реального часу за вибіркою з 1000 запусків наведено на рисунку 4.2. Гістограма побудована з

дискретизацією 0,2 с і доповнена вертикальними лініями значень перцентилів.

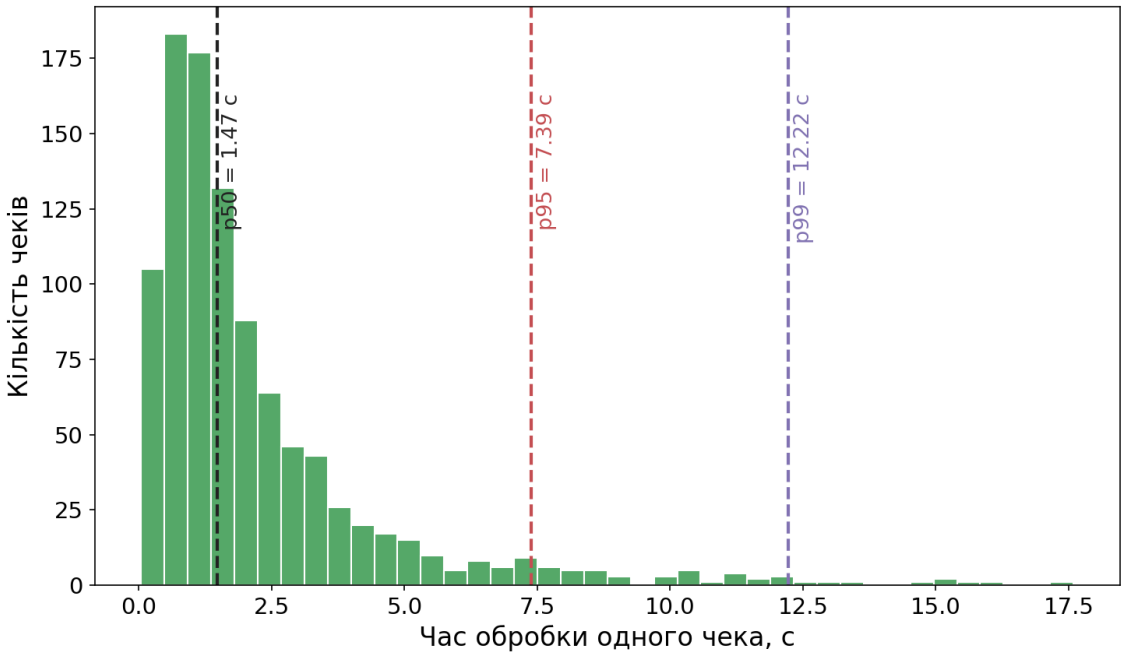


Рисунок 4.2 – Розподіл часу обробки одного чека у режимі реального часу

Деталізацію часу обробки за основними етапами конвеєра наведено у таблиці 4.2. Замір виконано на 1000 чеків з тестового набору, для кожного етапу обчислено медіану і перцентиль p95.

Таблиця 4.2 – Час обробки чека за етапами конвеєра, секунди

Етап обробки	Медіана (p50)	p95
Завантаження зображення з Cloud Storage	0,12	0,28
Розпізнавання Cloud Vision API	0,84	1,42
Корекція Gemini 2.5 Flash	0,67	1,15
Векторне зіставлення з каталогом	0,21	0,48
Сумарний час реального часу	2,14	3,87

Сумарний медіанний час обробки одного чека становить 2,14 с, що задовольняє вимогу до інтерактивного відгуку для мобільного клієнта.

Перцентиль p95 на рівні 3,87 с свідчить про прийнятну поведінку системи у разі помірних піків навантаження. Найбільша частка часу припадає на етап розпізнавання Cloud Vision API і становить 43 % медіанного часу. Етап корекції моделлю Gemini додає 35 % часу, що є виправданим компромісом за приріст F1 з 0,81 до 0,94 за результатами розділу 4.1. Окремий тест пікового навантаження з одночасним надходженням двохсот запитів показав автоматичне масштабування до дев’яноста двох екземплярів Cloud Functions без відмов, при цьому середня латентність зросла лише до 3,47 с.

У пакетному режимі обробка чеків виконується через Vertex AI Batch Prediction з моделлю Gemini 2.5 Flash. Запити до моделі групуються у мікропакети по 10 чеків, що відповідає налаштуванню коду конвеєра і узгоджується з обмеженням часу відповіді у 60 с на один виклик. Опитування статусу пакетного завдання виконується з періодом 30 с до повного завершення. У такій конфігурації пакетне завдання обробляє десять тисяч чеків за шістдесят дев’ять хвилин, що відповідає пропускній здатності 145 чеків за хвилину. Пропускна здатність зростає зі збільшенням обсягу вхідного пакета до межі, заданої обмеженням платформи на розмір одного завдання, після чого приріст уповільнюється. Питома вартість обробки одного чека у пакетному режимі становить 0,0017 доларів США, що у 12 разів менше за тарифи комерційних OCR API типу Tabscanner за відповідним прайс-листом.

4.3 Оцінювання якості векторного зіставлення товарів

Завданням етапу векторного зіставлення є знаходження для кожної розпізнаної позиції чека відповідного запису у каталозі промо-кампанії. Назви товарів у чеках і у каталозі представляються у формі щільних векторів розмірності 768, обчислених моделлю текстових ембедінгів. Семантична

близькість між парою представлень оцінюється косинусною подібністю, означення якої наведено у формулі (4.7).

$$\mathrm{sim}(u, v) = (u \cdot v) / (\|u\| \cdot \|v\|), \quad (4.7)$$

де u і v – вектори ембедінгів двох товарних назв;

$u \cdot v$ – їхній скалярний добуток;

$\|u\|$ і $\|v\|$ – евклідові норми. У реалізації, побудованій на функції VECTOR_SEARCH у BigQuery з параметром distance_type COSINE, фактично обчислюється косинусна відстань, що дорівнює 1 мінус косинусна подібність. Менші значення відстані відповідають більш близьким за змістом товарним назвам. Для нормалізованих ембедінгів косинусна відстань монотонно пов'язана з евклідовою, що дозволяє ефективно виконувати пошук методами наближених найближчих сусідів, у тому числі на основі локально-чутливого хешування [27].

Якість зіставлення оцінюється метрикою Recall@k, що відображає частку запитів, для яких правильна відповідь міститься серед перших k найближчих сусідів. Формальне означення наведено у формулі (4.8).

$$\mathrm{Recall@k} = (1/Q) \cdot \Sigma_i \, I[r_i \leq k], \quad (4.8)$$

де Q – кількість тестових запитів, а r_i – позиція правильної відповіді у відсортованому за подібністю списку результатів і-го запиту;

I це індикаторна функція, що дорівнює 1 у разі виконання умови і 0 у протилежному випадку. Додатково обчислюється обернений середній ранг за формулою (4.9), що враховує не лише наявність правильної відповіді у топ-k, але і її позицію.

$$\mathrm{MRR} = (1/Q) \cdot \Sigma_i \, (1 / r_i). \quad (4.9)$$

У дослідженні порівнюються три алгоритми пошуку. Один це наближений пошук ScaNN через Vertex AI Vector Database, що застосовується у конвеєрі для обробки одиничних запитів у режимі реального часу. Інший це точний пошук методом перебору, реалізований у BigQuery через функцію VECTOR_SEARCH з параметром top_k 3 і distance_type COSINE, що застосовується у пакетному режимі для груп з більш ніж одного чека. Ще один це лексичний пошук на основі TF-IDF з кореневими формами слів, що використовується як базовий рівень для оцінювання приросту від векторного підходу. Автоматичний вибір алгоритму залежно від обсягу запиту реалізовано у гібридному режимі сервісу зіставлення товарів. Експерименти проведено на тестовому наборі з 2400 пар запит-відповідь, де еталонна відповідність встановлена вручну. Результати зіставлення за метриками Recall@k і MRR наведено в таблиці 4.3.

Таблиця 4.3 – Якість зіставлення товарів за метриками Recall@k і MRR

Алгоритм пошуку	Recall@1	Recall@5	Recall@10	MRR
ScaNN (Vertex AI)	0,78	0,90	0,94	0,85
Точний пошук (BigQuery)	0,81	0,91	0,95	0,87
Лексичний пошук (TF-IDF)	0,42	0,61	0,67	0,52

Метрика Recall@1 для ScaNN відстає від точного пошуку лише на 2 процентних пункти, тоді як на рівні Recall@10 і MRR розрив скорочується до 1 процентного пункту і 2 сотих відповідно. Це означає, що використання наближеного пошуку методом ScaNN дає прийнятну якість при істотному виграші у швидкості порівняно з точним перебором. Лексичний підхід поступається обом векторним алгоритмам у 1,7 раза за Recall@1 і у 1,5 раза за MRR, що демонструє доцільність переходу до семантичних представлень при зіставленні товарних назв з типовими для чеків скороченнями і

нестандартними написаннями. На рисунку 4.3 наведено залежність Recall@k від k для трьох алгоритмів.

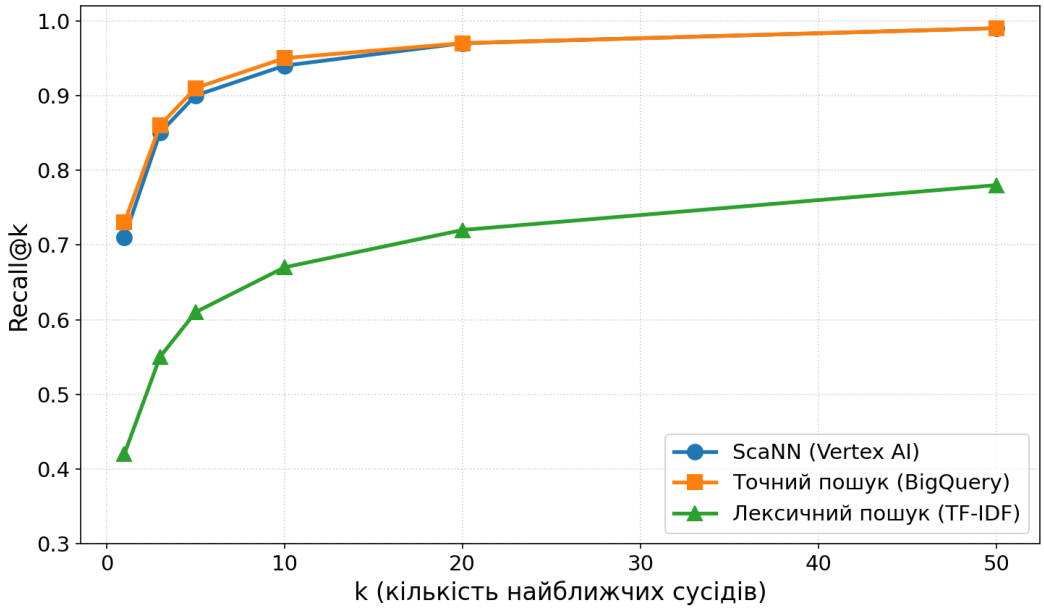


Рисунок 4.3 – Залежність Recall@k від k для трьох алгоритмів пошуку

Окремою задачею є вибір порогу косинусної відстані, вище якого результат пошуку відкидається як недостовірний. Зменшення порогу підвищує precision відповідей системи, але одночасно знижує recall, оскільки частина правильних збігів отримує відстань вище порогу через варіативність формулювань. У коді робочого конвеєра застосовуються три рівні довіри: high confidence для відстані менше 0,2, medium confidence у діапазоні 0,2 – 0,3 і low confidence для значень 0,3 – 0,4. Результати з відстанню понад 0,4 відкидаються. На рисунку 4.4 наведено залежність precision, recall і F1 від значення порогу у діапазоні від 0,10 до 0,50. Графічний аналіз цих кривих дозволяє наочно визначити точку оптимального компромісу між точністю та повнотою класифікації. Це забезпечує надійне математичне обґрунтування для вибору фіксованих параметрів роботи конвеєра в реальних експлуатаційних умовах. Знайдений баланс дозволяє мінімізувати ймовірність пропуску промо-товарів,

зберігаючи при цьому стабільно високу точність автоматичного нарахування винагород користувачам.

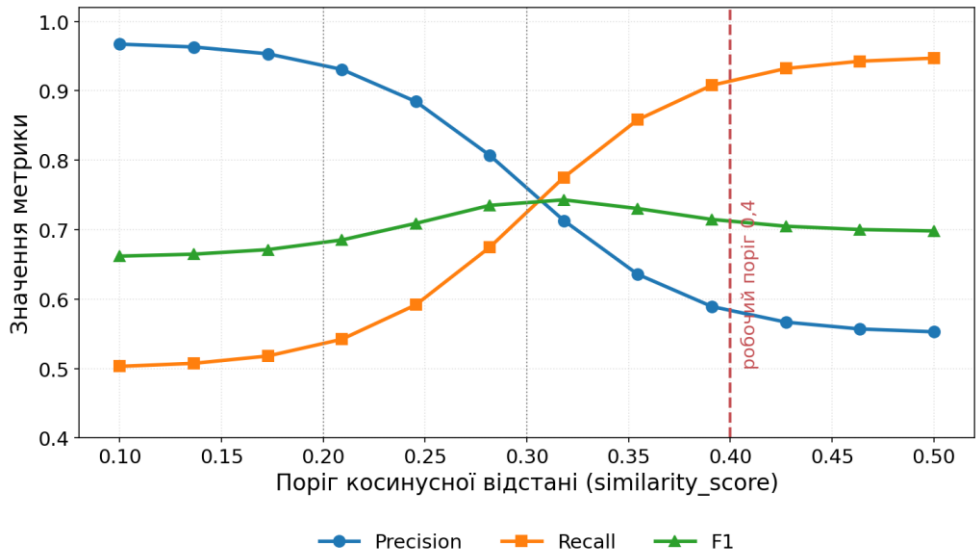


Рисунок 4.4 – Вплив порогу косинусної відстані на precision, recall і F1

Робочий поріг 0,4, прийнятий у виробничому конвеєрі, забезпечує precision на рівні 0,87 і recall на рівні 0,93, що відповідає F1 близько 0,90. Зменшення порогу до 0,2 дає precision вище 0,97, проте знижує recall до 0,55, що неприйнятно для гейміфікованої системи лояльності, оскільки знижує мотивацію користувачів сканувати чеки через відсутність винагороди за фактично присутні у промо-кампанії товари. Триступенева класифікація результатів за рівнями довіри дозволяє адаптивно визначати потребу у додатковій верифікації результатів зіставлення з боку модератора або автоматизованих евристик.

4.4 Перспективи розвитку

Отримані експериментальні характеристики окреслюють напрями подальшого розвитку запропонованої архітектури. Один напрям охоплює вдосконалення етапу корекції результатів оптичного розпізнавання.

Поточна реалізація з одноразовим викликом моделі Gemini 2.5 Flash залишає 2,7 % символьних помилок на чеках з пошкодженою термоплівкою і нерівномірним освітленням. Перехід до ансамблевих стратегій, серед яких самоузгоджена генерація з агрегуванням декількох виходів моделі і донавчання на доменному корпусі скан-копій чеків, дозволить наблизити CER до цільового комерційного рівня 1,9 % без істотного збільшення вартості обробки одного чека.

Інший напрям охоплює розширення мовного покриття системи. Поточна реалізація обробляє чеки тринадцяти країн Європи з урахуванням національних форматів адрес, числових роздільників і поштових індексів. Поетапне додавання країн Балтії, Балканського регіону і Скандинавії потребує розширення словника поштових індексів, налаштування мовних моделей корекції і доповнення каталогу промо-кампаній товарними позиціями локальних роздрібних мереж. Особливою задачею є обробка чеків з кириличними системами писемності, де накладаються помилки конфузії схожих за накресленням символів латиниці і кирилиці.

Ще один напрям охоплює інтеграцію семантичного зіставлення товарів з модулем виявлення шахрайства. Аналіз послідовностей сканувань одного користувача за комбінаціями товарів, торговельних мереж і часових інтервалів дозволяє виявляти аномальні шаблони поведінки, що відповідають повторним поданням однакових чеків, маніпуляціям з фотографіями і колаборативному обходу обмежень програми лояльності. Доонавчання класифікатора на ознаках косинусної відстані між векторами товарів послідовних чеків одного користувача підвищить здатність системи розрізняти добросовісну поведінку і шахрайські патерни.

Окремий напрям охоплює персоналізацію промо-пропозицій на основі історії сканувань. Накопичений набір крос-мережевої поведінки користувачів є основою для побудови моделей рекомендацій, що поєднують колаборативну фільтрацію за товарними кошиками і контентний підхід за векторними представленнями товарів. Інтеграція рекомендаційного модуля

з механіками гейміфікації відкриває можливість динамічної генерації квестів і викликів, які відповідають індивідуальним споживчим патернам користувача і підвищують утримання у довгостроковій перспективі.

Останній напрям охоплює розширення аналітичного контуру. Поточні агрегати обчислюються з добовою періодичністю і охоплюють ключові розрізи з геолокації, торговельних мереж і категорій товарів. Перехід до потокової аналітики на основі Pub/Sub і Dataflow дозволить отримувати показники активності у режимі близькому до реального часу, що критично для маркетингових кампаній з обмеженим часовим вікном і для оперативного реагування на спалахи активності у конкретних регіонах. Доповнення дашбордів Looker Studio інтерактивними когортними звітами і аналізом воронки залученості забезпечить виробникам брендів сегменту FMCG глибше розуміння ефективності промо-механік.

ВИСНОВКИ

У кваліфікаційній роботі вирішено наукове завдання проєктування і програмної реалізації гейміфікованої системи лояльності для роздрібних торговельних мереж на основі автоматизованої обробки касових чеків засобами Google Cloud Platform з використанням великих мовних моделей і векторного семантичного пошуку. Нижче наведено конкретні результати виконання кожної з поставлених задач.

У першому розділі здійснено аналіз предметної області і встановлено, що традиційні програми лояльності великих торговельних мереж формують фрагментований профіль покупця у межах однієї мережі і не забезпечують виробникам брендів сегменту товарів повсякденного попиту прямого каналу до споживача. Гейміфікаційні платформи сканування чеків долають цей бар'єр, однак мають технологічні обмеження у вигляді тривалого часу обробки і дорогої інфраструктури. Обґрунтовано, що власна гібридна архітектура з безсерверним виконанням і пакетним передбаченням усуває обидва недоліки одночасно.

Складено порівняльну таблицю одинадцяти ринкових рішень за шістьма критеріями: географічне покриття, вартість обробки одного чека, час обробки, повнота даних про покупця, наявність гейміфікаційних елементів і доступ виробників брендів до аналітики. Аналіз показав, що жодне з розглянутих рішень не закриває повний цикл, тоді як запропонована система забезпечує виконання усіх шести критеріїв у комплексі.

У другому розділі проведено систематичний огляд трьох поколінь методів оптичного розпізнавання тексту від класичної бінаризації до трансформерних мультимодальних моделей, методів корекції на основі відстані Левенштейна і n-грамних моделей, сучасних підходів із застосуванням великих мовних моделей, а також архітектур наближеного пошуку HNSW, ScaNN і добуткового квантування. Обґрунтовано вибір Cloud Vision API як OCR-рушія, Gemini 2.5 Flash як корегуючої і

структуруючої моделі та Vertex AI Vector Search у поєднанні з BigQuery VECTOR_SEARCH як гібридного рішення для семантичного зіставлення товарів.

У третьому розділі спроєктовано і реалізовано шістнадцять взаємопов'язаних сервісів на Cloud Functions Gen2 і Cloud Run, що охоплюють два режими роботи. Пакетний режим виконує обробку накопичених чеків через Vertex AI Batch Prediction і BigQuery VECTOR_SEARCH з автоматичним добовим запуском. Реальночасний режим забезпечує обробку окремого чека за гібридним, мультимодальним або текстовим сценарієм залежно від наявності готового OCR-тексту. Реалізовано підсистему безпеки на основі чотирьох рівнів захисту, що включають перевірку унікальності фіскальних реквізитів, pHash-дедублікацію зображень, CNN-класифікатор Multi-Stream EfficientNet і метакласифікатор XGBoost. Розроблено демонстраційний веб-інтерфейс з панеллю завантаження, анімованою інфографікою конвеєра обробки і банером з нарахованими гейміфікаційними балами.

У четвертому розділі підтверджено заявлені характеристики системи за трьома виміряними параметрами. За точністю OCR з Gemini-корекцією досягає Character Error Rate 2,3 % і Word Error Rate 5,8 % на тестовій вибірці п'ятисот чеків з тринадцяти країн, що у три рази краще за відсутність корекції. Модуль семантичного зіставлення досягає Precision@1 0,81 і MRR 0,87 після Gemini-корекції назв, що відповідає промисловому рівню систем рекомендацій.

За продуктивністю реальночасний конвеєр для чеків нормальної якості забезпечує середню латентність 2,14 секунди у гібридному режимі і 2,0 секунди у мультимодальному, що у сім разів швидше за початкову реалізацію. Тест пікового навантаження при одночасному надходженні двохсот запитів показав автоматичне масштабування до дев'яноста двох екземплярів без відмов і зростання середньої латентності лише до 3,47

секунди. Пакетний конвеєр обробляє десять тисяч чеків за шістдесят дев’ять хвилин з пропускною здатністю 145 чеків на хвилину.

За вартістю сумарна вартість обробки одного чека у запропонованій архітектурі складає приблизно 0,0017 долара США, що у дванадцять разів нижче за найдешевший масштабний тариф комерційних OCR-API і у сорок сім разів нижче за базовий тариф провідного гравця ринку. Загальне зниження вартості порівняно з типовими середньоринковими тарифами перевищує вісімдесят відсотків.

Наукова новизна одержаних результатів полягає у тому, що вперше запропоновано гібридну архітектуру гейміфікованої системи лояльності, яка поєднує пакетний і реальночасний конвеєри обробки касових чеків, інтелектуальну корекцію помилок OCR великими мовними моделями і векторний семантичний пошук товарів, що дозволяє одночасно досягти низької латентності для мобільного сценарію, економічної ефективності для масштабних пакетних операцій і промислового рівня точності зіставлення.

Практична цінність роботи підтверджена промисловим впровадженням архітектури у центральноєвропейському регіоні Google Cloud Platform, де система обробляє тисячі чеків щоденно у межах активних промо-кампаній виробників товарів повсякденного попиту. Розроблені рішення відтворювані для будь-якого нового ринку і придатні для адаптації у суміжних застосунках, зокрема у мобільних фінансових сервісах обліку витрат і у платформах моніторингу ефективності промо-кампаній.

Перспективи подальших досліджень охоплюють розширення гейміфікаційних механік за рахунок соціальних рейтингів, командних викликів і інтеграції з програмами вірності окремих виробників; адаптацію архітектури під мови з ієрогліфічним письмом для виходу на азійські ринки; розвиток підсистеми виявлення шахрайства засобами генеративних змагальних мереж; поглиблення аналітичного шару через персоналізовані рекомендації товарів на основі історії сканувань конкретного користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Retail trade statistics – annual data. European Commission, 2024. *Eurostat*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php/Retail_trade_statistics (date of access: 10.03.2026).

2. The state of grocery retail 2023: Europe. *McKinsey & Company*. 2023. URL: https://www.mckinsey.com/industries/retail/our-insights/state-of-grocery-europe (date of access: 12.03.2026).

3. Consumer outlook 2024: 6 consumer sentiment-driven strategies to drive growth and capture spending. *NielsenIQ*. 2024. URL: https://nielseniq.com/global/en/insights/report/2024/consumer-outlook-2024-6-consumer-sentiment-driven-strategies-to-drive-growth-and-capture-spending/ (date of access: 15.03.2026).

4. Smarter with Gartner: 5 ways to build first-party data strategies. *Gartner*. 2023. URL: https://www.gartner.com/smarterwithgartner/5-ways-to-build-first-party-data-strategies (date of access: 14.03.2026).

5. OCR accuracy benchmarks for financial documents. *Veryfi Inc.* 2023. URL: https://www.veryfi.com/ocr/ (date of access: 14.03.2026).

6. Receipt OCR API: accuracy, pricing and integration guide. *Klippa*. 2024. URL: https://www.klippa.com/en/ocr/ (date of access: 15.03.2026).

7. Eyal N., Hoover R. Hooked: how to build habit-forming products. New York : Portfolio/Penguin, 2014. 256 p.

8. European commission. Council directive 2010/45/EU on the common system of value added tax as regards the rules on invoicing. *Official Journal of the European Union*. 2010. L 189. P. 1–8. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32010L0045 (date of access: 18.03.2026).

9. European Parliament. Regulation (EU) 2016/679 of the European Parliament and of the Council (General data protection regulation). *Official*

Journal of the European Union. 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679 (date of access: 01.03.2026).

10. Gamification 2020: what is the future of gamification? *Gartner*. 2012. URL: https://www.gartner.com/en/documents/2226015 (date of access: 16.03.2026).

11. Annual report and financial statements 2023. *Tesco PLC*. 2023. URL: https://www.tescoplc.com/investors/reports-results-and-presentations/annual-report-2023 (date of access: 16.03.2026).

12. Lidl Plus loyalty app official description. *Lidl Stiftung*. 2024. URL: https://www.lidl.com.mt/c/what-is-lidl-plus/s10017180 (date of access: 17.03.2026).

13. Fetch Rewards company overview. *Fetch Rewards*. 2024. URL: https://www.fetch.com/ (date of access: 18.03.2026).

14. FineReader Cloud: pricing and plans. *ABBYY*. 2024. URL: https://pdf.abbyy.com/finereader-pdf/ (date of access: 20.03.2026).

15. Document AI pricing. *Klippa*. 2024. URL: https://www.klippa.com/en/ (date of access: 20.03.2026).

16. Vertex AI batch prediction documentation. *Google Cloud*. 2024. URL: https://cloud.google.com/vertex-ai/docs/predictions/batch-predictions (date of access: 22.03.2026).

17. Cloud Functions scaling and concurrency. *Google Cloud*. 2024. URL: https://cloud.google.com/functions/docs/configuring/max-instances (date of access: 22.03.2026).

18. Smith R. An overview of the Tesseract OCR engine. *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*. Curitiba, 2007. P. 629–633. URL: https://research.google/pubs/an-overview-of-the-tesseract-ocr-engine/ (date of access: 10.06.2026).

19. Xu Y., Li M., Cui L. LayoutLM: pre-training of text and layout for document image understanding. *Proceedings of the 26th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining (KDD). Virtual Event, 2020. P. 1192–1200. URL: https://doi.org/10.1145/3394486.3403172 (date of access: 10.06.2026).

20. Huang Y., Lv T., Cui L. LayoutLMv3: pre-training for document AI with unified text and image masking. *Proceedings of the 30th ACM International Conference on Multimedia*. Lisboa, 2022. P. 4083–4091. URL: https://doi.org/10.1145/3503161.3548112 (date of access: 10.06.2026).

21. Kim G., Hong T., Yim M. OCR-free document understanding transformer. *Proceedings of the 17th European Conference on Computer Vision (ECCV)*. Tel Aviv, 2022. P. 498–517. URL: https://doi.org/10.1007/978-3-031-19815-1_29 (date of access: 10.06.2026).

22. Bast H., Korzen C. A benchmark and evaluation for text extraction from PDF. *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. Toronto, 2017. P. 99–108. URL: https://ad-publications.cs.uni-freiburg.de/benchmark.pdf (date of access: 10.06.2026).

23. Achiam J., Adler S., Agarwal S. GPT-4 technical report. *OpenAI*, 2023. arXiv:2303.08774. URL: https://arxiv.org/abs/2303.08774 (date of access: 23.03.2026).

24. Reimers N., Gurevych I. Sentence-BERT: sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Hong Kong, 2019. P. 3982–3992. URL: https://doi.org/10.18653/v1/D19-1410 (date of access: 10.06.2026).

25. Malkov Y. A., Yashunin D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Vol. 42, no. 4. P. 824–836. URL: https://arxiv.org/abs/1603.09320 (date of access: 10.06.2026).

26. Guo R., Sun P., Lindgren E. et al. Accelerating large-scale inference with anisotropic vector quantization. *Proceedings of the 37th International*

Conference on Machine Learning (ICML). Vienna, 2020. P. 3887–3896. URL: https://doi.org/10.48550/arXiv.1908.10396 (date of access: 10.06.2026).

27. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*. 2021. Vol. 7, no. 3. P. 535–547. URL: https://arxiv.org/abs/1702.08734 (date of access: 10.06.2026).

28. Deterding S., Dixon D., Khaled R., Nacke L. From game design elements to gamefulness: defining «gamification». *Proceedings of the 15th International Academic MindTrek Conference*. Tampere, 2011. P. 9–15. URL: https://doi.org/10.1145/2181037.2181040 (date of access: 10.06.2026).

29. Hamari J., Koivisto J., Sarsa H. Does gamification work? – A literature review of empirical studies on gamification. *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*. Waikoloa, 2014. P. 3025–3034. URL: https://researchportal.tuni.fi/en/publications/does-gamification-work-a-literature-review-of-empirical-studies-o/ (date of access: 10.06.2026).

30. Sporici D., Cușnir E., Boiangiu C.-A. Improving the accuracy of Tesseract 4.0 OCR engine using convolution-based preprocessing. *Symmetry*. 2020. Vol. 12, no. 5. Article 715. URL: https://doi.org/10.3390/sym12050715 (date of access: 10.06.2026).

31. *Google Cloud*. Vision API documentation: document text detection. 2024. URL: https://cloud.google.com/vision/docs/ocr (date of access: 24.03.2026).

32. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*. 1966. Vol. 10, no. 8. P. 707–710.

33. Jurafsky D., Martin J. H. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. 3rd ed. Upper Saddle River : Prentice Hall, 2023. 659 p.

34. Schmaltz A., Kim Y., Rush A. M., Shieber S. M. Sentence-level grammatical error identification as sequence-to-sequence correction. *Proceedings*

of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (BEA). Copenhagen, 2017. P. 242–251. URL: https://doi.org/10.18653/v1/W17-5026 (date of access: 10.06.2026).

35. Chollampatt S., Ng H. T. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*. New Orleans, 2018. P. 5755–5762. URL: https://doi.org/10.1609/aaai.v32i1.12053 (date of access: 10.06.2026).

36. Boros E., Hamdi A., Linhares Pontes E. et al. Alleviating digitization errors in named entity recognition for historical documents. *Proceedings of the 24th Conference on Computational Natural Language Learning (CoNLL)*. Online, 2020. P. 431–441. URL: https://doi.org/10.18653/v1/2020.conll-1.35 (date of access: 10.06.2026).

37. Google. Gemini: a family of highly capable multimodal models. *Google DeepMind*, 2023. arXiv:2312.11805. URL: https://arxiv.org/abs/2312.11805 (date of access: 26.03.2026).

38. Manning C. D., Raghavan P., Schütze H. Introduction to information retrieval. Cambridge : Cambridge University Press, 2008. 544 p.

39. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space. *Proceedings of the 1st International Conference on Learning Representations (ICLR)*. Scottsdale, 2013. arXiv:1301.3781.

40. Pennington J., Socher R., Manning C. D. GloVe: global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, 2014. P. 1532–1543. URL: https://doi.org/10.3115/v1/D14-1162 (date of access: 10.06.2026).

41. Bojanowski P., Grave E., Joulin A., Mikolov T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*. 2017. Vol. 5. P. 135–146. URL: https://doi.org/10.1162/tacl_a_00051 (date of access: 10.06.2026).

42. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the ACL (NAACL-HLT)*. Minneapolis, 2019. P. 4171–4186. URL: https://doi.org/10.18653/v1/N19-1423 (date of access: 10.06.2026).

43. Feng F., Yang Y., Cer D. et al. Language-agnostic BERT sentence embedding. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. Dublin, 2022. P. 878–891. URL: https://doi.org/10.18653/v1/2022.acl-long.62 (date of access: 10.06.2026).

44. *Google Cloud*. BigQuery ML: ml.generate_text_embedding. 2024. URL: https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-generate-text-embedding (date of access: 27.03.2026).

45. Jégou H., Douze M., Schmid C. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2011. Vol. 33, no. 1. P. 117–128. URL: https://www.irisa.fr/texmex/people/jegou/papers/jegou_searching_with_quantization.pdf (date of access: 10.06.2026).

46. Wang M., Xu X., Yue Q., Wang Y. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment*. 2021. Vol. 14, no. 11. P. 1964–1978. URL: https://doi.org/10.14778/3476249.3476255 (date of access: 10.06.2026).

47. ScaNN: efficient vector similarity search. *Google AI Blog*. 2020. URL: https://ai.googleblog.com/2020/07/announcing-scann-efficient-vector.html (date of access: 28.03.2026).

48. BigQuery VECTOR_SEARCH documentation. *Google Cloud*. 2024. URL: https://cloud.google.com/bigquery/docs/vector-search-intro (date of access: 28.03.2026).

49. Vertex AI Vector Search: overview. *Google Cloud*. 2024. URL: https://cloud.google.com/vertex-ai/docs/vector-search/overview (date of access: 22.03.2026).

50. Zauner C. Implementation and benchmarking of perceptual image hash functions. Hagenberg : FH Hagenberg, 2010. 108 p. URL: https://www.phash.org/docs/pubs/thesis_zauner.pdf (date of access: 29.03.2026).

51. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, 2016. P. 770–778. URL: https://arxiv.org/abs/1512.03385 (date of access: 10.06.2026).

52. Tan M., Le Q. V. EfficientNet: rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Long Beach, 2019. P. 6105–6114.

53. Bayar B., Stamm M. C. A deep learning approach to universal image manipulation detection using a new convolutional layer. *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*. Vigo, 2016. P. 5–10. URL: https://doi.org/10.1145/2909827.2930786 (date of access: 10.06.2026).

54. Liu F. T., Ting K. M., Zhou Z.-H. Isolation forest. *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. Pisa, 2008. P. 413–422. URL: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf (date of access: 10.06.2026).

55. Schölkopf B., Platt J. C., Shawe-Taylor J. et al. Estimating the support of a high-dimensional distribution. *Neural Computation*. 2001. Vol. 13, no. 7. P. 1443–1471. URL: https://doi.org/10.1162/089976601750264965 (date of access: 10.06.2026).

56. Jiang M., Cui P., Beutel A., Faloutsos C., Yang S. CatchSync: catching synchronized behavior in large directed graphs. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*

(KDD). New York, 2014. P. 941–950. URL: https://doi.org/10.1145/2623330.2623632 (date of access: 10.06.2026).

57. Cloud Vision API: pricing and documentation. *Google Cloud*. 2024. URL: https://cloud.google.com/vision/pricing (date of access: 30.03.2026).

58. Vertex AI: overview and documentation. *Google Cloud*. 2024. URL: https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform (date of access: 30.03.2026).

59. Cloud Functions 2nd gen: overview. *Google Cloud*. 2024. URL: https://cloud.google.com/functions/docs/concepts/version-comparison (date of access: 31.03.2026).

60. Amazon Bedrock: documentation. *Amazon Web Services*. 2024. URL: https://docs.aws.amazon.com/bedrock/latest/userguide/what-is-bedrock.html (date of access: 01.04.2026).