Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ МУЛЬТИМОДАЛЬНОГО БОТА В TELEGRAM ДЛЯ ОЦИФРУВАННЯ ТА СИСТЕМАТИЗАЦІЇ РУКОПИСНИХ КУЛІНАРНИХ РЕЦЕПТІВ НА ОСНОВІ OCR ТА NLP
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Марюхна А. В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І. С.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Марюхні Анастасії Валеріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування Python, фреймворк Aiogram для створення ботів в Telegram, хмарний сервіс розпізнавання тексту Google Cloud Vision API, велика мовна модель Gemini 2.5, бібліотека комп'ютерного зору з відкритим кодом OpenCV, система управління базами даних PostgreSQL, бібліотека об'єктно-реляційного відображення даних SQLAlchemy, середовище розроблення PyCharm Professional, інструмент для створення діаграм PlantUML.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих ботів в Telegram і сучасних підходів для оцифрування та систематизації рукописних кулінарних рецептів.
2. Проєктування структури мультимодального бота в Telegram, бази даних та процесів оброблення кулінарних рецептів.
3. Вибір інструментальних засобів та обґрунтування технологічного стеку для розроблення бота в Telegram.
4. Програмна реалізація та тестування мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми оцифрування та систематизації кулінарного рукопису, постановка задачі, архітектурна схема компонентів системи, схема бази даних, діаграми варіантів використання, результати тестування, перспективи подальшої роботи, висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Проєктування структури та алгоритмів	23.04.26-07.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Творошенко І. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 128 с., 2 табл., 64 рис., 11 дод., 61 джерело.

ВЕЛИКА МОВНА МОДЕЛЬ, МУЛЬТИМОДАЛЬНИЙ БОТ, ОБРОБЛЕННЯ ПРИРОДНОЇ МОВИ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, ОЦИФРУВАННЯ КУЛІНАРНИХ РЕЦЕПТІВ, РУКОПИСНИЙ ТЕКСТ, PYTHON, TELEGRAM.

Об’єктом роботи є процес розроблення мультимодального бота в Telegram.

Метою роботи є розроблення та впровадження мультимодального бота в Telegram для автоматизації процесу оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP.

Під час роботи використано: Google Cloud Vision API та OpenCV для розпізнавання тексту, модель Gemini для структуризації та корекції кулінарних рецептів, Aiogram, PostgreSQL, Python.

Практична цінність полягає в автоматизації оцифрування рукописних рецептів та їх систематизації у персональну базу знань.

Розроблено бот, що розпізнає текст з фото, виконує структуризацію, пошук, корекцію та експорт у PDF кулінарних рецептів.

Область застосування: домашня кулінарія, збереження кулінарної спадщини у цифровому форматі.

LARGE LANGUAGE MODEL, MULTIMODAL BOT, NATURAL LANGUAGE PROCESSING, OPTICAL CHARACTER RECOGNITION, DIGITIZATION OF CULINARY RECIPES, HANDWRITTEN TEXT, PYTHON, TELEGRAM.

The objective of this work is the process of developing a multimodal bot in Telegram.

The goal of this work is to develop and implement a multimodal Telegram bot for automating the digitization and systematization of handwritten culinary recipes based on OCR and NLP.

The following were used during the work: Google Cloud Vision API and OpenCV for text recognition, Gemini model for structuring and correction of culinary recipes, Aiogram, PostgreSQL, Python.

The practical value lies in automating the digitization of handwritten recipes and their systematization into a personal knowledge base.

A bot has been developed providing text recognition from photos, structuring, search, correction, and PDF export of culinary recipes.

Applications: home cooking, preservation of culinary heritage in digital format.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

КБЖВ – калорії білки жири та вуглеводи

СУБД – система управління базами даних

ШІ – штучний інтелект

API – Application Programming Interface (прикладний програмний інтерфейс)

CER – Character Error Rate (коефіцієнт помилок на рівні символів)

CLAHE – Contrast Limited Adaptive Histogram Equalization (метод адаптивного вирінювання гістограми з обмеженням контрасту)

CNN – Convolutional Neural Networks (згорткова нейронна мережа)

CRUD – Create, Read, Update, Delete (4 основні функції управління даними «створення, читання, оновлення і вилучення»)

ER-діаграма – Entity-Relationship diagram (модель «сутність-зв’язок»)

F1-score – гармонійне середнє між точністю і повнотою класифікації

FSM – Finite State Machine (автомат зі скінченною множиною станів)

GIN – Generalized Inverted Index (узагальнений інвертований індекс)

ground truth – еталонні дані, що використовуються як зразок для оцінювання точності результатів автоматичного оброблення

HTML – HyperText Markup Language (мова розмітки гіпертексту)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

ID – Identifier (унікальний ідентифікатор)

iOS – iPhone Operating System (мобільна операційна система компанії Apple)

IP-адреса – Internet Protocol address (унікальний числовий ідентифікатор мережевого рівня)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

JSONB – JSON Binary (спеціальний тип даних у базах даних PostgreSQL, призначений для зберігання документів JSON у бінарному форматі)

LLM – Large Language Model (велика мовна модель)

long polling – механізм взаємодії з сервером, за якого клієнт надсилає запит і очікує відповідь до появи нових даних або завершення часу очікування

NLP – Natural Language Processing (обробка природної мови)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

ORM – Object-Relational Mapping (об’єктно-реляційне відображення)

PDF – Portable Document Format (портативний формат документів)

Precision – точність класифікації

Recall – повнота класифікації

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

SHA-256 – Secure Hash Algorithm 256-bi (256-бітний алгоритм безпечного хешування, що використовується для криптографічного захисту інформації)

SQL – Structured Query Language (структурована мова запитів для управління даними в реляційних базах даних)

SSL-сертифікат – Secure Sockets Layer (рівень захищених сокетів)

SVM – Support Vector Machines (метод опорних векторів)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

UUID – Universally Unique Identifier (універсальний унікальний ідентифікатор)

webhook – механізм зворотного виклику, за якого сервер самостійно надсилає дані клієнту через мережевий запит при настанні певної події

WER – Word Error Rate (коефіцієнт помилок на рівні слів)

кВ – кіловольт (одиниця вимірювання електричної напруги)

скл. – склянка (одиниця вимірювання об’єму)

ч.л. – чайна ложка (одиниця вимірювання об’єму)

°С – градус Цельсія (одиниця вимірювання температури)

% – відсоток (одиниця вимірювання відносних величин)

ВСТУП

У сучасному світі стрімка цифровізація повсякденного життя зумовлює необхідність перетворення аналогових даних у зручні електронні формати. Однією з найменш охоплених цифровою трансформацією сфер залишається побутове зберігання кулінарної інформації: мільйони унікальних сімейних рецептів існують лише на папері, що робить їх вразливими до фізичного зносу та незручними для повсякденного використання. Просте фотографування сторінок не вирішує цю проблему, оскільки зображення не підтримують пошук, фільтрацію чи редагування вмісту. Виникає потреба в інтелектуальному інструменті, здатному не лише зчитувати текст із фотографій, а й розуміти його семантичну структуру та перетворювати на функціональну цифрову базу знань.

Актуальність роботи полягає у відсутності комплексного рішення, яке б поєднувало оптичне розпізнавання символів (Optical Character Recognition, OCR) рукописного тексту, інтелектуальну структуризацію кулінарних даних та персоналізоване зберігання в єдиному доступному продукті. Технологічні передумови для створення такого інструменту вже сформовані: хмарні OCR-сервіси досягли рівня, достатнього для розпізнавання рукописного тексту різними мовами, а великі мовні моделі продемонстрували здатність до структуризації тексту довільного формату без необхідності створення жорстких правил для кожного типу документа. Проте, незважаючи на доступність окремих технологій, їх інтеграція в єдиний автоматизований конвеєр для кулінарної предметної області залишається нерозв’язаною задачею.

Проведений аналіз існуючих рішень підтвердив цю прогалину. Універсальні OCR-боти повертають неструктурований текст без розуміння предметної області, а спеціалізовані кулінарні боти працюють лише з готовими базами рецептів, не підтримуючи оцифрування власних записів користувача. Жоден із досліджених продуктів не забезпечує повного циклу від фотографії рукописного рецепту до структурованого запису з можливістю пошуку, фільтрації та інтелектуального редагування.

Темою даної кваліфікаційної роботи є розроблення мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі технологій оптичного розпізнавання символів та оброблення природної мови (Natural Language Processing, NLP). Для досягнення поставленої мети обрано архітектуру автоматичного конвеєра оброблення: попереднє оброблення зображення, розпізнавання тексту через хмарний сервіс, семантичний контроль, структуризація засобами великої мовної моделі та збереження в реляційній базі даних з підтримкою повнотекстового та нечіткого пошуку. Додатково реалізовано механізм інтелектуальної корекції рецептів через текстові інструкції природною мовою та експорт у формат PDF.

Результати роботи можуть бути використані для збереження сімейної кулінарної спадщини у цифровому форматі, систематизації персональних кулінарних архівів, а також як основа для розширення у напрямку рекомендаційних систем та колективних кулінарних баз.

1 АНАЛІЗ ІСНУЮЧИХ МУЛЬТИМОДАЛЬНИХ БОТІВ В TELEGRAM ДЛЯ ОЦИФРУВАННЯ ТА СИСТЕМАТИЗАЦІЇ РУКОПИСНИХ ТЕКСТІВ

1.1 Аналіз сучасних мультимодальних ботів в Telegram для оцифрування та систематизації рукописних текстів

Процес цифрової трансформації охоплює дедалі більше сфер людської життєдіяльності, включаючи побутові аспекти зберігання інформації. Сучасні користувачі прагнуть перенести паперові архіви, зокрема, сімейні кулінарні книги, у зручний цифровий формат. Месенджери, такі як Telegram, стали ідеальним середовищем для розгортання інтелектуальних інструментів, завдяки своїй доступності та потужному API. Використання ботів дозволяє автоматизувати рутинні завдання з оброблення зображень та тексту без встановлення додаткового програмного забезпечення.

Розроблення мультимодальних рішень для оцифрування рукописів є актуальним науковим та практичним завданням.

Мультимодальність у контексті чат-ботів передбачає здатність системи сприймати та обробляти різні типи вхідних даних одночасно. Особливо цікавим та актуальним є застосування цього підходу до оброблення таких побутових рукописних записів, як сімейні кулінарні рецепти. Користувач може надсилати фотографії сторінок або текстові уточнення до рецептів. Система повинна інтегрувати ці дані в єдину логічну структуру для подальшого пошуку та аналізу. Сучасні алгоритми машинного навчання дозволяють ефективно поєднувати візуальні та лінгвістичні ознаки об’єктів. Такий підхід значно підвищує точність розпізнавання складних рукописних текстів у порівнянні з класичними методами. Ринок існуючих ботів у Telegram пропонує чимало інструментів для простого розпізнавання тексту із друкованих документів.

Проте більшість із них демонструють низьку ефективність при роботі з індивідуальними почерками та нестандартним плануванням сторінок.

Рукописні рецепти часто містять скорочення, виправлення та специфічні кулінарні терміни, що ускладнює автоматичне оброблення. Спеціалізовані боти для кулінарії, зазвичай, обмежені лише базою готових рецептів без функцій оцифрування власного архіву. Відсутність комплексних рішень визначає необхідність у створенні нового, більш досконалого продукту на базі сучасних технологій.

Технологія оптичного розпізнавання символів пройшла довгий шлях від простих шаблонів до глибоких нейронних мереж. Сьогодні лідерами ринку є хмарні сервіси від Google, Microsoft та Amazon, які надають API для інтеграції у сторонні застосунки. Ці системи здатні розпізнавати десятки мов, включаючи підтримку кириличних шрифтів та рукописного введення. Завдяки моделі оплати за кількість запитів, використання таких API є економічно доцільним для проєктів із помірним навантаженням, а вбудовані алгоритми попереднього оброблення зображень на стороні сервісу зменшують необхідність у складній клієнтській підготовці даних.

Для успішної реалізації мультимодального бота необхідно враховувати особливості мобільної фотографії сторінок рецептів. Користувачі часто роблять знімки при поганому освітленні або під кутом, що створює геометричні спотворення тексту. Програмна частина бота повинна включати модулі для попередньої корекції зображень, видалення шумів та вирівнювання рядків. Це дозволяє суттєво зменшити кількість помилок на етапі роботи основного OCR-двигуна. Використання бібліотек на кшталт OpenCV є стандартом для вирішення подібних завдань у сучасному розробленні [1, 2].

Якість розпізнавання рукописного тексту залежить від обраного OCR-сервісу та його підтримки конкретної мови. Хмарні рішення, зокрема Google Cloud Vision, забезпечують підтримку рукописного введення для широкого спектру мов, включаючи українську мову, без необхідності навчання власних моделей [3].

Для об’єктивної оцінки точності розпізнавання використовується тестова вибірка рукописних рецептів з обчисленням коефіцієнта на рівні символів

(Character Error Rate, CER) та коефіцієнта на рівні слів (Word Error Rate, WER), що дозволяє кількісно визначити рівень помилок та обсяг необхідної ручної корекції.

Сучасний підхід до побудови інтелектуальних ботів передбачає делегування обчислювально складних задач зовнішнім хмарним сервісам. Це дає змогу зменшити навантаження на власну серверну інфраструктуру та забезпечити високу швидкість відгуку системи. Зокрема, хмарні API для оптичного розпізнавання та великі мовні моделі для структуризації тексту надають готові інструменти без необхідності розгортання та навчання власних моделей. Це дозволяє системі адаптуватися під конкретний домен знань через налаштування текстових інструкцій до моделі, а не через перенавчання нейронних мереж [4].

Аналіз функціональних можливостей ботів-конкурентів показує, що більшість із них не мають розвиненої системи систематизації. Після розпізнавання тексту користувач отримує «сиру» стрічку символів, яку важко використовувати для пошуку інгредієнтів. Справжній мультимодальний помічник має не лише зчитувати літери, а й розуміти структуру страви, що передбачає виділення назви, списку продуктів, часу приготування та покрокової інструкції. Реалізація такої структуризації вимагає застосування методів оброблення природної мови, зокрема сучасних великих мовних моделей (Large Language Model, LLM), здатних інтерпретувати контекст та перетворювати неструктурований текст у задану схему даних.

Використання великих мовних моделей дозволяє перетворити неструктурований текст у формат, придатний для зберігання у базі даних. На відміну від класичних NLP-бібліотек (SpaCy, NLTK), які потребують окремого налаштування правил для кожного типу сутностей, LLM здатні виконувати структуризацію тексту через єдину текстову інструкцію – розпізнавати назви продуктів, міри ваги, термічні операції та покрокові інструкції, враховуючи контекст рецепту. Це дає змогу реалізувати автоматичну категоризацію та пошук рецептів за інгредієнтами.

Такий рівень інтелектуального оброблення робить застосунок набагато кориснішим для повсякденного використання [5, 6].

Окрім технічних аспектів, важливо враховувати психологічний комфорт користувачів при переході на цифрові технології. Багато людей зберігають старі зошити як сімейну реліквію, тому бот повинен поважати цей емоційний зв’язок. Можливість бачити оригінальне фото поруч із розпізнаним текстом створює відчуття автентичності. Користувач бачить, що жодна деталь його сімейного спадку не була втрачена чи спотворена.

Для створення якісного бота в Telegram важливо ознайомитися з різними існуючими аналогами та можливостями платформ, що надаються для розробників для інтеграції з різними сервісами. Це дозволяє визначити ключові аспекти, які варто врахувати під час розроблення, виявити сильні та слабкі сторони аналогів і сформувати унікальні переваги власного продукту.

У процесі аналізу не було знайдено подібних ботів, які б спеціалізувались на комплексному завданні оцифрування та систематизації рукописних кулінарних рецептів. З огляду на специфіку теми роботи, для порівняльного аналізу було обрано дві групи ботів у Telegram. До першої групи належать боти, що реалізують функції оптичного розпізнавання тексту та забезпечують перетворення зображень у текстовий формат. До другої групи належать кулінарні боти, які надають користувачам рецепти, рекомендації або можливість пошуку страв за заданими параметрами.

Першим об’єктом аналізу у групі OCR-рішень став бот OCRBot [image to text] @ocrbot (Unofficial Telegram OCR Robot) [7].

Даний сервіс є інструментом для швидкої конвертації зображень у текст. Відсутність розгорнутої документації розробника вказує на використання стандартних бібліотек для оптичного розпізнавання з відкритим вихідним кодом.

Бот орієнтований на максимально спрощену взаємодію: користувач надсилає зображення і миттєво отримує текстову відповідь без додаткових налаштувань (рис. 1.1, рис. 1.2).

Рисунок 1.1 – Початок роботи з ботом OCRBot [image to text]

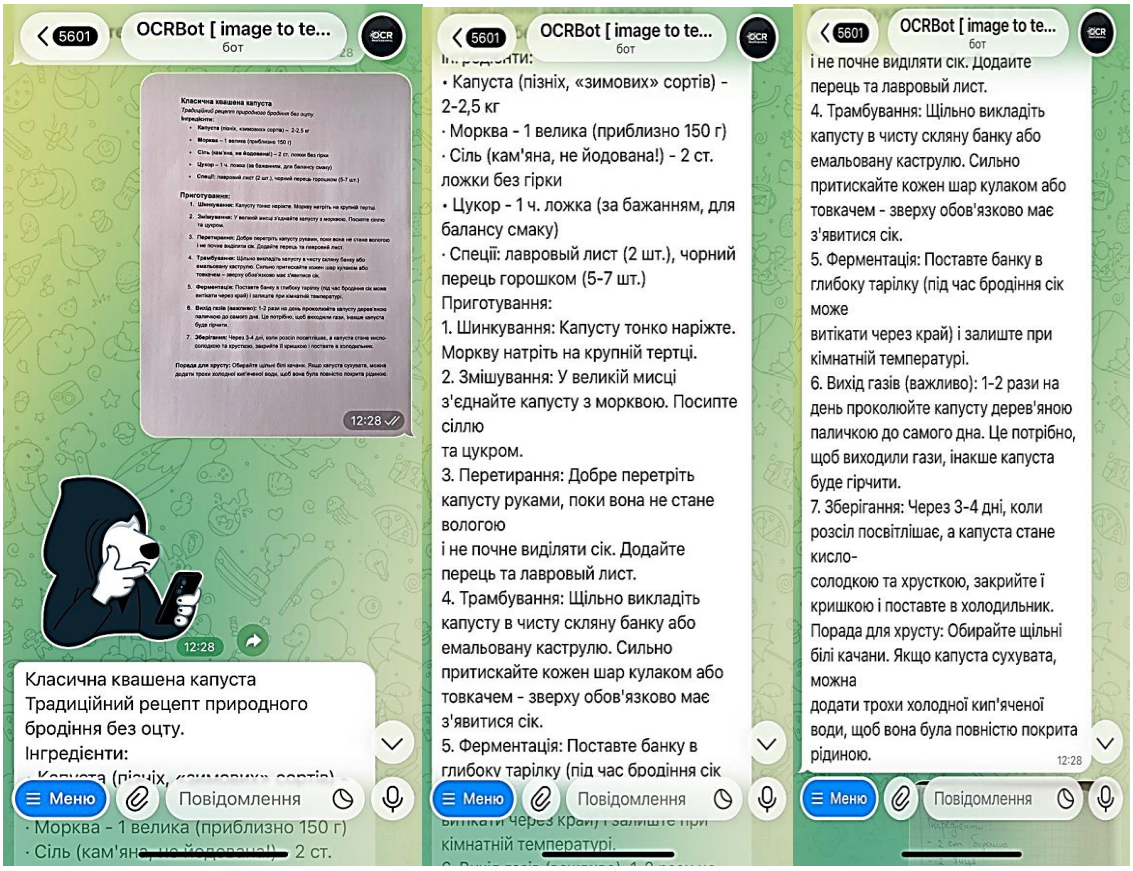


Рисунок 1.2 – Результат розпізнавання друкованого кулінарного тексту з фото за допомогою бота OCRBot [image to text]

Попереднє тестування на прикладі друкованого рецепта «Класична квашена капуста» показало, що бот демонструє високу ефективність при роботі з типографськими шрифтами. Система безпомилково відтворила структуру інгредієнтів, назву страви та покрокову інструкцію, зберігши пунктуацію та числові значення, що є характерним для двигунів типу Tesseract при роботі з чіткими контрастними зображеннями.

Проте під час практичного дослідження можливостей бота на прикладі рукописного рецепта «Кекси на молоці» (рис. 1.3) виявлено ряд критичних технічних недоліків, характерних для універсальних OCR-рішень.

Бот продемонстрував низьку точність розпізнавання індивідуального почерку, що призвело до високого рівня помилок у словах.

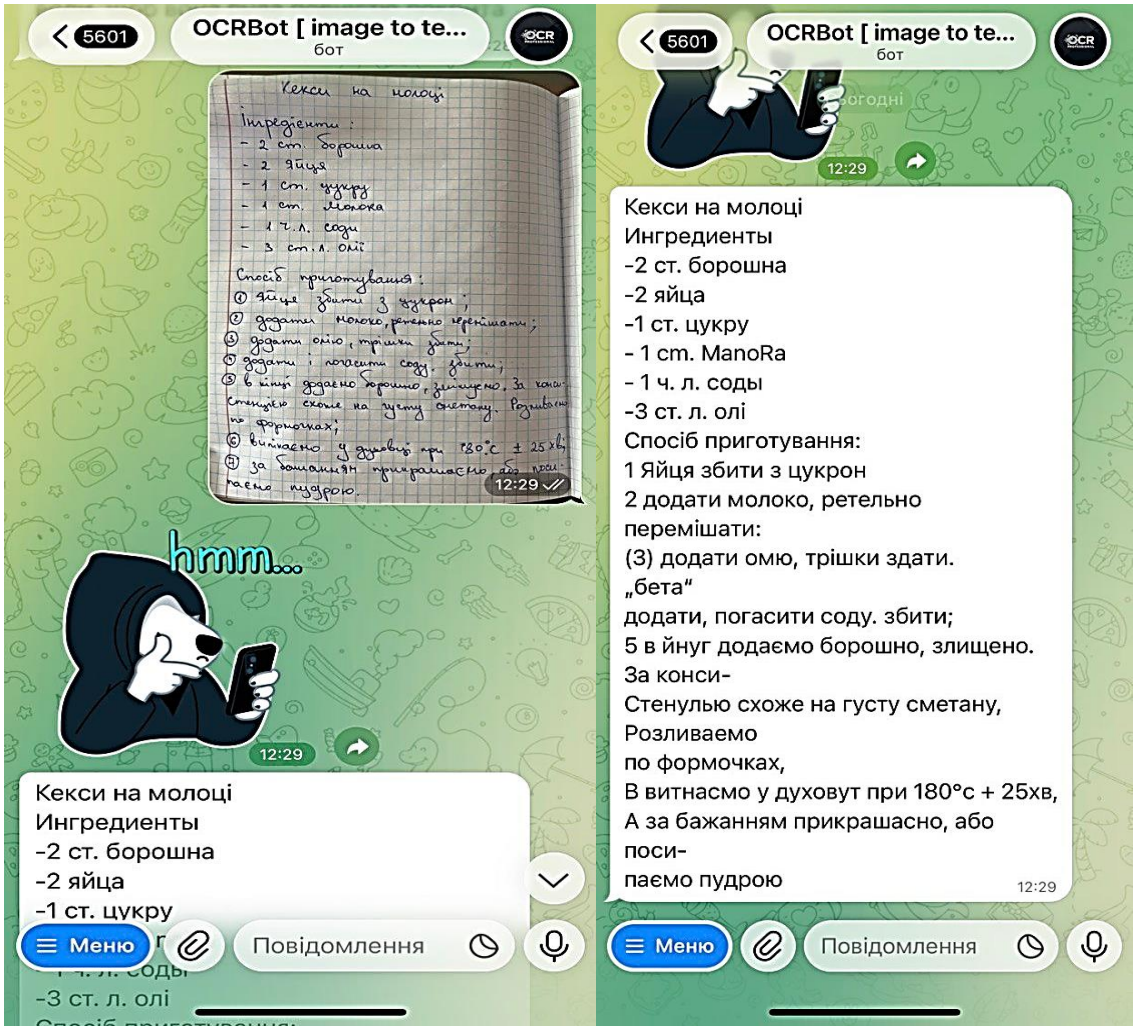


Рисунок 1.3 – Результат розпізнавання рукописного кулінарного тексту з фото за допомогою бота OCRBot [image to text]

Зокрема, специфічні кулінарні терміни та назви продуктів часто інтерпретувалися некоректно: наприклад, словосполучення «1 скл. молока» трансформовано у незрозумілу послідовність «1 cm. ManoRa», а назви інгредієнтів отримали помилкові фонетичні відповідники. Окрім того, система виявила нестабільність у визначенні мови тексту, замінивши український заголовок «Інгредієнти» на російськомовний аналог, що вказує на відсутність адаптованої лінгвістичної моделі для рукописної української лексики.

Окрему проблему становить втрата структури документа та некоректне зчитування числових значень у рукописному форматі. Бот не зміг правильно розпізнати кроки приготування та параметри температурного режиму, що робить отриманий текст непридатним для практичного використання без повної ручної корекції користувачем. Отриманий результат є типовою «сирою» стрічкою символів, де відсутній семантичний розподіл на логічні блоки, такі як назва страви, перелік продуктів та покрокова інструкція.

Другим об’єктом аналізу у групі інструментів оптичного розпізнавання став бот Argus (@argus_ocr_bot) [8]. Даний сервіс позиціонується як комплексний помічник для роботи з інформацією, що використовує передові архітектури нейромереж та алгоритми штучного інтелекту для вирішення широкого спектра завдань.

Технологічний стек сервісу дозволяє не лише виконувати високоточне розпізнавання друкованого та рукописного тексту, а й надає користувачу розширений набір інструментів післяетапного оброблення. Зокрема, завдяки інтеграції з нейромережами, бот підтримує миттєвий переклад вилученого тексту на різні мови та автоматичну корекцію орфографічних помилок. Важливою перевагою Argus є підтримка оброблення складних запитів та можливість структурування даних, наприклад, вилучення інформації з таблиць або вирішення математичних рівнянь безпосередньо із зображення.

Бот не потребує окремої реєстрації, забезпечуючи високу швидкість оброблення даних через зручний інтерфейс Telegram (рис. 1.4, рис. 1.5).

Рисунок 1.4 – Початок роботи з ботом Argus

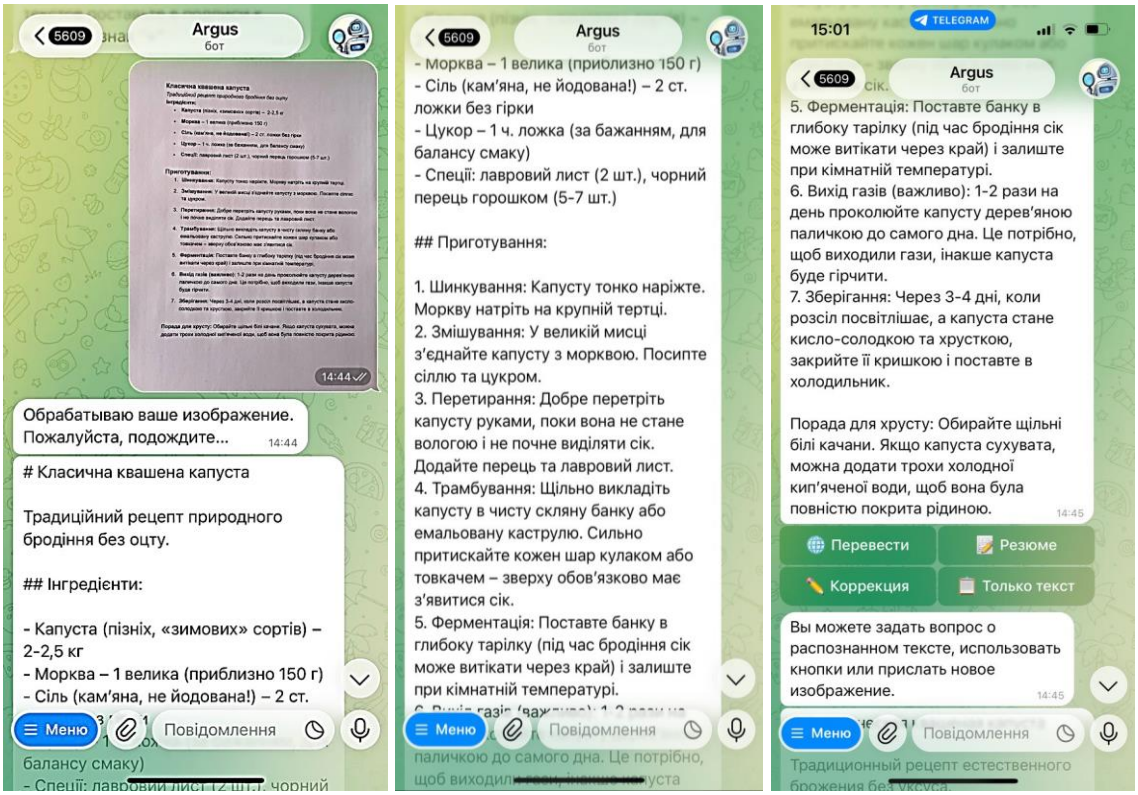


Рисунок 1.5 – Результат розпізнавання друкованого кулінарного тексту з фото за допомогою бота Argus

Окремої уваги заслуговує інтелектуальний модуль корекції тексту, інтегрований у функціонал боту Argus. Під час проведення експерименту з рукописним рецептом «Кекси на молоці» було встановлено, що на етапі первинного розпізнавання нейромережева система все ж припускається типових для OCR-моделей помилок. Зокрема, бот некоректно інтерпретував деякі скорочення та символи: замість «1 ч.л. соди» було розпізнано «1 т.л. соди», а технічні параметри температурного режиму випікання трансформувалися у помилкові «80°C ± 25 кВ» (рис. 1.6).

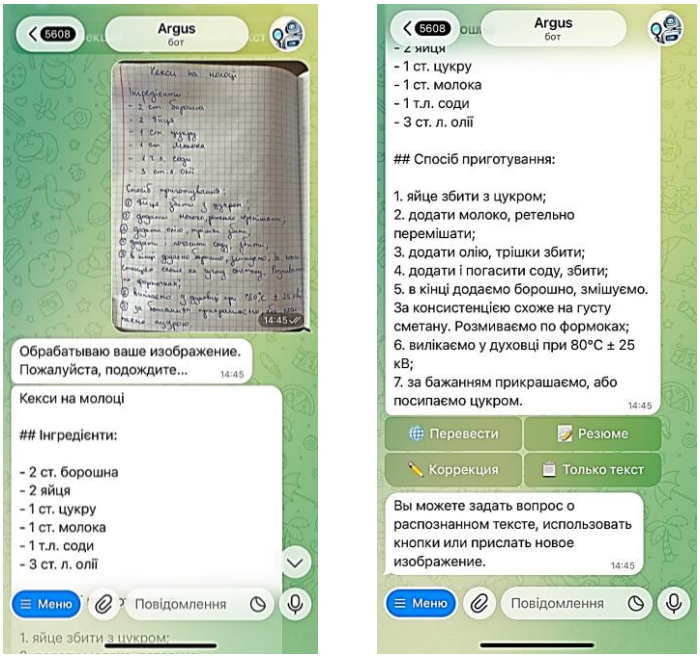


Рисунок 1.6 – Результат розпізнавання рукописного кулінарного тексту з фото за допомогою бота Argus

Однак застосування вбудованої функції «Корекція» дозволило системі задіяти потужності великої мовної моделі для суттєвого виправлення та покращення результату. Завдяки розумінню кулінарного контексту, алгоритм автоматично усунув технічні неточності.

Бот відновив логічне значення міри ваги «1 ч.л. соди», виправив опис кулінарного процесу з помилкового «розмиваємо» на «розмістити у формочках», а також адаптував температурний режим до реальних показників «180°C» замість раніше розпізнаних «80°C» (рис. 1.7).

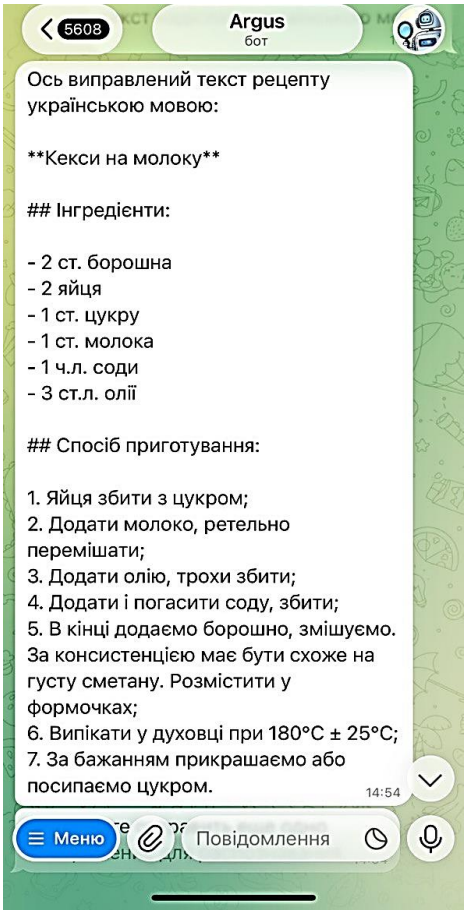


Рисунок 1.7 – Результат автоматичної корекції розпізнаного рукописного тексту за допомогою бота Argus

Такий результат демонструє значну перевагу використання сучасних нейромережевих підходів перед класичними OCR-бібліотеками, оскільки система здатна самостійно виправляти помилки на основі доменної логіки.

До другої групи аналізу належать спеціалізовані кулінарні асистенти, які демонструють підходи до систематизації даних та побудови зручної взаємодії з користувачем у межах предметної області. Першим об’єктом у цій категорії обрано бот Кулінарні рецепти (@tasty_cooking_bot) [9], який функціонує як розгалужений інтерактивний каталог готових рецептів.

Технічна архітектура даного сервісу базується на використанні розвиненої системи навігації, що забезпечує зручний розподіл контенту за тематичними категоріями. Такий функціонал реалізовано за допомогою вбудованих кнопок, які дозволяють користувачу ефективно взаємодіяти з інтерфейсом у межах одного вікна повідомлення (рис. 1.8).

Рисунок 1.8 – Інтерфейс вибору категорій страв у боті «Кулінарні рецепти»

Використання інтуїтивно зрозумілого меню забезпечує високу якість користувацького досвіду, оскільки дозволяє швидко отримувати доступ до необхідної інформації без необхідності вивчення складних текстових команд. Крім того, бот демонструє якісні підходи до візуалізації даних: рецепти подаються у структурованому вигляді з активним використанням спеціального форматування тексту та фотографією готового рецепту.

Це значно полегшує сприйняття інформації з екрана мобільного пристрою, що є критично важливим для застосунків, які використовуються безпосередньо в процесі приготування страв (рис. 1.9).

Однак, попри зразкову реалізацію інтерфейсу, аналіз боту «Кулінарні рецепти» показав, що він є «закритою» системою, орієнтованою виключно на споживання готового контенту. Користувач не має можливості додавати власні рецепти, редагувати наявні записи або створювати персональні архіви. Головним обмеженням у контексті нашого дослідження є повна відсутність мультимодальності: бот не підтримує завантаження фотографій для оцифрування чи редагування текстом.

Рисунок 1.9 – Візуалізація структурованого рецепта з фотографією готової страви у боті «Кулінарні рецепти»

Тому, сервіс є прикладом ефективної організації бази даних для читання, але він не вирішує задачу автоматизації введення нових даних із фізичних носіїв.

Далі було розглянуто бот AI Chef Bot (@Best_AI_Chef_Bot) [10], який є прикладом реалізації багатофункціонального помічника на основі технологій штучного інтелекту та методів оброблення природної мови.

Використання сучасних алгоритмів ШІ дозволяє сервісу підтримувати кілька інтелектуальних режимів взаємодії, що значно розширює його можливості порівняно зі звичайними статичними каталогами. Зокрема, завдяки інтеграції з нейромережами, режим «Рецепти» дозволяє не просто шукати, а генерувати унікальні кулінарні приписи на основі текстового переліку інгредієнтів. Режим «Калькулятор калорій» забезпечує інтелектуальний розрахунок персональних цілей споживання, а функція «Макроси інгредієнтів» надає детальну нутрієнтну інформацію про конкретні продукти, спираючись на великі бази даних харчової цінності (рис. 1.10 – рис. 1.12).

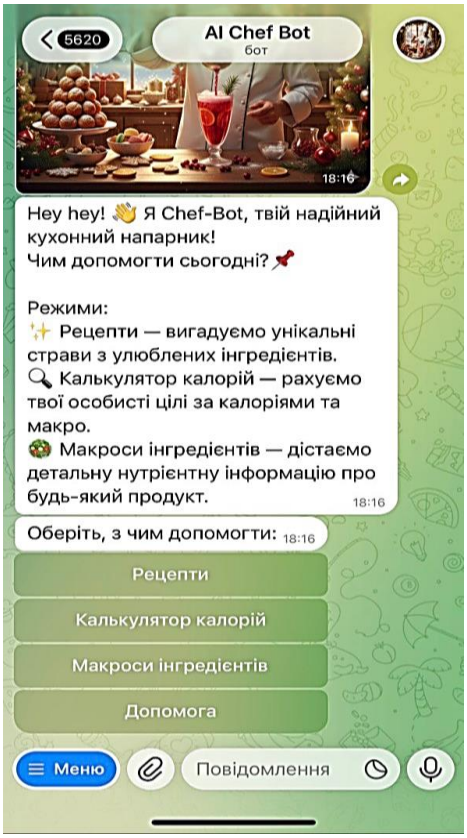


Рисунок 1.10 – Головне меню бота AI Chef Bot
із вибором інтелектуальних режимів роботи

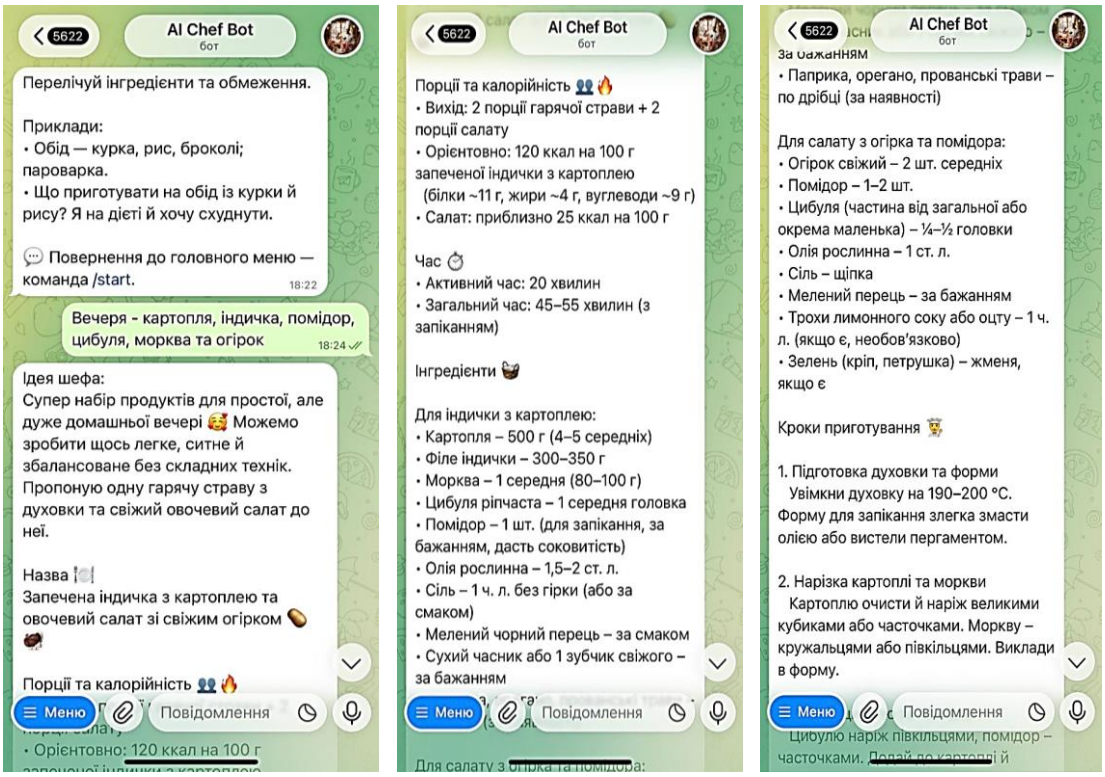


Рисунок 1.11 – Результат генерації рецепта за вказаним списком інгредієнтів
у боті AI Chef Bot

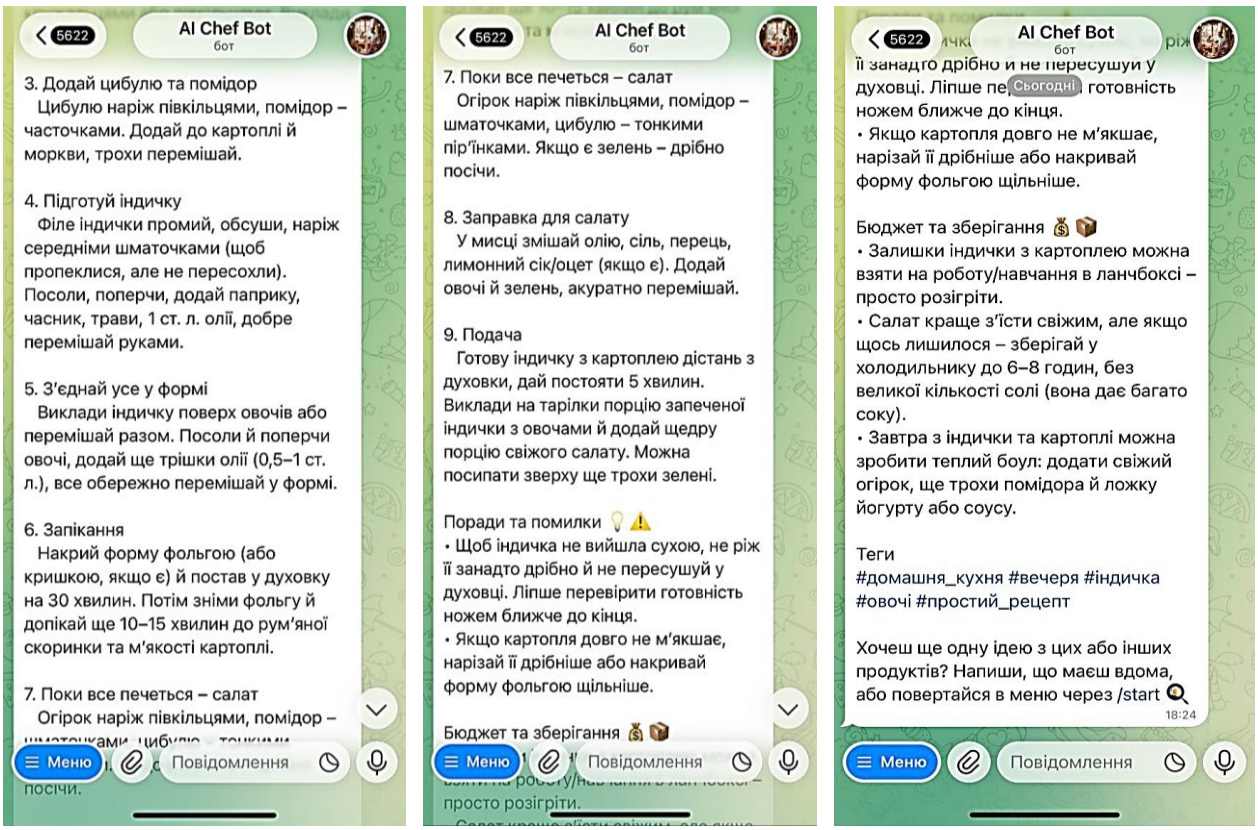


Рисунок 1.12 – Результат генерації рецепта за вказаним списком інгредієнтів (кроки приготування, поради та помилки, бюджет та зберігання згенерованого рецепту)

Головною перевагою AI Chef Bot є здатність ШІ-моделі структурувати фінальні відповіді, формуючи логічну послідовність приготування та розраховуючи КБЖВ у реальному часі. Проте, незважаючи на високу технологічність, суттєвим обмеженням сервісу залишається відсутність глибинної персоналізації та довготривалого зберігання даних. Бот функціонує як зовнішній консультативний ресурс: він надає відповіді на поточні запити, але не дозволяє користувачу накопичувати власний цифровий архів оцифрованих рецептів. Кожен сеанс взаємодії є відокремленим, що унеможливлює створення персональної бази знань на основі сімейної кулінарної спадщини. Окрім цього, як і в попередньому випадку, бот позбавлений OCR-модуля, що змушує користувача вводити всі дані вручну, нівелюючи переваги інтелектуального оброблення при роботі з паперовими носіями.

Підсумовуючи результати аналізу існуючих рішень, можна констатувати, що на сьогодні ринок пропонує широке різноманіття інструментів, проте жоден із них не забезпечує повного циклу оцифрування та систематизації кулінарних рукописів. Проаналізовані боти чітко розділяються на дві функціональні групи, кожна з яких має критичні недоліки для вирішення поставленої задачі.

Порівняння характеристик всіх розглянутих ботів подано в таблиці 1.1.

Таблиця 1.1 – Порівняння функціональних можливостей проаналізованих ботів у Telegram

Критерій	OCRBot	Argus	Кулінарні рецепти	AI Chef Bot
Розпізнавання друкованого тексту	Так	Так	Ні	Ні
Розпізнавання рукописного тексту	Так (з помилками)	Так	Ні	Ні
Підтримка української мови	Так (нестабільна)	Так	Так	Так
Інтелектуальна корекція тексту	Ні	Так	Ні	Так
Кулінарна спеціалізація	Ні	Ні	Так	Так
Автоматична структуризація рецепту	Ні	Ні (тільки за запитом)	Так	Так
Створення персональної бази знань	Ні	Ні	Ні	Ні
Мультимодальність	Ні	Так	Ні	Ні

Зокрема, універсальні OCR-сервіси, такі як @ocrbot та Argus, хоча й демонструють високу ефективність у роботі з друкованими текстами, виявляються нестабільними при розпізнаванні індивідуального рукопису та специфічної української лексики.

Навіть за умови успішної інтелектуальної корекції тексту, яку пропонує Argus, фінальний результат залишається неструктурованим масивом символів, що не дозволяє автоматично виділяти інгредієнти чи етапи приготування для подальшого пошуку.

З іншого боку, спеціалізовані кулінарні боти, прикладом яких є «Кулінарні рецепти» та AI Chef Bot, пропонують зразковий рівень візуалізації та структуризації даних, проте вони є «закритими» системами. Такі сервіси повністю позбавлені мультимодальних функцій. Це змушує користувача вводити дані вручну, що нівелює переваги автоматизації та робить неможливим оцифрування фізичних носіїв, таких як старі сімейні зошити чи записники.

Важливою прогалиною всіх аналізованих продуктів є повна відсутність можливості створення персональної бази знань. Жоден із ботів не забезпечує довготривалого зберігання оцифрованих даних у вигляді особистого архіву з функціями фільтрації та редагування.

Отже, аналіз підтверджує, що розроблення вузькоспеціалізованого мультимодального бота, що поєднує в собі адаптований OCR-двигун, NLP-аналіз для структуризації та систему персоналізованого зберігання даних, є актуальним практичним завданням.

1.2 Аналіз літературних джерел щодо апробації результатів розроблення мультимодальних ботів в Telegram для оцифрування та систематизації рукописних текстів, зокрема кулінарних рецептів

Наукова література останніх років активно обговорює питання автоматизації оброблення рукописних документів за допомогою штучного інтелекту. Дослідники звертають увагу на складність розпізнавання неструктурованих текстів у неформальних документах, таких як особисті записи. Багато праць присвячено порівнянню ефективності різних архітектур нейронних мереж, зокрема CNN та RNN, для завдань OCR [11].

Важливим аспектом є також використання трансформерів для розуміння контексту речень після етапу візуального розпізнавання. Огляд цих джерел дозволяє обрати найбільш оптимальний технологічний стек для розроблення бота.

Окремий пласт досліджень фокусується на застосуванні месенджерів як платформ для надання інтелектуальних послуг. Автори зазначають, що Telegram пропонує найбільш гнучкий інструментарій для розробників серед усіх популярних платформ. Наукові статті описують досвід створення ботів для освіти [12], медицини [13] та державного сектору [14], де критично важливим є швидкий доступ до даних. Використання вебхуків та асинхронних запитів дозволяє ботам обробляти великі масиви інформації без затримок. Ці напрацювання закладають теоретичний фундамент для проєктування архітектури майбутнього застосунку.

Питання розпізнавання саме кулінарних рецептів розглядається у контексті доменно-орієнтованого аналізу текстів [15]. Специфіка цієї галузі полягає у великій кількості числових значень, одиниць виміру та специфічних дієслів приготування. У літературі [16, 17] пропонується використовувати онтології для зв’язування різних назв одних і тих самих інгредієнтів. Це дозволяє системі розуміти, що «картопля» та «бульба» в рецептах різних регіонів означають один і той самий продукт. Створення таких словників є необхідним етапом для якісної систематизації оцифрованих даних.

Дослідження у сфері OCR рукописних текстів вказують на важливість етапу сегментації рядків та окремих слів [18]. Автори підкреслюють, що нахил рядків та нерівномірні інтервали є основними джерелами помилок у класичних алгоритмах. Використання глибоких мереж сегментації дозволяє ізолювати кожне слово для подальшої класифікації з високою точністю. Окремі праці описують методи синтетичної генерації даних для покращення якості розпізнавання на рідкісних мовах [19, 20]. Це особливо актуально для української мови, де обсяг доступних рукописних баз даних є обмеженим, що впливає на якість розпізнавання навіть у хмарних OCR-сервісах.

Проблематика NLP в кулінарній сфері також включає завдання автоматичного виділення структури страви з суцільного тексту [21]. Науковці пропонують використовувати графові моделі для відображення послідовності дій у процесі приготування. Кожна дія (наприклад, «нарізати», «смажити») стає вузлом графа, а інгредієнти – його атрибутами. Такий підхід дозволяє боту не просто зберігати текст, а й створювати інтерактивні інструкції. Користувач може отримувати нагадування про наступний крок або переглядати таймер прямо в чаті.

Важливим джерелом інформації є звіти про реалізацію відкритих бібліотек для оброблення природної мови, таких як SpaCy та NLTK [22, 23]. У них детально описані методи токенізації та лематизації, які працюють для слов'янських мов [24]. Автори наголошують на необхідності врахування відмінків та морфологічних особливостей при пошуку інгредієнтів у тексті. Без належного лінгвістичного оброблення пошук «яблуко» не знайде рецепт, де написано «три яблука». Використання цих знань дозволяє зробити пошук у боті набагато інтелектуальнішим.

Водночас стрімкий розвиток великих мовних моделей відкриває альтернативний підхід до задач витягування та структуризації інформації з тексту. У роботі [5] автори продемонстрували, що попередньо навчені LLM здатні виконувати одночасно розпізнавання іменованих сутностей та витягування зв'язків, перетворюючи неструктурований текст у формат JSON без необхідності створення окремих правил для кожного типу сутності.

На відміну від класичних NLP-бібліотек, де потрібне окреме налаштування токенізації, лематизації та правил для кожної мови, LLM-підхід дозволяє адаптувати систему до конкретного домену через налаштування шаблону текстової інструкції [4]. Це є особливо важливим для текстів з довільною структурою, таких як рукописні кулінарні рецепти, де порядок елементів, скорочення та стиль запису варіюються від автора до автора. Огляд класичних методів NLP, наведений вище, залишається важливим теоретичним контекстом, що дозволяє зрозуміти обмеження підходів на основі правил та обґрунтувати вибір LLM для задачі структуризації кулінарних рецептів.

Застосування сучасних мовних моделей безпосередньо до кулінарного домену досліджено у роботі [25], де автори запропонували методологію витягування харчових сутностей з кулінарних рецептів на основі інженерії текстових інструкцій у zero-shot-режимі, тобто без жодних попередньо розмічених навчальних даних. Експерименти проводились на мовних моделях, що дозволяє мінімізувати обчислювальні ресурси та підвищити швидкість оброблення.

Результати підтвердили, що LLM здатні розпізнавати назви інгредієнтів, одиниці виміру та кулінарні дії з перспективною точністю навіть без доменного донавчання. Це є вагомим аргументом на користь обраного у даній роботі підходу, оскільки рукописні кулінарні рецепти містять аналогічні типи сутностей, а створення великого розміченого корпусу для класичних NLP-методів є трудомістким та непрактичним для вузькоспеціалізованих задач.

Дослідження у галузі людино-машинної взаємодії свідчать, що тип інтерфейсу суттєво впливає на задоволеність користувача.

Зокрема, у роботі [26] експериментально встановлено, що меню-орієнтовані інтерфейси забезпечують вищу сприйняту автономність та нижче когнітивне навантаження порівняно з вільноформатними діалоговими інтерфейсами, що обґрунтовує вибір вбудованих клавіатур як основного засобу навігації у розроблюваній системі. Окреме видання [27] також підтверджує позитивний вплив використання емодзі у повідомленнях чат-ботів на рівень задоволеності взаємодією.

Наукові праці з комп’ютерного зору пропонують методики оцінювання якості розпізнавання тексту за допомогою метрик CER та WER [28]. Використання цих показників дозволяє об’єктивно оцінити ефективність обраної OCR-системи на специфічному наборі даних рукописних кулінарних рецептів. Дослідження показують, що попереднє оброблення зображень суттєво підвищує точність розпізнавання тексту.

Зокрема, у роботі [29] комбінація методу видалення тіней та оптимізованої конвертації у відтінки сірого дозволила знизити загальний рівень помилок OCR до 14,56%, а додаткове постоброблення – до 13,94%.

Огляд літератури щодо систематизації знань показує ефективність використання тегів та категорій для великих масивів даних [30, 31].

Автори пропонують алгоритми автоматичної класифікації текстів за допомогою методів машинного навчання, таких як Naive Bayes або SVM. Водночас використання LLM дозволяє виконувати класифікацію безпосередньо на етапі структуризації тексту. Модель здатна з контексту рецепту автоматично визначати категорію страви, тип кухні та релевантні теги без окремого класифікатора. Це значно спрощує архітектуру системи та забезпечує користувачу зручну навігацію без необхідності вручну сортувати записи.

Питання інтеграції ботів із зовнішніми сервісами через API також розглядається у фаховому виданні [32]. Це дозволяє делегувати обчислювально складні задачі, такі як оптичне розпізнавання тексту або інтелектуальна структуризація даних, спеціалізованим хмарним сервісам. Користувач отримує послугу в одному вікні месенджера, що підвищує цінність застосунку.

Проблеми оцифрування історичних документів мають багато спільного з розпізнаванням старих кулінарних зошитів. У літературі [33, 34] описані методи відновлення вицвілого чорнила або пошкодженого паперу на цифрових копіях зображень. Використання генеративно-змагальних мереж дозволяє реконструювати втрачені фрагменти тексту з високою вірогідністю. Хоча такі методи є обчислювально складними і виходять за межі поточної реалізації, вони становлять перспективний напрямок розвитку системи для роботи з пошкодженими рукописами.

Дослідження щодо апробації результатів подібних розроблень підкреслює важливість систематичного тестування на етапі розроблення [35]. Для оцінки якості OCR-конвеєру використовуються кількісні метрики CER та WER на тестовій вибірці рукописних рецептів, а коректність NLP-систематизації перевіряється шляхом порівняння отриманого JSON з очікуваною структурою.

Ітераційне вдосконалення шаблонів інструкцій для LLM на основі аналізу помилок дозволяє поступово підвищувати точність системи без зміни архітектури.

Важливим аспектом є підтримка багатомовності у розпізнаванні текстів. Кулінарні рецепти часто містять назви інгредієнтів іноземного походження, які можуть бути написані мовою оригіналу, наприклад, французькі чи італійські кулінарні терміни.

Сучасні хмарні OCR-сервіси, зокрема Google Cloud Vision, підтримують автоматичне визначення мови та розпізнавання кількох мов в одному зображенні, що дозволяє коректно обробляти рецепти зі змішаною лексикою без додаткового налаштування. У випадках, коли автоматичне визначення мови є недостатньо точним, API дозволяє передати підказку мови для покращення результатів розпізнавання [3].

На завершення огляду літератури можна стверджувати, що теоретична база для створення мультимодального бота є достатньою. Аналіз класичних методів NLP виявив їхні обмеження для текстів з довільною структурою, тоді як дослідження ефективності LLM для структурованого витягування даних підтверджують доцільність обраного підходу. Сучасні хмарні сервіси для OCR та великі мовні моделі для структуризації тексту дозволяють реалізувати складну систему без необхідності навчання власних моделей. Вивчені джерела вказують на потенційні труднощі та пропонують перевірені методи їх подолання.

1.3 Постановка задачі

Проблема систематизації персональних архівів залишається актуальним завданням, оскільки велика кількість кулінарних рецептів досі існує виключно у форматі рукописних записників. Такий спосіб зберігання обмежує можливості пошуку та створює ризики фізичної втрати інформації. Звичайна цифрова фіксація носіїв у вигляді фотографій є недостатньою, оскільки вона не формує семантичної структури документа.

Впровадження технологій оптичного розпізнавання символів та інтелектуального оброблення природної мови забезпечує комплексне вирішення цієї проблеми, перетворюючи неструктуровані зображення на функціональну цифрову базу знань.

Реалізація подібного інструменту на базі месенджера Telegram забезпечує користувачам максимальну мобільність та доступність сервісу без необхідності встановлення додаткового спеціалізованого програмного забезпечення. Особливої актуальності робота набуває завдяки застосуванню сучасних методів інтелектуального аналізу тексту, що дозволяють розпізнавати семантичну структуру рецепта та автоматично розділяти його на логічні блоки – перелік продуктів та покрокові інструкції.

Розроблення системи, адаптованої до специфіки української мови та особливостей рукописного тексту, є важливою ланкою у створенні персоналізованих розумних асистентів для ведення домашнього господарства. Отже, створення мультимодального бота є актуальною відповіддю на сучасний виклик збереження культурно-побутових традицій.

Об’єктом роботи є процес розроблення мультимодального бота в Telegram.

Метою даної роботи є розроблення та впровадження мультимодального бота в Telegram для автоматизації процесу оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP.

Для досягнення поставленої мети необхідно вирішити низку аналітичних та технічних завдань на різних рівнях архітектури:

– проведення комплексного аналізу існуючих мультимодальних ботів у месенджері Telegram, виявивши їхні переваги та функціональні обмеження в задачах оцифрування рукописних текстів;

– здійснення систематичного огляду наукової літератури та сучасних підходів до розпізнавання та структурування кулінарної інформації, враховуючи специфіку української мови;

– визначення вимог до мультимодальної взаємодії користувача з системою, що включає обробку візуальних та текстових даних для створення цілісного цифрового рецепта;

– вибір та налаштування оптимального OCR-двигуна, адаптованого до специфіки рукописного введення. Потрібно розробити алгоритми попереднього оброблення зображень для компенсації низької якості мобільних знімків;

– розроблення NLP-модуля для інтелектуального аналізу та систематизації розпізнаного тексту українською мовою. Завдання включає розпізнавання сутностей та структурування частин рецепта на основі великих мовних моделей, що забезпечує високу точність оброблення семантичних зв’язків без необхідності донавчання специфічних моделей;

– проєктування та реалізація бази даних для надійного зберігання оцифрованої інформації. Структура бази має підтримувати зв’язки між користувачами, рецептами, фотографіями та тегами. Необхідно забезпечити високу швидкість виконання пошукових запитів навіть при великому обсязі даних;

– розроблення логіки взаємодії користувача з ботом через Telegram API. Інтерфейс має бути максимально простим, з використанням кнопок швидкого доступу та інтуїтивно зрозумілих команд;

– тестування розробленої системи на реальних зображеннях рукописних рецептів та оцінювання якості розпізнавання та структуризації даних.

2 РОЗРОБЛЕННЯ СТРУКТУРИ МУЛЬТИМОДАЛЬНОГО БОТА В TELEGRAM ДЛЯ ОЦИФРУВАННЯ ТА СИСТЕМАТИЗАЦІЇ РУКОПИСНИХ КУЛІНАРНИХ РЕЦЕПТІВ НА ОСНОВІ OCR ТА NLP

2.1 Особливості розроблення мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP

Сучасні тенденції цифровізації побутових процесів зумовлюють необхідність створення ефективних інструментів для автоматизації рутинних завдань, зокрема у сфері збереження кулінарних традицій. Велика кількість унікальних сімейних рецептів досі зберігається виключно у паперовому вигляді, що ускладнює їх швидкий пошук та систематизацію.

Розроблення мультимодального бота в Telegram спрямоване на розв’язання цієї проблеми шляхом поєднання зручного інтерфейсу месенджера з передовими технологіями інтелектуального оброблення даних. Ключовим елементом системи є впровадження OCR-рішень, які дозволяють трансформувати фотографії рукописних записів у структурований цифровий формат з високою точністю. Застосування алгоритмів оброблення природної мови забезпечує автоматичне виділення сутностей, таких як назви страв, переліки інгредієнтів та кроки приготування, для їх подальшої класифікації. Такий комплексний підхід дозволяє створити надійну інтелектуальну платформу, яка перетворює хаотичні записи на впорядковану персональну базу знань, доступну з будь-якого мобільного пристрою.

Важливим аспектом постановки задачі є вибір месенджеру для реалізації проєкту. Платформою обрано месенджер Telegram, що зумовлено кількома практичними чинниками. Месенджер забезпечує крос-платформність: застосунок однаково працює на смартфонах з операційними системами iOS та Android, а також у настільній версії. Користувач може завантажити фотографію з мобільного пристрою, а редагувати результат на комп’ютері.

Інформація синхронізується між усіма пристроями через централізовану базу даних. Крім того, Telegram надає програмний інтерфейс для розроблення ботів із підтримкою вбудованих клавіатур, діалогових сценаріїв та форматованих повідомлень, що дозволяє побудувати повноцінний інтерфейс взаємодії без розроблення окремого мобільного застосунку.

Система має поєднувати кілька технологічних доменів: оптичне розпізнавання тексту для перетворення зображення на текст, оброблення природної мови для структуризації розпізнаного тексту та реляційну базу даних для зберігання та пошуку результатів.

Функціональні вимоги до системи визначено на основі типового сценарію використання: користувач фотографує рукописний рецепт та надсилає зображення боту, після чого система має автоматично виконати повний цикл оброблення та повернути структурований результат. Відповідно до цього сформовано такі вимоги:

– завантаження фотографії рукописного рецепта з автоматичним виявленням дублікатів серед раніше завантажених зображень поточного користувача;

– автоматичне розпізнавання рукописного тексту українською мовою із попереднім обробленням зображення для підвищення якості розпізнавання;

– семантичний контроль розпізнаного тексту, тобто перевірка належності до кулінарної тематики перед подальшим обробленням;

– автоматична структуризація тексту з виділенням назви страви, переліку інгредієнтів із зазначенням кількості та одиниць виміру, покрокових інструкцій приготування, часу приготування, кількості порцій та рівня складності;

– автоматичне визначення категорії страви, типу кухні, рівня складності та тегів на основі контексту рецепта для подальшої систематизації;

– можливість коригування отриманого результату двома способами: точковим редагуванням окремих полів та інтелектуальною корекцією за текстовим описом бажаних змін;

– збереження рецептів у персональну колекцію з можливістю фільтрації за категорією, тегами, типом кухні та рівнем складності;

– пошук збережених рецептів за назвою та за інгредієнтами, зокрема нечіткий пошук, стійкий до помилок розпізнавання та варіантів написання;

– експорт рецепту у формат PDF та можливість перегляду оригінальної фотографії рукопису поруч зі структурованим результатом.

Нефункціональні вимоги стосуються надійності, швидкодії та безпеки програмного продукту. Всі персональні дані користувачів мають бути захищені відповідно до сучасних вимог щодо захисту персональних даних [36]. Бот повинен коректно обробляти помилкові дії користувача та надавати зрозумілі підказки щодо їх виправлення. На кожному етапі оброблення система забезпечує зворотний зв’язок, інформуючи користувача про поточний стан та можливі помилки зрозумілою мовою замість технічних повідомлень про винятки.

У межах роботи також необхідно провести тестування розробленого бота на різних наборах даних. Будуть використані фотографії рецептів з різним почерком, освітленням та станом паперу. Це дозволить оцінити реальну ефективність обраних методів OCR та NLP у польових умовах. Порівняння результатів розпізнавання з еталонними текстами дозволить вирахувати точність роботи системи. На основі отриманих даних будуть зроблені висновки про готовність бота до експлуатації.

Очікуваним результатом роботи є функціональний прототип бота в Telegram, що демонструє повний цикл оцифрування рукописного рецепту, від фотографії до структурованого запису в базі даних.

Користувач зможе розпочати роботу з ботом, просто натиснувши кнопку «Старт» та надіславши фото рецепта. Система повинна продемонструвати високий рівень автономності та мінімальну кількість помилок при обробленні рецептів з різним почерком та умовами зйомки.

Отриманий програмний продукт стане доказом ефективності поєднання технологій OCR та NLP для побутових потреб. Подальші розділи присвячені безпосередньому проєктуванню структури бази даних та моделюванню архітектури бота.

2.2 Особливості структури бази даних мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP

2.2.1 Аналіз особливостей даних та обґрунтування моделі зберігання даних

Проєктування бази даних для мультимодального бота вимагає попереднього аналізу типів даних, що циркулюють у системі, оскільки саме природа цих даних визначає вибір моделі зберігання та архітектурні рішення.

Система оцифрування рукописних кулінарних рецептів оперує трьома принципово різними категоріями даних, кожна з яких висуває власні вимоги до зберігання та оброблення.

Перша категорія – це структуровані дані. До них належать профілі користувачів бота (Telegram ID, ім’я, мова інтерфейсу), рецепти з чітко визначеними атрибутами (назва, опис, категорія, кухня), інгредієнти з кількістю та одиницями виміру, а також покрокові інструкції приготування. Ці дані мають стабільну, передбачувану структуру, між ними існують чіткі зв’язки (один користувач має багато рецептів, один рецепт містить багато інгредієнтів, один інгредієнт входить у багато рецептів), і до них висуваються вимоги цілісності, наприклад, інгредієнт у рецепті повинен посилатися на реально існуючий запис у довіднику інгредієнтів.

Друга категорія – напівструктуровані дані. Це артефакти, що генеруються зовнішніми сервісами під час оброблення: сирий текст, який повертає OCR-сервіс із супутніми метаданими (рівень впевненості розпізнавання кожного слова, визначена мова), а також повні відповіді LLM-моделі, що виконує структурний аналіз розпізнаного тексту.

Особливість цих даних полягає в тому, що їхній формат контролюється не розроблюваною системою, а зовнішніми сервісами, і може змінюватися з оновленням версій API. Жорстке визначення схеми для таких даних є недоцільним, бо воно вимагатиме модифікації структури бази при кожній зміні формату відповіді зовнішнього сервісу.

Третя категорія – це бінарні дані. Це фотографії рукописних рецептів, які завантажують користувачі. Зберігання бінарних файлів безпосередньо в базі даних є неефективним з точки зору продуктивності та масштабування, тому в системі зберігаються лише посилання – ідентифікатор файлу в Telegram API для повторного завантаження та URL у зовнішньому об’єктному сховищі.

Окрім типізації даних, на вибір моделі зберігання впливають функціональні вимоги системи. Ключова вимога – підтримка пошуку рецептів за інгредієнтами з урахуванням синонімів та регіональних варіантів назв, характерних для української мови (наприклад, «картопля», «бараболя» та «картошка» мають розпізнаватися як один інгредієнт). Це вимагає нормалізації довідника інгредієнтів та окремої таблиці синонімів. Ще одна вимога – це толерантність пошуку до помилок OCR, оскільки розпізнавання рукописного тексту неминуче генерує неточності. Система має знаходити інгредієнт «борошно», навіть якщо OCR повернув «бороашно».

На основі проведеного аналізу обрано реляційну модель як основну модель зберігання. Цей вибір обумовлений кількома факторами.

По-перше, домен кулінарних рецептів має виражену реляційну природу: сутності пов’язані чіткими зв’язками різної кардинальності, які природно виражаються через зовнішні ключі.

По-друге, вимога нормалізації інгредієнтів для надійного пошуку із синонімами реалізується саме через реляційні механізми, а саме довідникові таблиці та зв’язкові таблиці типу «багато-до-багатьох».

По-третє, реляційна модель забезпечує цілісність посилань на рівні СУБД, що гарантує консистентність даних без додаткової логіки на рівні застосунку.

Водночас для напівструктурованих даних (метадані OCR, відповіді LLM) застосовано гібридний підхід, тобто використання JSONB-полів усередині реляційної схеми. Це дозволяє зберігати повну відповідь зовнішнього сервісу без необхідності проєктувати жорстку схему для даних, формат яких не контролюється системою, і водночас залишатися в єдиному середовищі зберігання без впровадження окремої документної бази.

Окремо слід виділити архітектурний принцип, покладений в основу проєктування: незмінність сирих даних. Оригінальне зображення, сирий OCR-текст та повна відповідь LLM-моделі зберігаються паралельно зі структурованим результатом і ніколи не перезаписуються. Це рішення обумовлене специфікою OCR/NLP конвеєрів: при вдосконаленні моделей розпізнавання або зміні інструкцій для LLM-аналізатора з’являється можливість повторно обробити вже завантажені рецепти без залучення користувача, отримуючи покращені результати на тих самих вхідних даних. Без збереження сирих артефактів таке повторне оброблення було б неможливим.

2.2.2 Проєктування структури бази даних

На основі проведеного аналізу особливостей даних спроєктовано реляційну структуру бази даних, що складається з чотирнадцяти сутностей, організованих у три логічні шари: шар користувацьких даних, шар артефактів оброблення та шар довідників. Концептуальну модель даних представлено на рисунку 2.1 у вигляді ER-діаграми.

Шар користувацьких даних містить сутності, що безпосередньо відповідають бізнес-об’єктам предметної області – користувачам та їхнім рецептам.

Сутність users зберігає профілі користувачів бота. Основним ідентифікатором слугує внутрішній UUID, тоді як Telegram ID зберігається окремо з обмеженням унікальності.

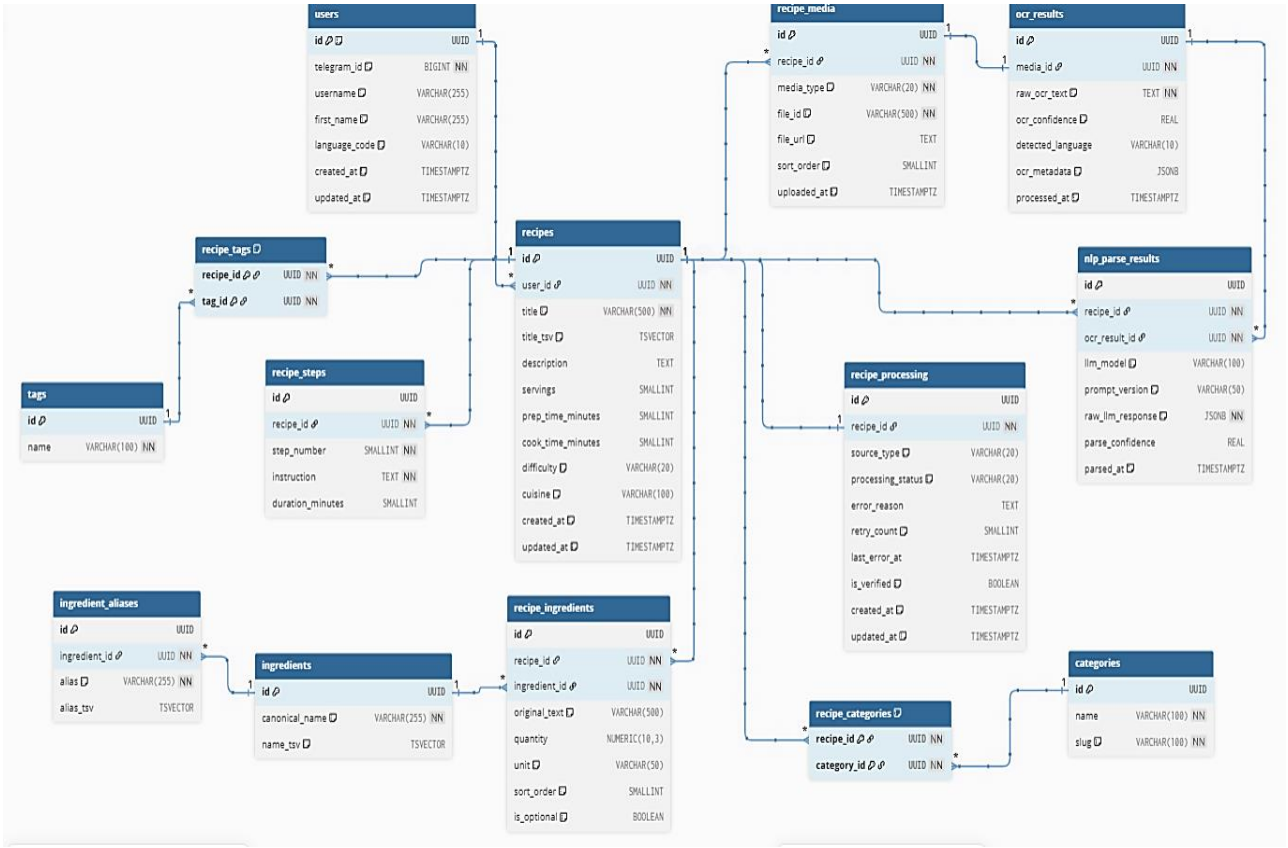


Рисунок 2.1 – ER-діаграма структури бази даних

Таке рішення прийнято для забезпечення незалежності внутрішньої логіки системи від зовнішнього ідентифікатора Telegram API, тому у разі міграції бота на іншу платформу внутрішні зв’язки між таблицями не потребуватимуть змін. Додатково зберігаються ім’я користувача та мова інтерфейсу з типовим значенням «uk», що дозволяє адаптувати відповіді бота.

Сутність recipes є центральною таблицею системи, навколо якої побудована вся структура. Кожен рецепт належить конкретному користувачу через зовнішній ключ user_id. Обов’язковим атрибутом є лише назва рецепту – всі інші поля (опис, кількість порцій, час підготовки та приготування, складність, тип кухні) є необов’язковими.

Це рішення обумовлене тим, що джерелом даних є рукописний текст, у якому зазвичай зафіксовано лише назву, інгредієнти та кроки приготування, тоді як такі атрибути, як час приготування чи складність, рідко зазначаються в домашніх записах. Ці поля передбачені для подальшого збагачення – вручну користувачем або автоматично LLM-аналізатором, якщо відповідна інформація присутня в тексті.

Окремо в таблиці зберігається згенерований повнотекстовий вектор назви (тип TSVECTOR), що використовується для повнотекстового пошуку рецептів.

Сутність recipe_steps зберігає покрокові інструкції приготування. Кожен крок належить конкретному рецепту, має порядковий номер та текст інструкції. Необов'язковий атрибут тривалості кроку в хвилинах заповнюється, якщо модуль NLP зміг витягти цю інформацію з тексту.

Шар артефактів виконання містить сутності, що фіксують проміжні та кінцеві результати конвеєра оцифрування: від завантаженого зображення через OCR-розпізнавання до семантичного оброблення тексту.

Сутність recipe_media зберігає метадані завантажених зображень рукописних рецептів. Кожен запис пов'язаний з конкретним рецептом та містить ідентифікатор файлу в Telegram API (file_id), який дозволяє повторно завантажити зображення без його фізичного зберігання на сервері бота, а також URL у зовнішньому об'єктному сховищі для довготривалого зберігання. Атрибут media_type розрізняє фотографії та документи, оскільки Telegram API обробляє їх по-різному. Поточна реалізація передбачає одне зображення на рецепт, проте структура спроєктована з урахуванням можливого розширення для підтримки багатосторінкових рецептів – зв'язок між recipes та recipe_media є типу «один-до-багатьох», а атрибут sort_order дозволяє зберігати порядок сторінок.

Сутність ocr_results зберігає результати оптичного розпізнавання символів. Зв'язок із recipe_media є типу «один-до-одного», тобто кожне зображення має рівно один результат OCR. Ключовим атрибутом є raw_ocr_text повний розпізнаний текст, що зберігається незалежно від якості розпізнавання.

Числовий атрибут ocr_confidence фіксує середню впевненість розпізнавання у діапазоні від 0 до 1, що дозволяє системі автоматично позначати результати з низькою якістю для ручної перевірки. Атрибут detected_language зберігає мову, визначену OCR-сервісом. Окремо в JSON-полі ocr_metadata зберігаються додаткові структурні дані від OCR-сервісу: координати текстових блоків на зображенні, порівневий рівень впевненості та інша діагностична інформація, формат якої визначається зовнішнім API.

Сутність nlp_parse_results фіксує результати структурного оброблення розпізнаного тексту великою мовною моделлю. Зв’язок із recipes є типу «один-до-багатьох», що забезпечує версіонування: під час повторного аналізу, зміни моделі або текстової інструкції, а також при інтелектуальній корекції рецепта зберігається новий запис поруч із попереднім. Для кожного результату фіксується назва використаної LLM-моделі та версія інструкції, що дозволяє відстежувати, яка комбінація параметрів дала кращий результат. Повна відповідь LLM зберігається у JSON-полі raw_llm_response як аудиторський запис; у випадку інтелектуальної корекції це поле додатково містить коментар користувача та попередній стан рецепту, що забезпечує повну історію змін. Додатково зберігається зв’язок із конкретним записом ocr_results через зовнішній ключ ocr_result_id, що забезпечує повну простежуваність ланцюжка: зображення → OCR-текст → відповідь LLM → структуровані дані.

Сутність recipe_processing відстежує поточний стан оброблення рецепту. Зв’язок із recipes є типу «один-до-одного» – кожен рецепт має рівно один запис стану. Виділення цієї інформації в окрему сутність, а не додавання відповідних атрибутів безпосередньо до таблиці recipes, обумовлене принципом розмежування відповідальності: таблиця recipes зберігає бізнес-дані рецепту, тоді як recipe_processing містить виключно технічні метадані конвеєру оброблення. Атрибут processing_status відображає поточну фазу перетворення та набуває значень pending, ocr_done, parsed, verified або failed. Атрибут source_type вказує на джерело рецепту – OCR-розпізнавання, ручне введення або імпорт.

Для діагностики помилок передбачені атрибути error_reason, retry_count та last_error_at, що дозволяють реалізувати механізм повторних спроб оброблення. Шар довідників забезпечує нормалізацію, класифікацію та пошук рецептів.

Сутність ingredients є довідником канонічних назв інгредієнтів. Кожен інгредієнт зберігається в єдиному екземплярі з нормалізованою назвою (наприклад, «борошно пшеничне», «цибуля ріпчаста»).

Обмеження унікальності на полі canonical_name гарантує відсутність дублікатів у довіднику. Для підтримки повнотекстового пошуку зберігається згенерований TSVECTOR-вектор назви.

Сутність ingredient_aliases зберігає синоніми та регіональні варіанти назв інгредієнтів. Один інгредієнт може мати довільну кількість синонімів, наприклад, канонічний запис «картопля» може мати синоніми «бараболя», «картошка», «бульба». Це рішення є критичним для коректного пошуку в контексті української мови, де один інгредієнт може мати кілька усталених назв залежно від регіону. Без таблиці синонімів пошук за запитом «бараболя» не знайшов би рецептів, де NLP-модуль нормалізував цей інгредієнт як «картопля». Довідник синонімів формується на етапі розгортання системи на основі зібраного переліку регіональних варіантів назв українською мовою та доповнюється в процесі експлуатації при виявленні нових варіантів.

Зв’язкова сутність recipe_ingredients реалізує зв’язок типу «багато-до-багатьох» між рецептами та інгредієнтами. Окрім зовнішніх ключів на recipe_id та ingredient_id, ця таблиця зберігає як структуровані дані (числова кількість quantity, одиниця виміру unit), так і оригінальний текст із рукопису в полі original_text. Збереження оригінального тексту (наприклад, «2 скл. борошна» або «жменька кропу») поруч зі структурованим представленням є принциповим рішенням: воно дозволяє відобразити рецепт автентично, як його записав автор, і водночас забезпечити структурований пошук та можливість перерахунку кількості на іншу кількість порцій. Атрибут sort_order зберігає порядок інгредієнтів у рецепті, а is_optional позначає необов’язкові інгредієнти.

Сутність categories містить контрольований довідник для базової класифікації рецептів («Супи», «Десерти», «Випічка»). На відміну від сутності tags, яка реалізує систему вільного тегування з можливістю динамічного додавання значень («бюджетне», «без глютену»), категорії формують чітку таксономічну структуру. Довідник категорій заповнюється на етапі ініціалізації системи попередньо визначеним переліком значень, що забезпечує контрольовану класифікацію та уникнення дублікатів або некоректних категорій, які могла б згенерувати мовна модель.

Хоча зв’язкова сутність recipe_categories технічно реалізує відношення типу «багато-до-багатьох», на рівні прикладної логіки поточної версії системи встановлено обмеження: кожен рецепт асоціюється лише з однією категорією. Таке рішення забезпечує сувору класифікацію даних, зберігаючи при цьому гнучкість архітектури для розширення функціоналу в майбутньому. Водночас для сутності tags передбачено повноцінне використання зв’язку «багато-до-багатьох», що дозволяє присвоювати одному рецепту необмежену кількість дескрипторів. Обидві зв’язкові таблиці використовують композитні первинні ключі, сформовані з пар зовнішніх ключів відповідних сутностей.

2.2.3 Забезпечення цілісності даних та нормалізація

Спроєктована структура бази даних потребує чітко визначених правил підтримки цілісності, оскільки система оперує складною мережею взаємопов’язаних сутностей, видалення одного рецепту зачіпає до восьми залежних таблиць, і некоректна поведінка при видаленні може призвести як до втрати цінних даних, так і до появи «осиротілих» записів, що не належать жодному рецепту.

Для зовнішніх ключів, що пов’язують дочірні сутності рецепту з таблицею recipes, застосовано правило ON DELETE CASCADE. Це означає, що при видаленні рецепту автоматично видаляються всі пов’язані записи: метадані зображення у recipe_media, результати розпізнавання в ocr_results, результати структуризації в nlp_parse_results, стан опрацювання в recipe_processing, зв’язки з інгредієнтами в recipe_ingredients, покрокові інструкції в recipe_steps, а також записи у зв’язкових таблицях recipe_categories та recipe_tags. Таке рішення відповідає бізнес-логіці: рецепт є атомарною одиницею з точки зору користувача, тому його вилучення має призводити до повного очищення бази даних від усіх супутніх технічних артефактів та проміжних результатів опрацювання.

Аналогічне правило ON DELETE CASCADE застосовано для зв’язку між recipes та users, при видаленні облікового запису користувача видаляються всі його рецепти з усіма залежними даними. Це забезпечує право користувача на повне видалення своїх даних із системи.

Для зовнішніх ключів, що посилаються на довідникові таблиці ingredients, categories та tags, застосовано протилежне правило – ON DELETE RESTRICT. Воно забороняє видалення запису з довідника, доки на нього існує хоча б одне посилання з інших таблиць. Наприклад, неможливо видалити інгредієнт «борошно» з довідника, якщо він використовується хоча б в одному рецепті. Це запобігає випадковій втраті зв’язків та появі посилань на неіснуючі записи. Якщо необхідно видалити інгредієнт з довідника, спочатку потрібно або видалити всі рецепти, що його містять, або замінити посилання на інший інгредієнт.

Окрім каскадної поведінки, цілісність даних забезпечується обмеженнями на рівні окремих таблиць. Обмеження унікальності (UNIQUE) на полях telegram_id у таблиці users, canonical_name у таблиці ingredients та name у таблицях categories і tags гарантують відсутність дублікатів у ключових точках системи. Обмеження NOT NULL на обов’язкових атрибутах (title у recipes, raw_ocr_text у ocr_results, instruction у recipe_steps) унеможливлюють створення записів без мінімально необхідної інформації. Обмеження CHECK на полі difficulty у таблиці recipes обмежує допустимі значення складності рецепту переліком «легко», «середньо», «складно», запобігаючи потраплянню довільних рядків.

Спроєктована схема відповідає третій нормальній формі. Кожна таблиця має визначений первинний ключ, усі не ключові атрибути функціонально залежать від повного первинного ключа, і в схемі відсутні транзитивні залежності між не ключовими атрибутами.

Дотримання третьої нормальної форми найбільш наочно проявляється у проєктуванні шару довідників. Виділення інгредієнтів у окремий довідник усуває транзитивну залежність, яка виникла б при зберіганні назви інгредієнта безпосередньо в таблиці recipe_ingredients: у такому випадку canonical_name залежав би від ingredient_id, який, у свою чергу, не є первинним ключем таблиці recipe_ingredients.

Аналогічно, зв’язкові таблиці recipe_categories та recipe_tags усувають повторювані групи, які виникли б при зберіганні списків категорій або тегів безпосередньо в таблиці рецептів, наприклад, у вигляді текстового поля з розділювачами або масиву значень, що порушувало б першу нормальну форму.

Свідомим та обґрунтованим відхиленням від повної нормалізації є JSONB-поля ocr_metadata в таблиці ocr_results та raw_llm_response в таблиці nlp_parse_results. Ці поля зберігають денормалізовані структури, проте їх нормалізація була б недоцільною з кількох причин.

По-перше, формат цих даних контролюється зовнішніми сервісами і може змінюватися з оновленням їхніх API, тому жорстка реляційна схема вимагала б модифікації при кожній такій зміні.

По-друге, нормалізація відповіді OCR-сервісу вимагала б створення окремих таблиць для текстових блоків, координат, рівнів впевненості кожного слова, що суттєво ускладнило б схему.

По-третє, ці дані використовуються виключно для діагностики та аудиту, система не виконує реляційних запитів до внутрішньої структури JSONB-полів, а отже, реляційне представлення не дало б практичної переваги для пошуку чи фільтрації.

2.2.4 Вимоги до пошуку та індексування

Система оцифрування рукописних рецептів висуває специфічні вимоги до пошуку, що відрізняють її від типових CRUD-застосунків. Основна складність полягає в тому, що дані, за якими здійснюється пошук, проходять через ланцюжок перетворень – рукописний текст розпізнається OCR-сервісом з неминучими помилками, а потім структурується NLP-модулем, який також може допускати неточності. Це означає, що пошукова підсистема має бути толерантною до помилок на кожному етапі цього ланцюжка.

Аналіз сценаріїв використання бота дозволив виділити чотири основні сценарії пошуку, кожен з яких висуває власні вимоги до індексування.

Перший сценарій – це пошук рецептів за назвою. Користувач вводить назву страви або її частину, і система має знайти відповідні рецепти. Цей сценарій вимагає повнотекстового пошуку з підтримкою морфології – запит «вареники» має знаходити рецепти з назвою «Вареники з картоплею» та «Лінивий вареник». Використовуються TSVECTOR-поля, що зберігають обчислені повнотекстові вектори назв рецептів, та GIN-індекси, що забезпечують пошук за цими векторами без повного сканування таблиці.

Другий сценарій відповідає за пошук рецептів за інгредієнтами. Користувач вказує один або кілька інгредієнтів, і система повертає рецепти, що їх містять. Складність цього сценарію полягає в необхідності враховувати синоніми та регіональні варіанти назв. Запит «лаврушка» має знаходити рецепти, де інгредієнт нормалізовано як «лавровий лист». Для цього повнотекстовий пошук застосовується одночасно до двох таблиць: ingredients (за полем canonical_name) та ingredient_aliases (за полем alias), з відповідними GIN-індексами на TSVECTOR-полях обох таблиць.

Третій сценарій – це нечіткий пошук з толерантністю до помилок OCR. Це специфічна вимога, безпосередньо пов’язана з природою даних у системі.

Розпізнавання рукописного тексту неминуче генерує помилки: літери можуть бути розпізнані неправильно, пропущені або додані. Наприклад, OCR може повернути «бороашно» замість «борошно» або «ципуля» замість «цибуля». Система має знаходити правильний інгредієнт навіть при наявності таких спотворень. Для цього використовується триграмний пошук на основі розширення pg_trgm, яке порівнює рядки за збігом трисимвольних підрядків. GIN-індекси з оператором gin_trgm_ops на полях canonical_name та alias забезпечують ефективне виконання нечітких запитів. Функція similarity() повертає числову міру подібності двох рядків у діапазоні від 0 до 1, що дозволяє ранжувати результати за ступенем відповідності та встановлювати пороговий рівень для фільтрації нерелевантних збігів.

Четвертий сценарій є фільтрацією за категоріями, тегами та іншими атрибутами. Користувач обирає категорію («Напій», «Десерт») або тег («швидке», «бюджетне»), і система повертає відповідні рецепти. Цей сценарій не вимагає повнотекстового чи нечіткого пошуку – достатньо точного зіставлення за зовнішніми ключами. Для ефективного виконання таких запитів використовуються стандартні B-Tree індекси на полях зовнішніх ключів у зв'язкових таблицях recipe_categories та recipe_tags, а також на полях user_id та cuisine в таблиці recipes. B-Tree індекси також застосовуються до поля processing_status у таблиці recipe_processing, що дозволяє швидко відбирати рецепти за станом опрацювання для формування черги повторного виконання операцій або відображення користувачу лише повністю завершених кулінарних записів.

Отже, стратегія індексування поєднує три типи індексів, кожен з яких відповідає за окремий клас пошукових запитів: GIN-індекси на TSVECTOR-полях для повнотекстового пошуку з морфологією, GIN-індекси з підтримкою триграм для нечіткого пошуку з толерантністю до помилок OCR, та B-Tree індекси для точної фільтрації за зовнішніми ключами та атрибутами. Така комбінація забезпечує ефективне покриття всіх визначених сценаріїв пошуку, зберігаючи при цьому прийнятну продуктивність при зростанні обсягу даних.

Підсумовуючи підрозділ 2.2, спроєктована структура бази даних відображає специфіку предметної області, а саме оцифрування рукописних кулінарних рецептів, та враховує особливості даних, що генеруються OCR/NLP модулями. Обрана реляційна модель з елементами документного зберігання забезпечує баланс між строгістю структури для бізнес-даних та гнучкістю для артефактів зовнішніх сервісів. Принцип незмінності сирих даних гарантує можливість повторного опрацювання при вдосконаленні моделей розпізнавання.

Визначені правила каскадної поведінки зовнішніх ключів та обмеження цілісності гарантують консистентність даних на рівні СУБД без додаткової логіки на рівні застосунку.

2.3 Моделювання структури мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP

2.3.1 Моделювання загальної архітектури бота

Архітектура розроблюваного мультимодального бота має бути побудована за принципом модульності, що передбачає чіткий розподіл функціональних обов’язків між окремими компонентами системи. Модульний підхід обрано з огляду на те, що система інтегрує кілька зовнішніх сервісів, кожен з яких може змінювати свій API або бути замінений альтернативним рішенням. Відокремлення компонентів дозволяє вносити такі зміни локально, не впливаючи на решту системи.

З чотирьох основних компонентів складається архітектура бота: модуль взаємодії з Telegram API, модуль оптичного розпізнавання тексту, модуль оброблення природної мови та модуль роботи з базою даних. Координація послідовності виконання операцій між компонентами покладена на сервісний шар у складі модуля бота, який забезпечує виклик OCR та NLP конвеєрів, механізм повторних спроб при тимчасовій недоступності зовнішніх сервісів та логування результатів кожного етапу оброблення.

Модуль взаємодії з Telegram API є вхідною точкою системи. Його завданням є маршрутизація вхідних повідомлень до відповідних обробників залежно від типу повідомлення: текстова команда, зображення або натискання вбудованої кнопки. Для організації обробників застосовано підхід на основі компонентів маршрутизації (Router), що дозволяє логічно групувати функціональність за призначенням (реєстрація, завантаження фото, перегляд рецептів, пошук) та підключати нові групи обробників без модифікації існуючих.

Модуль OCR відповідає за перетворення фотографії рукописного рецепту на машиночитаний текст із попереднім обробленням зображення та перевіркою результату, а NLP-модуль перетворює неструктурований текст у впорядковану схему даних рецепту засобами великої мовної моделі.

Модуль роботи з базою даних спроєктований за патерном Repository, що передбачає створення окремого репозиторію для кожної сутності (користувач, рецепт, інгредієнт). Патерн Repository обрано для абстрагування шару доступу до даних від решти логіки застосунку. Взаємодію з базою даних передбачено реалізувати в асинхронному режимі, щоб запити до бази не блокували основний цикл оброблення повідомлень. Це критично важливо для забезпечення відгуку бота при одночасній роботі кількох користувачів.

Загальна послідовність опрацювання рецепту визначається сервісним шаром:

отримання зображення від користувача → передача до OCR-конвеєра → автоматична передача результату в NLP-конвеєр → збереження структурованого рецепту в базу даних → формування відповіді користувачу.

Весь конвеєр виконується автоматично як єдиний процес: користувач надсилає фото і одразу отримує структурований результат для перевірки, після чого може зберегти рецепт, відредагувати окремі поля або скорегувати результат за допомогою мовної моделі. Така послідовність забезпечує поетапну трансформацію даних від неструктурованого зображення до структурованого запису та мінімізує кількість взаємодій з користувачем.

Взаємодію із зовнішніми сервісами спроєктовано як асинхронну, що дозволяє боту продовжувати опрацювання інших повідомлень під час очікування відповідей від хмарних API. Це рішення обумовлене тим, що затримки зовнішніх сервісів можуть сягати кількох секунд, і синхронне очікування призводило б до блокування роботи для всіх користувачів.

На рисунку 2.2 представлено діаграму компонентів, яка відображає загальну архітектуру бота та взаємозв’язки між його модулями і зовнішніми сервісами.

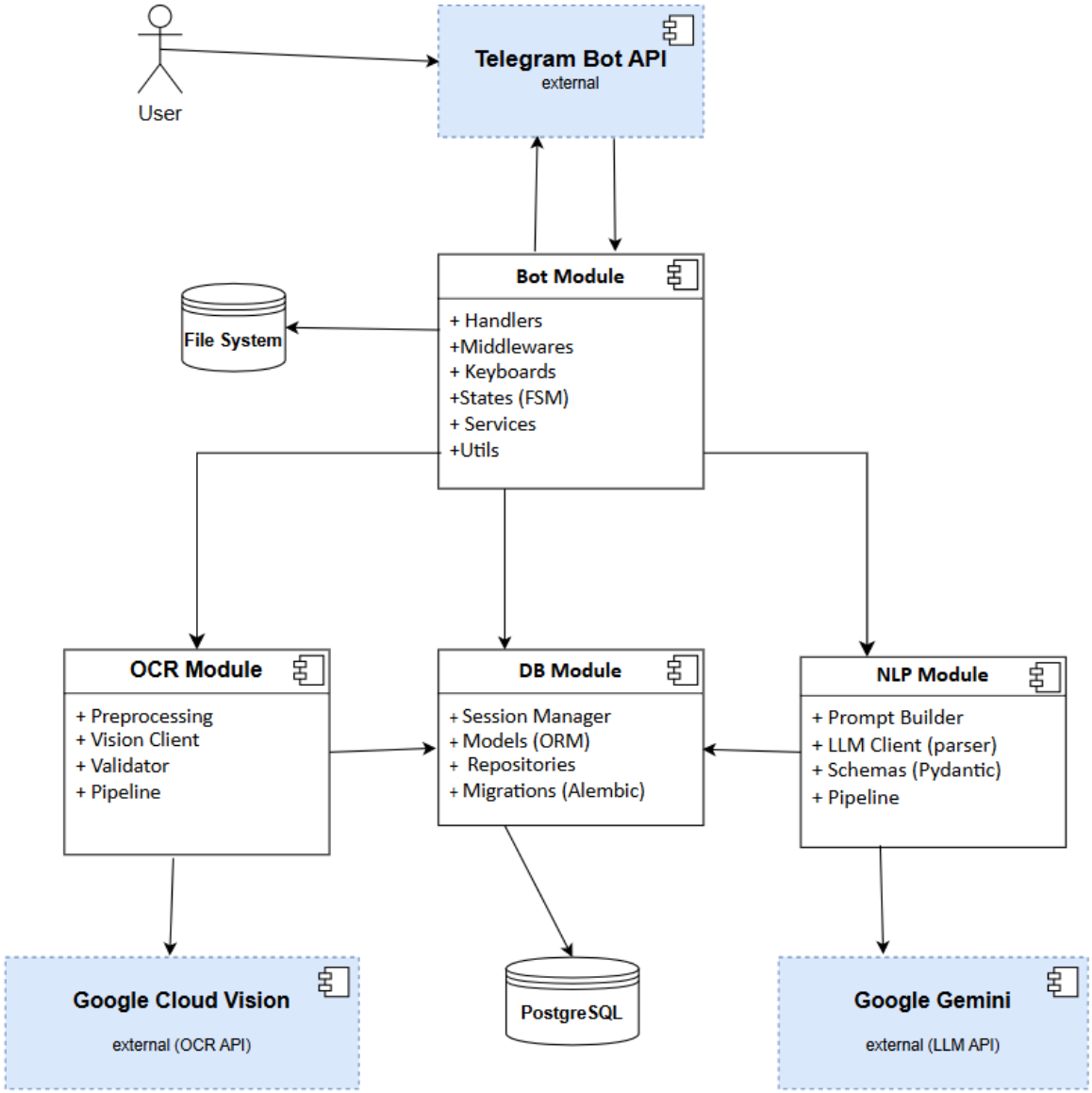


Рисунок 2.2 – Загальна архітектурна схема компонентів системи оцифрування кулінарних рецептів

2.3.2 Моделювання процесу оцифрування рукописного тексту

Процес оцифрування рукописного тексту є ключовим етапом роботи бота, який перетворює фотографію рукописного рецепту на машиночитаний текст. Цей процес спроєктований як послідовний конвеєр, що складається з кількох етапів, кожен з яких виконує окрему функцію та може завершитися як успішно, так і з помилкою, що потребує відповідної реакції системи.

Процес починається з дії користувача, а саме надсилання фотографії рукописного рецепту в чат з ботом. Модуль Bot приймає зображення, зберігає його у файловій системі та обчислює хеш зображення для виявлення дублікатів.

Якщо фото з таким хешем вже існує в базі даних поточного користувача, система не створює новий запис, а пропонує переглянути вже наявний рецепт. Якщо фото нове, створюється запис рецепта зі статусом очікування, і система автоматично переходить до етапу розпізнавання тексту без додаткового підтвердження з боку користувача. Запропонована логіка взаємодії скорочує кількість кроків взаємодії та забезпечує швидший відгук системи.

Першим етапом конвеєра є попереднє оброблення зображення для підвищення якості розпізнавання, зокрема нормалізація кольору, роздільної здатності та орієнтації тексту. Це компенсує типові недоліки мобільних знімків: низький контраст, завелику або замалу роздільну здатність та нахил аркуша при фотографуванні.

Другим етапом є надсилання обробленого зображення до хмарного сервісу розпізнавання тексту. Сервіс повертає розпізнаний текст із метаданими, зокрема рівнем впевненості розпізнавання та визначеною мовою тексту.

Третім етапом конвеєра є контроль отриманих даних, що передбачає дві послідовні перевірки. Перша з них визначає фізичну наявність розпізнаного тексту. Якщо за результатами OCR-аналізу текст виявився порожнім або занадто коротким, процес опрацювання припиняється. У такому разі користувач отримує рекомендацію надіслати якісніше фото, а тимчасовий запис і файл зображення видаляються із системи.

Друга перевірка визначає належність тексту до кулінарної тематики. Алгоритм аналізує наявність семантичних маркерів різного ступеня специфічності – від кулінарних дій, які однозначно вказують на рецепт, до загальних назв продуктів, які можуть зустрічатися і в інших контекстах. Логічну структуру алгоритму наведено на рисунку А.1.

Якщо текст не містить достатньої кількості таких ознак, система не відхиляє його безповоротно, а запитує підтвердження у користувача.

Користувач отримує повідомлення про те, що текст не містить достатньо ознак кулінарного рецепту, та має можливість підтвердити продовження оброблення або скасувати операцію. Такий підхід дозволяє уникнути хибних відхилень для нетипових рецептів, які можуть не містити характерних кулінарних термінів, водночас зберігаючи захист від випадкового завантаження нерелевантних зображень. У разі скасування відповідні записи в базі даних та файл зображення видаляються із системи.

Якщо обидві перевірки пройдені успішно, результати OCR зберігаються в базі даних: розпізнаний текст, рівень впевненості розпізнавання, визначена мова тексту, а також метадані від хмарного сервісу. Статус рецепту оновлюється на «OCR_DONE», після чого система автоматично передає результат до NLP-конвеєра для структуризації рецепту.

На будь-якому етапі конвеєра може виникнути непередбачена помилка, наприклад, недоступність зовнішнього сервісу чи проблеми з мережею, некоректний формат відповіді. У такому випадку статус рецепту змінюється на «FAILED» із збереженням причини помилки, а користувач отримує зрозуміле повідомлення замість технічної інформації про виняток.

На рисунку Б.1 представлено діаграму діяльності, яка відображає повний процес оцифрування рукописного тексту від надсилання фотографії користувачем до збереження результатів OCR.

2.3.3 Моделювання процесу структуризації рецепту засобами NLP

Процес структуризації є другим ключовим етапом оброблення рецепту, під час якого суцільний текстовий блок, отриманий на етапі OCR, перетворюється на впорядковану структуру з виділеними семантичними компонентами рецепту.

Для вирішення цього завдання обрано підхід на основі великої мовної моделі, яка здатна інтерпретувати текст довільної структури та виділяти з нього смислові елементи без попередньо заданих жорстких правил. Процес структуризації запускається автоматично після успішного завершення етапу OCR як наступний крок єдиного конвеєра оброблення. Статус рецепту змінюється на «NLP_PROCESSING», і управління передається NLP-конвеєру.

Першим етапом є формування текстової інструкції для мовної моделі. Цей запит складається з двох частин: системних вказівок та користувацького повідомлення. Системна інструкція визначає роль моделі та правила розбору рецепту, які саме поля потрібно виділити, в якому форматі повернути результат, як обробляти неоднозначності та відсутні дані. Користувацьке повідомлення містить безпосередньо розпізнаний текст рецепту. Це рішення забезпечує стабільність формату відповіді незалежно від змісту конкретного рецепту. Модель має виділити з вхідного тексту наступні семантичні компоненти: назву страви; опис страви; перелік інгредієнтів із зазначенням назви, кількості та одиниці виміру для кожного; покрокові інструкції приготування з можливим зазначенням тривалості кожного кроку; загальний час приготування; кількість порцій; тип кухні; рівень складності; категорії страви; теги для пошуку. Результат повертається у форматі структурованого JSON, що забезпечує однозначний автоматичний розбір відповіді.

Другим етапом є виклик мовної моделі через API з передачею сформованої інструкції. На цьому етапі передбачено механізм повторних спроб при тимчасовій недоступності сервісу мовної моделі. Якщо сервіс повертає помилку перевантаження, система автоматично повторює запит зі зростаючими інтервалами очікування. Кількість повторних спроб обмежена, і якщо жодна спроба не є успішною, процес завершується з помилкою.

Третім етапом є перевірка структури отриманої відповіді. Отриманий JSON додатково проходить верифікацію за допомогою схем даних, що гарантує відповідність структури очікуваному формату. Схема описує очікувані типи полів, обов’язкові та опціональні атрибути, допустимі діапазони значень.

Наприклад, кількість інгредієнтів має бути числовим значенням, час приготування є додатним цілим числом у хвилинах, а рівень складності – одним зі значень визначеного переліку. Якщо відповідь моделі не проходить цю перевірку, процес завершується з помилкою, оскільки параметр формату на рівні API має гарантувати коректний JSON, а невідповідність схемі свідчить про принципову неможливість розбору конкретного тексту.

Після успішної перевірки структури результат зберігається в базі даних як проміжний результат NLP-оброблення із зазначенням використаної моделі та метаданих запиту. Статус рецепту змінюється на «PARSED», і користувачу надсилається повідомлення з відформатованим результатом структуризації. На завершальному етапі користувачу надається можливість верифікувати результат та обрати подальший сценарій дій, зокрема збереження, редагування чи скасування рецепта.

Варто зауважити, що перелік інгредієнтів та інструкції зберігаються у форматі, сформованому мовною моделлю, без можливості атомарного редагування через інтерфейс бота. Для їх зміни передбачено механізм інтелектуальної корекції, що надає можливість описати бажані зміни природною мовою, після чого мовна модель автоматично вносить відповідні правки до структурованого рецепту. Результат кожної корекції зберігається в базі даних як окремий запис із зазначенням коментаря користувача та попереднього стану рецепту, що забезпечує повну історію змін.

У разі підтвердження збереження статус рецепту оновлюється на «VERIFIED», що робить запис доступним у загальній колекції користувача.При виборі команди скасування всі тимчасові дані рецепту безповоротно видаляються із системи. Аналогічно до OCR-процесу, на будь-якому етапі NLP-конвеєра може виникнути непередбачена помилка, наприклад, недоступність мовної моделі або перевищення ліміту запитів. У такому випадку статус рецепта змінюється на «FAILED» із збереженням причини помилки, а користувач отримує зрозуміле повідомлення.

Вибір підходу на основі мовної моделі замість класичних методів NLP (регулярних виразів, аналізаторів на основі правил) обумовлений характером вхідних даних. Рукописні рецепти мають довільну структуру: автори використовують різну послідовність елементів, нестандартні скорочення, неформальну лексику, змішують мови. Мовна модель здатна інтерпретувати такі тексти контекстуально, тоді як підхід на основі жорстко заданих правил розбору потребував би створення великої кількості шаблонів для кожного варіанту оформлення, що є непрактичним і ненадійним. На рисунку В.1 представлено діаграму діяльності, яка відображає повний процес структуризації рецепта від отримання розпізнаного тексту до збереження або відхилення результату користувачем.

2.3.4 Моделювання взаємодії користувача з ботом

Для визначення функціональних меж системи та формалізації вимог до поведінки бота виконано моделювання сценаріїв взаємодії користувача з системою за допомогою діаграми варіантів використання. Кожен варіант використання описує завершений функціональний процес з точки зору користувача та визначає очікувану поведінку системи.

Єдиним актором системи є користувач месенджера Telegram. Реєстрація в системі має бути максимально спрощеною, тобто без окремих форм авторизації чи введення персональних даних. Тому обрано підхід автоматичної реєстрації при першому зверненні до бота: система створює обліковий запис на основі Telegram-ідентифікатора, що є унікальним для кожного користувача.

Цей сценарій є передумовою для всіх інших, бо без початкової реєстрації решта функцій має бути недоступною. Центральним варіантом використання є додавання нового рецепту. Користувач надсилає фотографію рукописного рецепту, після чого система автоматично виконує послідовність оброблення: збереження зображення, перевірку на дублікати, розпізнавання тексту та структуризацію.

На кожному етапі необхідно забезпечити зворотний зв’язок – користувач має бачити поточний стан оброблення та мати можливість впливати на результат. Детальне моделювання цього процесу наведено у підрозділах 2.3.2 та 2.3.3.

Робота зі збереженими рецептами потребує кількох способів навігації: перегляд усієї колекції, фільтрація за категоріями, тегами, рівнем складності та типом кухні. Оскільки колекція може містити значну кількість записів, необхідно передбачити механізм пагінації, а саме посторінкового перегляду списків. При виборі фільтра система має показувати доступні значення з кількістю рецептів для кожного, щоб користувач міг оцінити розподіл своєї колекції перед вибором.

Перегляд конкретного рецепта має відображати всі його деталі: назву, опис, інгредієнти з кількістю та одиницями виміру, покрокові інструкції, час приготування, кількість порцій, категорії та теги. Окремо передбачено можливість перегляду оригінального фото рукопису, що дає змогу користувачу звірити структурований результат з першоджерелом та зберігає відчуття автентичності сімейного запису.

Окремим варіантом використання є пошук рецептів. Цей сценарій потребує переведення бота в режим очікування текстового введення. При проєктуванні пошуку визначено потребу у двох стратегіях: повнотекстовий пошук за назвою рецепту та нечіткий пошук за назвами інгредієнтів. Об’єднання результатів обох стратегій з видаленням дублікатів забезпечує ширше покриття пошукових запитів. Нечіткий пошук є особливо важливим для системи, що працює з рукописними рецептами, адже назви інгредієнтів можуть бути розпізнані з незначними помилками. На будь-якому етапі користувач може скасувати пошук.

Редагування рецепту спроєктовано через два механізми внесення змін. Перший – це ручне редагування окремих полів (назви, опису, часу приготування, кількості порцій, рівня складності, категорії та тегів). Кожне поле потребує окремого діалогу з перевіркою введеного значення. Для полів з

обмеженим набором значень (рівень складності, категорії) доцільно запропонувати вибір із готового переліку, а для полів з довільним текстом – введення у вільній формі. Другий механізм передбачає інтелектуальну корекцію, яка дозволяє описати бажані зміни текстовим коментарем. Цей підхід є особливо корисним для модифікації інгредієнтів та кроків приготування, які не мають окремого інтерфейсу для атомарного редагування.

Операція видалення рецепту має включати етап підтвердження для запобігання випадковій втраті даних. Система має запитувати явне підтвердження перед виконанням незворотної операції.

Експорт рецепту передбачає генерацію документа у форматі PDF з повним оформленням рецепту: назвою, інгредієнтами, покроковими інструкціями та метаданими. Згенерований документ надсилається користувачу безпосередньо в чат як файл для завантаження, що дозволяє зберегти рецепт на пристрої, роздрукувати або надіслати іншим користувачам.

Процес отримання довідки супроводжується наданням користувачу впорядкованого опису всіх доступних команд та можливостей бота, що дозволяє швидко зорієнтуватися у функціональності системи.

При проєктуванні взаємодії також визначено необхідність коректного оброблення невідомих повідомлень. Якщо користувач надсилає дані, які система не може інтерпретувати – текст без активного стану очікування, стікер, документ, голосове повідомлення – бот має надіслати пояснення щодо підтримуваних типів вхідних даних та запропонувати скористатися довідкою.

Для навігації можна обрати підхід на основі інтерактивних меню з кнопками вибору. Це рішення обумовлено тим, що користувачу не потрібно запам’ятовувати текстові команди, забезпечується можливість багаторівневої навігації з поверненням на попередній рівень, а кнопки мінімізують помилки введення.

На рисунку Г.1 представлено діаграму варіантів використання, яка відображає повний набір функціональних сценаріїв взаємодії користувача з ботом.

3 РОЗРОБЛЕННЯ МУЛЬТИМОДАЛЬНОГО БОТА В TELEGRAM ДЛЯ ОЦИФРУВАННЯ ТА СИСТЕМАТИЗАЦІЇ РУКОПИСНИХ КУЛІНАРНИХ РЕЦЕПТІВ

3.1 Вибір інструментальних засобів для реалізації поставленої задачі

Базовою мовою програмування обрано Python версії 3.12 з огляду на розвинену екосистему бібліотек для роботи зі ШІ, вбудовану підтримку асинхронного програмування через конструкції async/await та сучасну систему анотацій типів [37]. Розроблення системи виконано у середовищі PyCharm Professional.

Взаємодія з месенджером Telegram реалізована за допомогою фреймворку Aiogram версії 3, спроєктованого виключно для асинхронної роботи. Асинхронність є принциповою вимогою, оскільки кожен запит користувача породжує ланцюг звернень до зовнішніх сервісів – і блокування на будь-якому з них зупинило б оброблення інших повідомлень. Альтернативний фреймворк python-telegram-bot підтримує як синхронний, так і асинхронний режими, проте саме орієнтація Aiogram виключно на асинхронну модель забезпечує єдиний архітектурний підхід без змішування парадигм [38]. Фреймворк надає систему маршрутизаторів для групування обробників, вбудований автомат зі скінченною множиною станів (Finite State Machine, FSM) для реалізації діалогових сценаріїв та механізм проміжних обробників для наскрізних операцій, таких як журналювання та впровадження сесій.

Для оптичного розпізнавання рукописного тексту обрано хмарний сервіс Google Cloud Vision API, який поєднує кілька суттєвих переваг для поставленої задачі: підтримку розпізнавання рукописного тексту через метод document_text_detection, можливість передачі мовної підказки (параметр language_hints) для підвищення точності на кириличних текстах, повернення ієрархічної структури результатів із рівнем впевненості для кожного розпізнаного символу та автоматичне визначення мови документа [39].

Під час попередніх експериментів сервіс продемонстрував прийнятну точність на українському рукописі. Додатковим фактором стала наявність безкоштовного тарифного плану з лімітом у 1000 запитів на місяць, що покрив потреби етапу розроблення. Попереднє оброблення зображень перед надсиланням до сервісу реалізовано засобами бібліотеки OpenCV, яка надає необхідні функції конвертації колірних просторів, масштабування та геометричних перетворень [1].

Структуризацію розпізнаного тексту, тобто перетворення суцільного тексту рецепта на JSON-об’єкт із полями назви, інгредієнтів та кроків, покладено на велику мовну модель Gemini 2.5 Flash від Google. Вибір цієї моделі обумовлено двома факторами. По-перше, параметр response_mime_type зі значенням application/json на рівні інтерфейсу програмування гарантує повернення результату виключно у форматі JSON без супровідного тексту чи додаткового форматування, що усуває потребу у складному розборі неструктурованої відповіді [40].

По-друге, варіант Flash оптимізовано під швидкість виконання, що відповідає характеру задачі витягування полів із кулінарного тексту. Платформа Google AI Studio на момент реалізації надавала безкоштовний тарифний план із достатніми лімітами запитів, тоді як альтернативні моделі зіставної якості передбачали платну тарифікацію з першого запиту. Контроль структури відповідей моделі забезпечує бібліотека Pydantic, яка дозволяє декларативно описати очікувану схему даних з автоматичною перевіркою типів, обов’язкових полів та допустимих діапазонів значень.

Системою управління базами даних обрано PostgreSQL [41]. Таке рішення зумовлене конкретними вимогами проєкту до пошукових можливостей. Вбудований повнотекстовий пошук на основі типу TSVECTOR із GIN-індексами забезпечує пошук рецептів за назвою з урахуванням морфології української мови. Розширення pg_trgm реалізує нечіткий пошук на основі триграм, що є критичним для забезпечення стійкості до помилок OCR-розпізнавання.

Тип даних JSONB дозволяє зберігати напівструктуровані відповіді зовнішніх сервісів без потреби проєктувати під них окремі таблиці. Серед розглянутих альтернатив SQLite не забезпечує повнотекстового та нечіткого пошуку на необхідному рівні, а документні СУБД не гарантують цілісності посилань між зв’язаними сутностями [42].

Взаємодія з базою даних на рівні застосунку відбувається через засоби об’єктно-реляційного відображення (Object-Relational Mapping, ORM) бібліотеки SQLAlchemy в парі з асинхронним драйвером asyncpg, який виконує запити без блокування циклу оброблення повідомлень. Міграціями схеми бази даних керує бібліотека Alembic. Формування PDF-документів при експорті рецептів покладено на бібліотеку ReportLab із підключенням шрифту DejaVu Sans для підтримки кирилиці.

3.2 Етапи розроблення мультимодального бота в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP

3.2.1 Реалізація архітектури проєкту та бази даних

Проєкт побудовано за модульним принципом: кореневий каталог містить п’ять пакетів, кожен з яких відповідає за окремий функціональний домен. Загальну структуру наведено на рисунку 3.1.

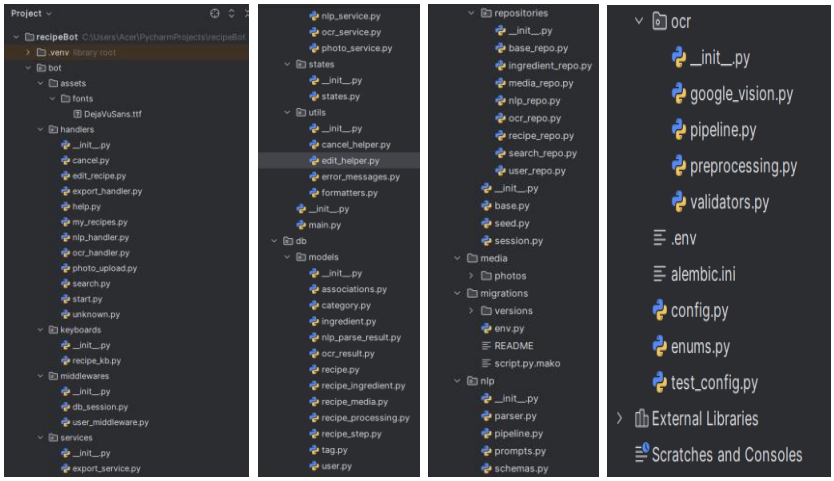


Рисунок 3.1 – Структура файлів проєкту

Центральним є пакет bot/, організований за принципом розділення відповідальності. Усередині виділено шість підпакетів (табл. 3.1).

Таблиця 3.1 – Структура та призначення підпакетів модуля bot/

Підпакет	Призначення
handlers/	Обробники повідомлень і зворотних викликів (11 модулів)
services/	Сервісний шар – координація між handlers та зовнішніми модулями
keyboards/	Вбудовані-клавіатури
middlewares/	Проміжні обробники (сесія БД, автентифікація користувача)
states/	Стани скінченного автомата для діалогів
utils/	Форматування, повідомлення про помилки, допоміжні функції

Сервісний шар (services/) координує взаємодію з трьома зовнішніми залежностями: Google Vision, Gemini та Telegram API, і зосереджує логіку повторних спроб оброблення, верифікації та збереження даних. Завдяки цьому обробники у handlers/ містять лише логіку взаємодії з користувачем: приймання подій, формування відповідей та навігацію між екранами. Стани FSM для діалогів зосереджено в одному файлі states/states.py, а вбудовані клавіатури – у keyboards/recipe_kb.py, що забезпечує їх перевикористання в кількох обробниках.

Пакети ocr/, nlp/ та db/ реалізують три основні підсистеми: розпізнавання тексту, його структуризацію та зберігання даних відповідно. Внутрішню будову ocr/ та nlp/ розглянуто далі у цьому розділі.

Конфігурацію системи зосереджено у кореневому каталозі: config.py зчитує параметри зі змінних оточення через pydantic-settings з автоматичною перевіркою відповідності типів, enums.py визначає замкнуті набори значень для статусів оброблення та рівнів складності, а файл .env зберігає конфіденційні дані: токен бота, параметри підключення та ключі зовнішніх сервісів. Файл alembic.ini налаштовує інструмент міграцій.

Взаємодію з Telegram Bot API реалізовано через механізм long polling. На відміну від альтернативного підходу на основі вебхуків (webhook), який потребує публічної IP-адреси, налаштованого SSL-сертифіката та вебсервера для приймання вхідних HTTP-запитів від Telegram, long polling працює у зворотному напрямку: бот самостійно надсилає періодичні запити до серверів Telegram методом getUpdates та отримує нові повідомлення у відповіді. Це спрощує розгортання, оскільки бот може працювати в межах приватних мереж або на локальному сервері без додаткової інфраструктури. Бібліотека Aiogram абстрагує цей механізм у методі start_polling, який автоматично керує циклом запитів, обробляє перевищення лімітів очікування та повторює з'єднання у разі тимчасових мережевих збоїв. Для цільового навантаження системи, розрахованої на індивідуальне використання, пропускної здатності long polling достатньо, а перехід на вебхуки залишається можливим без зміни логіки обробників.

Точкою входу є bot/main.py (рис. Д.1). Порядок реєстрації маршрутизаторів у ньому принциповий: Aiogram перевіряє обробники послідовно, тому команди зареєстровано першими, обробник /cancel розміщено перед обробниками станів FSM, а unknown.py останній для перехоплення нерозпізнаних повідомлень. Функція on_startup виконується одноразово при запуску та забезпечує початкове наповнення довідникових таблиць.

Реалізацію бази даних виконано відповідно до структури, спроєктованої у підрозділі 2.2. Пакет db/ містить базовий допоміжний клас-домішок (mixin), ORM-моделі сутностей, репозиторії для доступу до даних та конфігурацію підключення.

У файлі base.py визначено два класи-домішки (рис. Д.2): UUIDPrimaryKeyMixin додає первинний ключ типу UUID, що генерується на стороні застосунку до виконання INSERT-запиту, а TimestampMixin – часові мітки створення та оновлення, які генеруються на стороні PostgreSQL функцією NOW().

Усі чотирнадцять сутностей реалізовано як ORM-моделі в підпакеті db/models/. Центральною є модель рецепту (рис. 3.2), більшість полів якої визначено як необов’язкові, оскільки рукописні рецепти рідко містять повну метаінформацію. Зовнішній ключ user_id з правилом ondelete="CASCADE" забезпечує автоматичне видалення рецептів у разі видалення облікового запису, а параметр index=True на полях user_id та cuisine створює B-Tree індекси для найчастіших сценаріїв фільтрації.

Рисунок 3.2 – ORM-модель рецепту (db/models/recipe.py)

Для зберігання напівструктурованих даних від зовнішніх сервісів застосовано тип JSONB: у моделі OCRResult поле ocr_metadata зберігає повну відповідь сервісу розпізнавання, а у NLPParseResult поле raw_llm_response – відповідь мовної моделі. Зв’язки типу «багато-до-багатьох» між рецептами, категоріями та тегами реалізовано через зв’язкові таблиці з композитними первинними ключами. Правила цілісності продубльовано на рівні ORM та на рівні схеми бази даних.

Доступ до даних абстраговано через патерн Repository. RecipeRepository забезпечує повний набір операцій з рецептами, а SearchRepository поєднує дві стратегії пошуку: повнотекстовий за назвою через TSVECTOR із GIN-індексом та нечіткий за інгредієнтами на основі розширення pg_trgm (рис. 3.3).

Функція similarity() порівнює рядки за збігом трисимвольних підрядків із пороговим рівнем 0,3, а пошук виконується одночасно за канонічними назвами та синонімами з подальшим усуненням дублікатів.

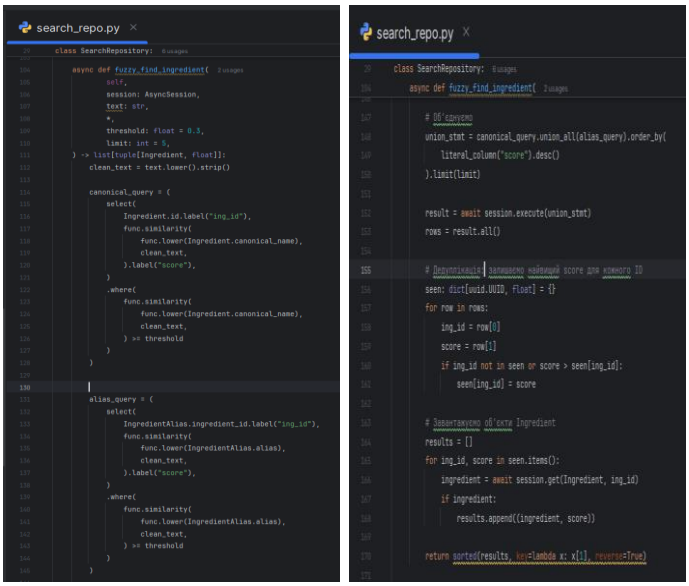


Рисунок 3.3 – Нечіткий пошук інгредієнтів (db/repositories/search_repo.py)

Підключення до бази даних реалізовано через асинхронний набір з’єднань у session.py (рис. Д.3) з десятьма постійними з’єднаннями та можливістю створення до двадцяти додаткових.

Автоматичне створення сесії для кожного запиту забезпечує проміжний обробник DbSessionMiddleware (рис. Д.4) через механізм впровадження залежностей Aiogram. Конструкція async with гарантує закриття сесії навіть при винятковій ситуації. Наступним виконується UserMiddleware, який знаходить користувача за ідентифікатором Telegram та передає його в обробник.

Для управління міграціями схеми застосовано Alembic, кожна міграція якого генерується автоматично на основі порівняння поточного стану моделей зі схемою бази. Початкове наповнення довідникових таблиць реалізовано у seed.py з використанням конструкції on_conflict_do_nothing, що гарантує ідемпотентність при повторних запусках (рис. Д.5). Довідник категорій містить дев’ять значень, а довідник синонімів інгредієнтів – понад п’ятдесят канонічних назв з регіональними варіантами (рис. Е.1).

3.2.2 Реалізація OCR-конвеєра

Конвеєр оптичного розпізнавання тексту реалізовано у пакеті ocr/ та координовано через сервісний шар bot/services/ocr_service.py. Аналіз зображення проходить три послідовні етапи: попереднє оброблення, розпізнавання хмарним сервісом та семантичну перевірку результату.

Першим етапом є попереднє оброблення зображення, яке реалізовано у модулі preprocessing.py (рис. 3.4).

preprocessing.py

```python
import cv2
import numpy as np

def to_grayscale(image: np.ndarray) -> np.ndarray:  1 usage
    if len(image.shape) == 2:
        return image
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

def resize_image(image: np.ndarray, max_width: int = 1800) -> np.ndarray:  1 usage
    h, w = image.shape[:2]
    if w <= max_width:
        return image

    scale = max_width / w
    new_w = max_width
    new_h = int(h * scale)
    return cv2.resize(image,  dsize: (new_w, new_h), interpolation=cv2.INTER_AREA)

def deskew(image: np.ndarray) -> np.ndarray:  1 usage
    thresh = cv2.threshold(
        image,  thresh: 0,  maxval: 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU
    )[1]
    coords = np.column_stack(np.where(thresh > 0))

    if len(coords) < 50:
        return image
    angle = cv2.minAreaRect(coords)[-1]

    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    if abs(angle) < 0.5 or abs(angle) > 15:
        return image

    h, w = image.shape[:2]
    center = (w // 2, h // 2)
    matrix = cv2.getRotationMatrix2D(center, angle,  scale: 1.0)
    rotated = cv2.warpAffine(
        image, matrix,  dsize: (w, h),
        flags=cv2.INTER_CUBIC,
        borderMode=cv2.BORDER_REPLICATE,
    )
    return rotated


def preprocess(image: np.ndarray) -> np.ndarray:  2 usages
    """
    Головна точка входу для препроцесингу зображень.
    Виконує послідовність: Grayscale -> Resize -> Deskew.
    """
    result = to_grayscale(image)
    result = resize_image(result, max_width=1800)
    result = deskew(result)
    return result
```

Рисунок 3.4 – Конвеєр попереднього оброблення зображення (ocr/preprocessing.py)

Конвеєр містить три операції:

– перетворення у градації сірого: видалення кольорової інформації та зменшення обсягу даних у три рази;

– масштабування до ширини 1800 пікселів зі збереженням пропорцій, тобто приведення зображення до діапазону 1024-2048 пікселів, рекомендованого документацією Google Cloud Vision API. Для зменшення роздільної здатності застосовано інтерполяцію Area, яка усереднює значення пікселів та забезпечує найвищу якість при зниженні роздільної здатності;

– корекція нахилу: визначення кута повороту тексту через мінімальний обмежуючий прямокутник текстових пікселів та його компенсація. Встановлено два захисних обмеження: нахил менший за 0,5 градусів ігнорується як незначний, а нахил більше 15 градусів відхиляється як ймовірне хибне спрацьовування на зображеннях зі складною геометрією.

За взаємодію з Google Cloud Vision API відповідає модуль google_vision.py (рис. 3.5).

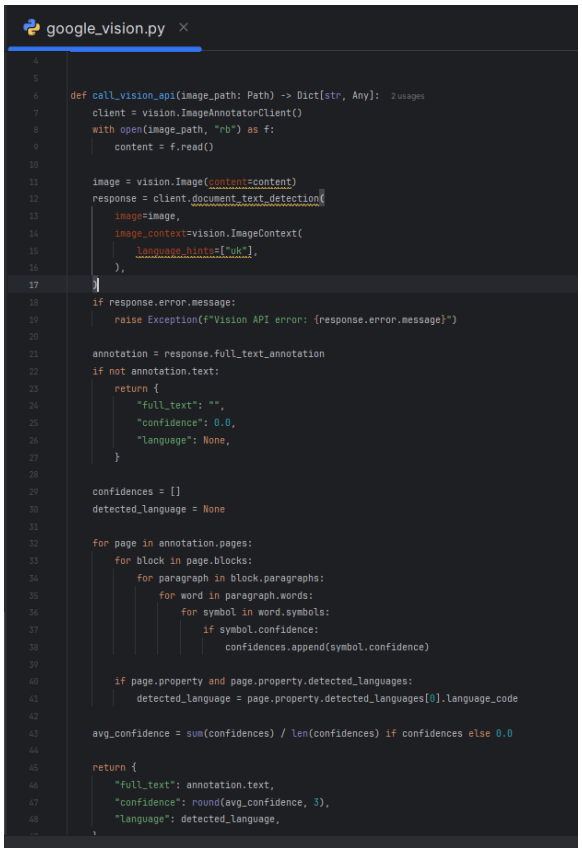


Рисунок 3.5 – Взаємодія з Google Cloud Vision API (ocr/google_vision.py)

Розпізнавання виконується методом document_text_detection, оптимізованим для щільних документів та рукописного тексту. Параметр language_hints зі значенням uk передає сервісу контекстну підказку щодо очікуваної мови, що підвищує точність оброблення кирилиці.

Сервіс повертає ієрархічну структуру: сторінки, блоки, параграфи, слова, символи. На основі рівня впевненості кожного символу обчислюється середня впевненість для всього зображення. Це значення зберігається у моделі OCRResult як діагностичний показник якості розпізнавання.

Послідовність етапів координує pipeline.py: завантаження зображення з файлової системи, передача через попереднє оброблення, збереження результату у тимчасовий файл для хмарного сервісу та його видалення після отримання відповіді. Розпізнаний текст передається до модуля перевірки.

Семантичний контроль реалізовано у validators.py у два етапи. Насамперед перевіряється фізична наявність тексту: порожній результат або менше п’яти слів класифікується як некоректні вхідні дані. Далі визначається належність тексту до кулінарної тематики через алгоритм категоризованого аналізу на основі стемів – початкових частин слів, які покривають усі морфологічні форми.

Стеми поділено на дві категорії за ступенем специфічності. До сильних стемів належать кулінарні дії та спеціалізований посуд, що практично не зустрічаються поза контекстом рецептів: «смажит», «варит», «нарізат», «каструл», «сковород». А слабкі – це назви продуктів та одиниці виміру, які трапляються і в інших текстах: «борошн», «молок», «цибул», «грам». Повний перелік наведено у додатку Ж (рис. Ж.1, рис. Ж.2).

Логіка прийняття рішення враховує категорію збігів: два або більше сильних стемів однозначно класифікують текст як рецепт; один сильний у поєднанні з трьома слабкими – також; наявність лише слабких стемів, незалежно від кількості, недостатня для позитивного висновку. Це рішення мінімізує хибні спрацьовування, бо слабкі стеми «грам» чи «літр» можуть траплятись і в текстах з фізики або хімії. Реалізацію алгоритму наведено на рисунку 3.6.

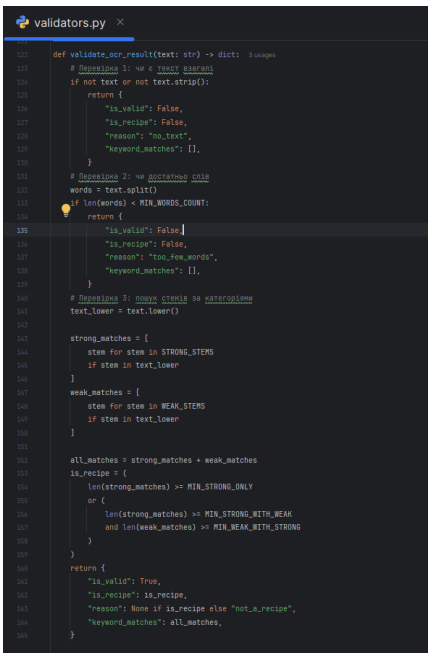


Рисунок 3.6 – Алгоритм семантичної перевірки тексту (ocr/validators.py)

У разі, коли модуль перевірки не знаходить достатньо ознак кулінарного рецепта, система запитує підтвердження у користувача перед продовженням оброблення.

Координацію між конвеєром та обробниками бота забезпечує сервісний шар ocr_service.py. При надходженні зображення сервіс змінює статус рецепта на OCR_PROCESSING, запускає конвеєр, зберігає результат кожного етапу у відповідних моделях бази даних та оновлює статус на OCR_DONE. Якщо на будь-якому етапі виникає помилка, причина фіксується в базі, а статус змінюється на FAILED, що дозволяє користувачу повторити спробу пізніше.

3.2.3 Реалізація NLP-конвеєра

Завдання трансформації неструктурованого тексту, отриманого на етапі оптичного розпізнавання, у коректно сформований JSON-об’єкт вирішується конвеєром інтелектуальної структуризації. Його програмну реалізацію розміщено в пакеті nlp/, а керування логікою взаємодії здійснюється через сервісний модуль bot/services/nlp_service.py.

Загальний алгоритм роботи охоплює чотири кроки: формування інструкцій для мовної моделі, звернення до програмного інтерфейсу з отриманням відповіді, перевірку коректності структури засобами бібліотеки Pydantic та фінальне збереження даних у базі.

Якість структуризації безпосередньо залежить від формулювання системної інструкції, зосередженої у модулі prompts.py. Інструкція визначає роль моделі як експерта з розпізнавання кулінарних рецептів українською мовою та містить три групи правил (повний текст наведено у лістингу И.1).

Перша група – це загальні правила структуризації, які поділяються на два блоки за принципом роботи з інформацією. Блок витягування зобов'язує модель використовувати виключно інформацію з вхідного тексту без додавання власних припущень, встановлювати порожнє значення для полів, які неможливо визначити, послідовно нумерувати кроки, витягувати часові параметри та доповнювати перелік інгредієнтів тими, що згадуються лише у кроках приготування.

Блок виведення даних дозволяє моделі визначати за контекстом категорію страви з фіксованого переліку, кухню, теги та рівень складності. Складність визначається за кількісними критеріями: кількістю інгредієнтів, кількістю кроків та наявністю складних технік, причому за розбіжності між показниками обирається вищий рівень. Модель також формує короткий опис страви та оцінку впевненості у коректності результату в діапазоні від 0,0 до 1,0.

Друга група правил регламентує оброблення інгредієнтів: нормалізацію назв до називного відмінку однини, розділення кількості та одиниць виміру, перетворення діапазонів на середнє значення та збереження нестандартних кулінарних мір («пучок», «жменя», «зубчик»).

Третя група визначає правила оброблення кроків. Вона зобов'язує модель виправляти помилки OCR-розпізнавання, зберігаючи оригінальний зміст. В інструкцію вбудовано правила корекції типових підмін кирилиці: модель отримує приклади («mа» → «та», «Нa» → «На», «слію» → «олію») та вказівку спиратися на контекст речення для їх виправлення.

Очікувану структуру відповіді описано у модулі schemas.py (рис. 3.7). Коренева модель ParsedRecipe містить тринадцять полів. Більшість полів визначено як необов'язкові з типом Optional, оскільки рукописні рецепти рідко містять повну метаінформацію. Кожен інгредієнт описано вкладеною моделлю IngredientItem, а кожен крок – моделлю StepItem.

Для забезпечення коректності даних на рівні схеми реалізовано групи функцій перевірки. Методи приведення типів (round_integers, coerce_quantity) обробляють ситуації з некоректними дробовими значеннями або розділювачами. Функції контролю допустимих значень нормалізують рівень складності та обмежують оцінку впевненості. Перевірка наявності хоча б одного інгредієнта та кроку генерує попередження у журналі подій без відхилення результату, делегуючи виправлення механізму самокорекції.

```
class ParsedRecipe(BaseModel):  10 usages
    """Повний структурований рецепт від LLM."""
    title: str = Field(min_length=1, description="Назва страви")
    description: str | None = Field(
        default=None,
        description="Короткий опис страви",
    )
    servings: int | None = Field(
        default=None,
        description="Кількість порцій",
    )
    prep_time_minutes: int | None = Field(
        default=None,
        description="Час підготовки у хвилинах",
    )
    cook_time_minutes: int | None = Field(
        default=None,
        description="Час приготування у хвилинах",
    )
    difficulty: str | None = Field(
        default=None,
        description="Складність: легко, середньо, складно",
    )
    cuisine: str | None = Field(
        default=None,
        description="Кухня: українська, італійська тощо",
    )
    category: str | None = Field(
        default=None,
        description="Категорія страви",
    )
    tags: list[str] = Field(
        default_factory=list,
        description="Теги",
    )
    ingredients: list[IngredientItem] = Field(
        default_factory=list,
        description="Список інгредієнтів",
    )
    steps: list[StepItem] = Field(
        default_factory=list,
        description="Кроки приготування",
    )
    confidence: float = Field(
        default=0.0,
        ge=0.0,
        le=1.0,
        description="Оцінка впевненості парсингу (0.0–1.0)",
    )
```

Рисунок 3.7 – Pydantic-схема моделі ParsedRecipe (nlp/schemas.py)

Безпосередню взаємодію з API Google Gemini реалізовано у модулі parser.py (рис. 3.8). Запит формується із системною інструкцією, текстом рецепта та конфігурацією, де параметр температури генерації (temperature) встановлено на рівні 0,1 для забезпечення високої детермінованості та стабільності результатів. Параметр response_mime_type="application/json" гарантує повернення JSON, проте в алгоритмі аналізу відповіді додатково реалізовано три стратегії витягування тексту (пошук Markdown-блоку, пошук за фігурними дужками, сирий текст) для стійкості до варіативності відповідей.

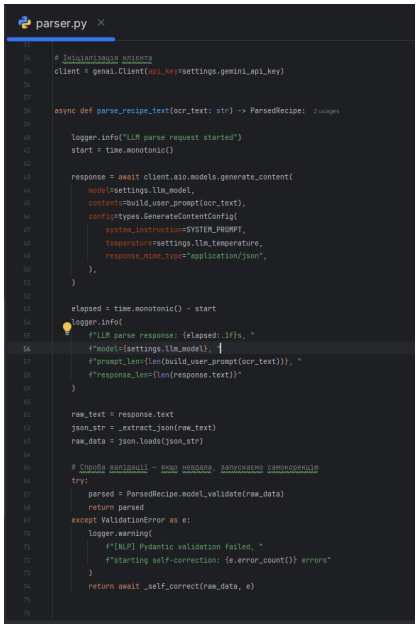


Рисунок 3.8 – Виклик мовної моделі та первинна перевірка (nlp/parser.py)

Після отримання JSON-об’єкта здійснюється його структурна перевірка. У разі виявлення невідповідностей ініціюється механізм самокорекції мовної моделі (рис. 3.9). Відповідно до результатів експериментального тестування, цикл коригування обмежено двома ітераціями: друга спроба успішно усуває більшість помилок форматування, тоді як третя не забезпечує суттєвого покращення результату. На кожному кроці виявлені недоліки трансформуються у текстовий опис та передаються моделі у вигляді спеціалізованої інструкції для самокорекції (лістинг И.2). Якщо після завершення ліміту спроб дані залишаються некоректними, алгоритм генерує виняткову ситуацію.

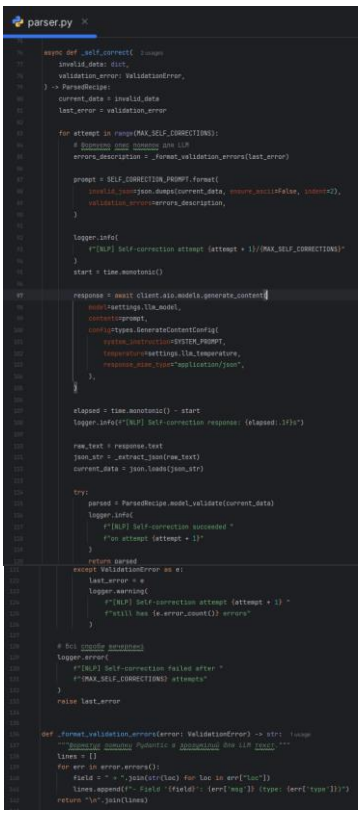


Рисунок 3.9 – Механізм самокорекції мовної моделі (nlp/parser.py)

Окрім початкової структуризації, конвеєр підтримує корекцію вже збереженого рецепта за текстовим коментарем користувача. Функція correct_recipe_text приймає поточний JSON-стан рецепта та коментар (наприклад, «видали цибулю і додай часник»). Запит корекції (лістинг И.3) інструктує модель вносити лише запитані зміни: додавання інгредієнта розширює список, видалення кроку супроводжується перенумерацією, назви нормалізуються до називного відмінка. З метою контролю витрат на використання API мовної моделі та запобігання надмірному навантаженню на систему, кількість спроб інтелектуальної корекції для одного запису обмежена двома. Результат корекції проходить ту саму перевірку і самокорекцію, що й початкова структуризація.

Повний цикл оброблення координується модулем pipeline.py. Функція process_recipe отримує текст розпізнавання з бази, змінює статус на NLP_PROCESSING, викликає мовну модель, зберігає повну відповідь з метаданими, оновлює поля рецепту, зберігає кроки, зіставляє інгредієнти з довідником синонімів, прив’язує категорію та створює або прив’язує теги.

Зіставлення інгредієнтів із довідником означає, що для кожної назви зі структурованого результату виконується пошук канонічного запису у довідниковій таблиці або створення нового, бо це забезпечує єдність найменувань і коректну роботу пошуку за синонімами. Категорія зіставляється без урахування регістру, за відсутності відповідного запису прив’язка пропускається. Теги, на відміну від категорій, створюються автоматично. Після успішного завершення статус змінюється на PARSED.

Стійкість до тимчасових збоїв зовнішніх API забезпечує сервісний шар nlp_service.py через механізм повторних спроб (рис. 3.10). Використовується алгоритм експоненційної затримки з інтервалами 2, 4 та 8 секунд. У разі помилки виконується скасування транзакції перед наступною спробою. Якщо всі чотири спроби вичерпано, статус рецепту повертається до «OCR_DONE», що дозволяє ініціювати повторне оброблення пізніше. Історія всіх звернень до великої мовної моделі (включно з корекціями типу «correction») фіксується у базі даних для відстеження послідовності змін.

Рисунок 3.10 – Програмна реалізація механізму повторних спроб структуризації (сервісний шар)

3.2.4 Реалізація інтерфейсу бота

Інтерфейсний шар зосереджено у підпакеті handlers/. Архітектуру маршрутизаторів (Router) та порядок їх реєстрації описано у підрозділі 3.2.1; тут розглянуто реалізацію конкретних сценаріїв взаємодії з користувачем.

Точкою входу для нових користувачів є обробник команди /start (модуль start.py, рис. 3.11). При першому зверненні створюється профіль у базі даних із фіксацією ідентифікатора Telegram, імені та мовних налаштувань. При повторних зверненнях система розпізнає користувача та виводить персоналізоване вітання.

start.py

```python
from aiogram import Router
from aiogram.filters import CommandStart
from aiogram.types import Message
from sqlalchemy.ext.asyncio import AsyncSession

from db.repositories.user_repo import UserRepository

router = Router(name="start")


@router.message(CommandStart())
async def cmd_start(message: Message, session: AsyncSession):
    user_repo = UserRepository()

    user, created = await user_repo.get_or_create(
        session=session,
        telegram_id=message.from_user.id,
        username=message.from_user.username,
        first_name=message.from_user.first_name,
        language_code=message.from_user.language_code or "uk",
    )
    await session.commit()

    if created:
        text = (
            f"Вітаю, <b>{user.first_name or 'друже'}</b>! 👋\n\n"
            "Я – <b>Цифровий кулінарний архів</b>.\n"
            "Допоможу вам оцифрувати рукописні рецепти.\n\n"
            "📸 Надішліть мені фото рецепту – і я збережу його для вашої кулінарної книги!"
            "💡 <i>Щоб дізнатися більше про мої можливості, натисніть /help</i>"
        )
    else:
        text = (
            f"З поверненням, <b>{user.first_name or 'друже'}</b>! 👋\n\n"
            "📸 Надішліть мені фото рецепту – і я збережу його для вашої кулінарної книги!"
            "❓ <i>Потрібна підказка? Натисніть /help для виклику довідки.</i>"
        )

    await message.answer(text)
```

Рисунок 3.11 – Обробник команди /start (handlers/start.py)

Основний сценарій – оцифрування рецепту – починається з надсилання фотографії. Модуль photo_upload.py приймає зображення та делегує виконання базових операцій сервісу photo_service.py (рис. 3.12). Функція download_photo отримує файл через API Telegram, генерує унікальне ім’я засобами модуля uuid та зберігає зображення у визначений каталог.

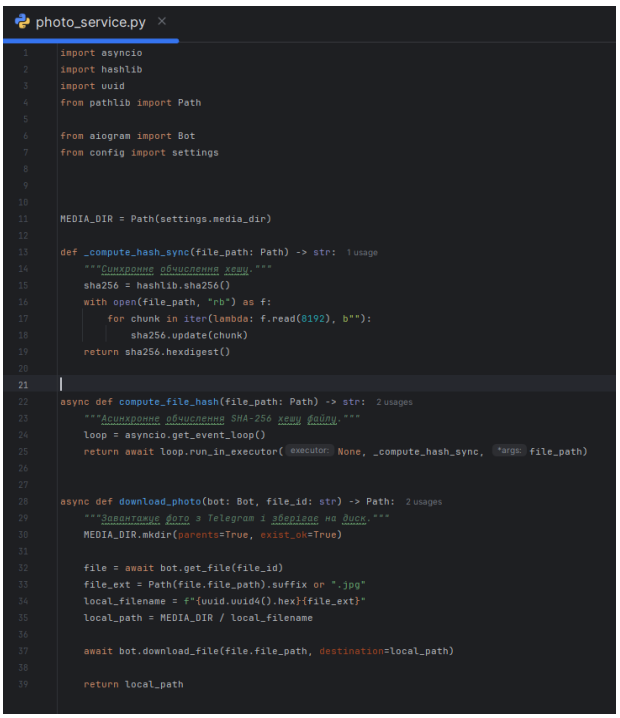


Рисунок 3.12 – Сервіс завантаження фото та обчислення хешу (bot/services/photo_service.py)

Функція compute_file_hash обчислює хеш SHA-256 в окремому потоці, щоб не блокувати асинхронний цикл оброблення повідомлень.

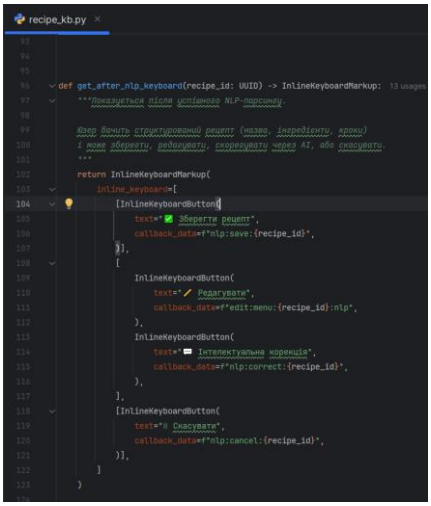
Отриманий хеш порівнюється з раніше завантаженими фотографіями поточного користувача. У разі збігу оброблення переривається: бот надсилає посилання на збережений рецепт із кнопками перегляду. Такий підхід запобігає повторному використанню лімітів запитів Google Vision та Gemini на ідентичні зображення. Якщо фотографія унікальна, автоматично запускається OCR-конвеєр.

Після завершення розпізнавання текст проходить семантичний контроль (алгоритм описано у підрозділі 3.2.3). Якщо модуль контролю не знаходить достатньо кулінарних маркерів, користувач отримує запит на підтвердження подальшого оброблення. Після підтвердження або у разі успішного проходження перевірки запускається NLP-конвеєр.

Результат структуризації відображається у вигляді попереднього перегляду з вбудованою клавіатурою. Для форматування повідомлень використовується HTML-розмітка, яку підтримує Telegram Bot API.

У модулі utils/formatters.py реалізовано дві функції форматування: format_recipe_preview для попереднього перегляду після структуризації (лаконічний стиль з відображенням оцінки впевненості моделі) та format_recipe_full для перегляду збережених рецептів (розширений стиль з візуальними позначками для часу, порцій, складності та категорій).

Меню попереднього перегляду пропонує чотири дії: збереження, інтелектуальну корекцію, ручне редагування та скасування. Кожна кнопка містить у службовому полі зворотного виклику (callback_data) префікс дії та ідентифікатор рецепту, оскільки це дозволяє обробникам однозначно визначити і операцію, і цільовий об’єкт (рис. 3.13).



Лістинг 3.13 – Програмна реалізація динамічної вбудованої клавіатури меню попереднього перегляду (bot/keyboards/recipe_kb.py)

Маршрутизацію дій після натискання кнопок виконує nlp_handler.py. Збереження фіксує статус рецепту як VERIFIED. Скасування видаляє тимчасовий запис.

Найскладнішим є сценарій інтелектуальної корекції (рис. 3.14): система переходить у стан очікування текстового коментаря, після чого текстовий коментар користувача передається мовній моделі. Кількість послідовних корекцій обмежено трьома ітераціями, після цього система пропонує зберегти поточний результат або перейти до ручного редагування.

Рисунок 3.14 – Інтелектуальна корекція рецепту (handlers/nlp_handler.py)

Ручне редагування (модуль edit_recipe.py) дозволяє точково змінювати окремі поля: назву, опис, час приготування, порції, теги. Вибір поля активує відповідний стан FSM, відповідно наступне повідомлення перезаписує значення у базі даних та повертає оновлену картку рецепту.

Команда /my_recipes (модуль my_recipes.py) відкриває меню колекції збережених рецептів. Користувач може переглянути повний список або застосувати фільтр за категорією, тегом, типом кухні чи рівнем складності. Повний список виводиться посторінково із навігаційними кнопками та лічильником поточної сторінки. Кожен фільтр спочатку показує доступні значення з кількістю рецептів, а після вибору – відфільтрований список.

Контекст обраного фільтра зберігається у стані FSM, для того щоб елемент керування «Назад» із меню перегляду рецепту повертав користувача саме до відфільтрованого списку. Детальний перегляд окремого запису містить повну інформацію та набір дій: редагування, експорт у PDF, перегляд оригінальної фотографії рукопису та видалення з підтвердженням.

Механізм експорту у PDF реалізовано у export_handler.py. Обробник завантажує дані рецепту, ініціює генерацію документа через сервісний шар (бібліотека ReportLab із підтримкою кирилиці через шрифт DejaVu Sans) та надсилає файл у чат. Тимчасовий файл видаляється одразу після відправлення. Перегляд оригінальної фотографії дозволяє порівняти результат структуризації з вихідним рукописом.

Команда /search (модуль search.py) переводить інтерфейс у стан очікування запиту (рис. 3.15). Введений текст передається до SearchRepository, який виконує комбінований пошук: повнотекстовий за назвою та нечіткий триграмний за інгредієнтами (деталі реалізації описано у підрозділі 3.2.2). Результати очищуються від дублікатів на рівні SQL-запиту і виводяться у вигляді списку з вбудованими кнопками.

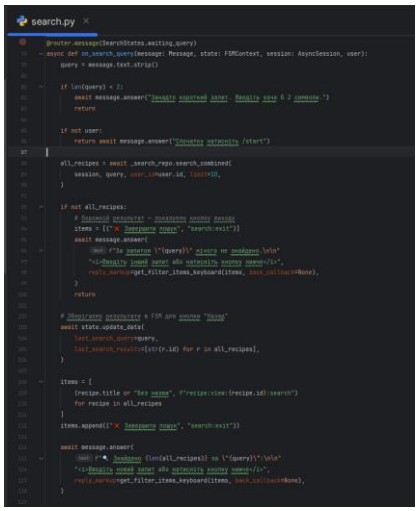


Рисунок 3.15 – Програмна реалізація обробника пошукового запиту (handlers/search.py)

Централізований обробник /cancel (модуль cancel.py) аналізує поточний стан скінченного автомата та виконує контекстне скидання: очищує стан або, якщо користувач перебував у процесі редагування, повертає інтерфейс до екрана збереженого рецепту. Обробник unknown.py перехоплює всі повідомлення, що не відповідають жодному зареєстрованому маршрутизатору, та інформує користувача про доступні команди.

3.3 Тестування розробленого мультимодального бота в Telegram та аналіз результатів

3.3.1 Тестування модуля оптичного розпізнавання тексту

Модуль оптичного розпізнавання символів є першим етапом оброблення даних у розробленому боті. Від якості його роботи безпосередньо залежить точність подальшого структурування рецепта засобами великої мовної моделі. У зв’язку з цим тестування OCR-модуля мало на меті вирішення двох завдань: визначення оптимальної комбінації методів попереднього оброблення зображень та оцінку якості розпізнавання рукописного українського тексту на репрезентативній вибірці.

Для проведення експериментів було сформовано тестову вибірку з 15 фотографій рукописних кулінарних рецептів від різних авторів. Зображення відрізнялись за типом почерку, рівнем освітлення, якістю фотографування та наявністю геометричних дефектів, що дозволило наблизити умови тестування до реальних сценаріїв експлуатації бота. Для кожного фото було підготовлено еталонний текст (ground truth) шляхом ручної транскрипції, який слугував основою для кількісного порівняння.

Оцінка якості розпізнавання здійснювалась за допомогою двох стандартних метрик. Основною метрикою обрано WER, тобто відношення мінімальної кількості операцій редагування (вставка , видалення, заміна) до загальної кількості слів в еталонному тексті. Метрика безпосередньо відображає здатність системи коректно відтворювати окремі слова, що є критичним для подальшого визначення сутностей мовною моделлю. Додатково використовувалась метрика CER, яка розраховується аналогічно, але для окремих символів, що дозволило отримати більш деталізовану оцінку відхилень.

На першому етапі було проведено порівняльний аналіз десяти варіантів конвеєра попереднього оброблення зображень.

Досліджувані методи включали: конвертацію у градації сірого, геометричне масштабування з корекцією нахилу, адаптивне перетворення, глобальну бінаризацію за методом Оцу, адаптивне підвищення контрасту (CLAHE), розмиття за Гаусом у поєднанні з бінаризацією, двосторонню фільтрацію, алгоритм нелокального усереднення, підвищення різкості, а також їх комплексну комбінацію.

Кожен метод тестувався на всіх 15 зображеннях.

Базовий рівень розпізнавання (без попереднього оброблення) становив WER = 46,1% та CER = 25,3%.

Результати порівняння зображено на рисунку 3.16.

Варіант	WER	CER	ΔWER	ΔCER	
Combo 2: Gray+Resize+Deskew	44.1%	23.0%	-2.0	-2.3	← найкращий
Combo 8: Gray+FastNlMeans	44.9%	22.6%	-1.2	-2.7	
Combo 1: Тільки Grayscale	45.0%	24.1%	-1.1	-1.2	
Оригінал (без обробки)	46.1%	25.3%	0.0	0.0	
Combo 7: Gray+Bilateral	46.6%	23.2%	+ 0.5	-2.1	
Combo 9: Gray+Sharpen	46.8%	25.7%	+ 0.7	+ 0.4	
Combo 10: Усі разом	52.2%	26.8%	+ 6.1	+ 1.5	
Combo 5: Gray+CLAHE	55.9%	29.5%	+ 9.8	+ 4.2	
Combo 4: Gray+Otsu	60.0%	41.8%	+ 13.9	+ 16.5	
Combo 6: Gray+Gauss+Otsu	68.0%	51.2%	+ 21.9	+ 25.9	
Combo 3: Gray+Adaptive	96.2%	92.1%	+ 50.1	+ 66.8	

Рисунок 3.16 – Результати порівняння метрик розпізнавання (WER та CER) для різних конвеєрів оброблення

Аналіз отриманих результатів дозволив виявити ключові закономірності. Застовування методів бінаризації виявилось найбільш деструктивним: адаптивне порогове перетворення збільшило WER до 96,2%, глобальна бінаризація за Оцу – до 60%, а комбінація розмиття за Гаусом з бінаризацією – до 68%. Таке суттєве погіршення пояснюється видаленням градієнтів яскравості при перетворенні зображення у двоколірне, тоді як саме ці градієнти є важливими ознаками для нейромережевих моделей Google Cloud Vision. Адаптивне підвищення контрасту також негативно вплинуло на результат (WER = 55,9%), оскільки надмірне підсилення контрасту провокувало появу артефактів на нерівномірно освітлених ділянках.

Комплексне застосування всіх десяти методів оброблення одночасно призвело до WER = 52,2%. Це підтверджує гіпотезу про конфлікт між окремими етапами: наприклад, підвищення різкості після шумозаглушення частково повертає видалений шум, а бінаризація після CLAHE підсилює створені артефакти.

Натомість методи, які зберігають градієнти яскравості та виконують лише геометричні корекції, продемонстрували покращення результатів.

Найвищу ефективність показав конвеєр, що складається з конвертації у градації сірого, масштабування та алгоритмічної корекції нахилу: WER = 44,1%, CER = 23% (покращення на 2% за WER та на 2,3% за CER порівняно з базовим рівнем). Другим за ефективністю виявився алгоритм нелокального усереднення (WER = 44,9%).

На другому етапі було проведено детальний порецептний аналіз обраного оптимального конвеєра порівняно з базовим розпізнаванням (рис. 3.17).

№	Рецепт	WER ориг	WER обр	ΔWER	CER ориг	CER обр	ΔCER	Статус
1	recipe_01	56.6%	49.4%	-7.2	33.2%	20.0%	-13.2	покращено
2	recipe_02	47.8%	52.8%	+5.0	32.4%	36.1%	+3.7	погіршено
3	recipe_03	56.2%	53.7%	-2.5	42.1%	38.7%	-3.4	покращено
4	recipe_04	27.9%	36.5%	+8.6	19.5%	27.8%	+8.3	погіршено
5	recipe_05	44.5%	45.4%	+0.9	31.2%	30.5%	-0.7	погіршено
6	recipe_06	58.6%	50.0%	-8.6	22.7%	18.8%	-3.9	покращено
7	recipe_07	40.0%	41.1%	+1.1	18.3%	21.1%	+2.8	погіршено
8	recipe_08	48.6%	38.9%	-9.7	23.2%	16.3%	-6.9	покращено
9	recipe_09	46.2%	57.7%	+11.5	13.7%	11.5%	-2.2	погіршено
10	recipe_10	72.7%	56.4%	-16.3	50.8%	31.1%	-19.7	покращено
11	recipe_11	55.9%	49.2%	-6.7	32.5%	31.3%	-1.2	покращено
12	recipe_12	38.0%	30.0%	-8.0	14.0%	12.1%	-1.9	покращено
13	recipe_13	46.8%	48.9%	+2.1	25.2%	26.1%	+0.9	погіршено
14	recipe_14	37.1%	37.1%	0.0	15.3%	18.4%	+3.1	без змін
15	recipe_15	14.0%	14.0%	0.0	5.3%	5.3%	0.0	без змін
	СЕРЕДНЄ	46.1%	44.1%	-2.0	25.3%	23.0%	-2.3	

Рисунок 3.17 – Детальне порівняння якості розпізнавання по кожному рецепту

Отримані дані свідчать про неоднорідність впливу попереднього оброблення на різні типи зображень. З 15 рецептів покращення зафіксовано у 7 випадках (46,7%), погіршення у 6 (40%), а без змін – у 2 (13,3%). Найбільш позитивний ефект відзначено для рецепта №10 – покращення WER на 16,3% , з 72,7% до 56,4%.

Враховуючи найгіршу початкову якість цього зображення, результат доводить ефективність геометричних корекцій саме для фотографій зі значними дефектами. Суттєве покращення також спостерігалось для рецептів №8 (на 9,7%), №6 (на 8,6%) , та №12 (на 8%).

Цікавою є аномалія, зафіксована для рецепта №9: спостерігалось зростання WER на 11,5%, при одночасному зменшенні CER на 2,2%. Це пояснюється тим, що алгоритм корекції нахилу змінив геометрію зображення, внаслідок чого OCR-сервіс інакше сегментував текст на слова. Така зміна сегментації суттєво вплинула на показник WER, але майже не позначилась на якості окремих символів. Для рецепта №15 результати виявились ідентичними в обох випадках (WER = 14%, CER = 5,3%), що підтверджує надлишковість додаткового оброблення для зображень високої якості.

Результати тестування показали, що хмарний сервіс Google Cloud Vision API містить потужні вбудовані механізми адаптивної фільтрації, що суттєво знижує потребу у зовнішньому попередньому обробленні. Використання жорстоких методів бінаризації є недоцільним для нейромережевих OCR-рішень. Водночас застосування мінімалістичного конвеєра (градації сірого, масштабування та корекція нахилу) є виправданим компромісом, який покращує середню якість розпізнавання за рахунок корекції зображень із геометричними дефектами. Саме цей підхід було обрано для реалізації у розробленій системі.

3.3.2 Тестування модуля структурування тексту на основі великої мовної моделі

Тестування модуля оброблення природної мови, реалізованого на базі Gemini Flash 2.5, було спрямоване на оцінку його здатності перетворювати неструктурований текст із наявними OCR-помилками у структурований JSON-об’єкт.

Головним завданням експерименту стала перевірка точності вилучення сутностей (назва страви, інгредієнти, кроки приготування) та коректності генерації метаданих (складність, кухня, категорія, теги).

Тестування здійснювалось на тих самих 15 рецептах, що й для OCR-модуля. Для кожного рецепта вручну підготовлено еталонний текст з очікуваними значеннями всіх полів. Відповідь моделі було експортовано з бази даних бота, де вони зберігаються під час штатної роботи системи. Порівняння еталонних та отриманих об’єктів виконувалось автоматизовано за допомогою спеціально розробленого Python-скрипту.

Для об’єктивної оцінки результатів необхідно класифікувати поля згенерованого JSON-об’єкта за їхньою природою. Перша група – це поля прямого витягування (назва, інгредієнти, кроки, кількість порцій та час приготування). Їхні значення мають визначатись виключно з вихідного тексту, а генерація неіснуючих даних (галюцинації) класифікується як помилка.

До другої групи належать поля контекстного виведення, тобто категорія страви, кухня,складність, теги та опис. Значення цих полів модель генерує самостійно на основі контексту рецепта. Така поведінка є запланованою архітектурною особливістю, спрямованою на збагачення оцифрованих даних для їх подальшої систематизації.

Для оцінки полів прямого витягування зі списковою структурою (інгредієнти та кроки) використовувались метрики Precision, Recall та F1-score. Оскільки модель може дещо змінити формулювання порівняно з оригіналом, наприклад, перефразувати крок чи нормалізувати назву інгредієнта, порівняння проводилось із застосуванням нечіткого зіставлення рядків на базі алгоритму SequenceMatcher.

Поріг подібності було встановлено на рівні 0,65 для інгредієнтів та 0,50 для кроків, де перефразування трапляється частіше. Для кількісних параметрів та простих полів застосовувалась метрика Accuracy. Зведені результати тестування наведено на рисунку 3.18.

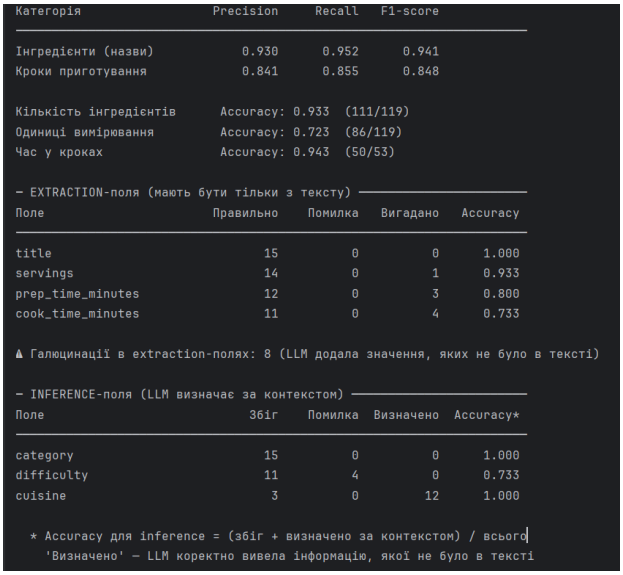


Рисунок 3.18 – Зведені результати тестування NLP-модуля

Найкращі показники отримано для назв інгредієнтів – F1 = 0,941. Слід зазначити, що на вхід модель отримувала текст із середнім показником WER 44,1%, тобто майже кожне друге слово містило помилку розпізнавання. Попри це, Gemini здатна відновлювати правильні назви за контекстом: «ойрок» перетворюється на «огірок», «бол орошна» – на «борошно», а « деконом» – на «бекон». Здатність компенсувати помилки є ключовою перевагою архітектури, що поєднує оптичне розпізнавання з великою мовною моделлю.

Кількість інгредієнтів визначається з точністю 93,3%. Модель коректно обробляє як цілі числа, так і дробові значення, діапазони. Помітно слабшим є результат для одиниць вимірювання – лише 72,3%. Аналіз помилок показав, що основна проблема полягає у неоднозначності скорочень рукописного тексту. Скорочення «ст.» може означати як «стакан», так і «столова ложка», і модель не завжди обирає правильний варіант. Крім того спостерігається непослідовна нормалізація: в одному рецептів модель записує «склянка», а в іншому вказує «стакан» для тієї самої одиниці.

Кроки приготування визначаються з показником F1 = 0,848. Зниження цього показника порівняно з інгредієнтами пояснюється природою самих даних: модель іноді перефразовує інструкції, об’єднує два кроки в один або розбиває складний крок на частини.

Тривалість окремих етапів, зазначена в кроках, визначається з точністю 94,3%, що є практично важливим результатом, бо дозволяє в перспективі реалізувати функцію автоматичних таймерів.

Назва страви розпізнається бездоганно у всіх 15 випадках. Кількість порцій визначено правильно в 14 з 15 рецептів, час приготування вказано з точністю 73,3% , а час підготовки – 80%. Загалом зафіксовано 8 випадків галюцинацій у полях прямого витягування, коли модель генерувала значення часу або порцій, яких не було у вихідному тексті.

Категорія страви визначена правильно для всіх рецептів вибірки: кекси віднесено до випічки, борщ – до першої страви, а смузі до напоїв. Коректно також було визначено кухню у всіх випадках. У 11 з 15 випадках складність зазначено вірно. Помилки здебільшого стосувались рецептів, де кількість інгредієнтів та кроків вказували на різні рівні складності.

Отже, результати тестування підтверджують ефективність використання Gemini Flash 2.5 у ролі модуля структурування тексту. Модель не лише успішно справляється з вилученням даних з тексту, що містить численні помилки розпізнавання, але й доповнює рецепт корисними метаданими для подальшої систематизації.

3.3.3 Функціональне тестування розробленого бота

Завершальним етапом тестування стала перевірка працездатності бота як цілісної системи у середовищі месенджера Telegram. На цьому етапі перевірено усі сценарії взаємодії, реалізовані у підрозділі 3.2.5.

Насамперед тестування розпочато з огляду зовнішнього представлення бота та процесу першого запуску. На рисунку 3.19 показано екран профілю бота з кнопкою «Розпочати», привітальне повідомлення після виконання команди /start , меню доступних команд та результат виклику довідки /help.

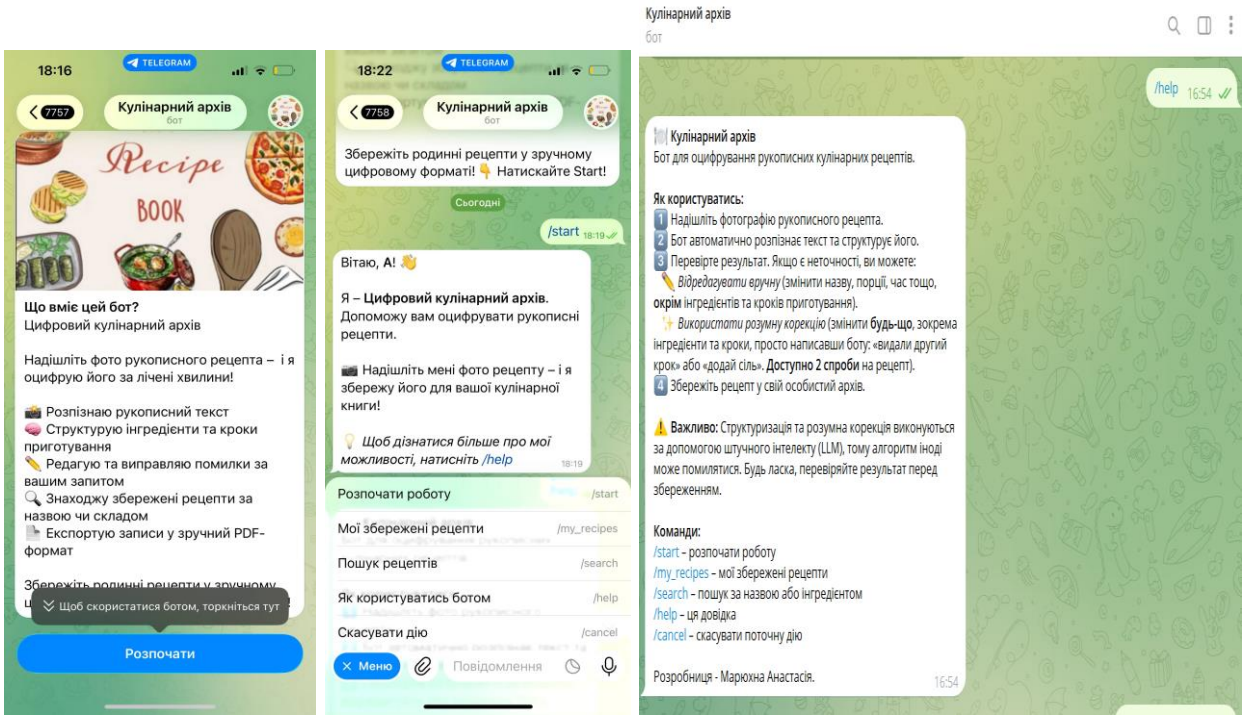


Рисунок 3.19 – Початок роботи з ботом: профіль, привітання, меню команд та довідка

Після перевірки початкового запуску було протестовано основний сценарій, а саме повний етап оцифрування рукописного рецепта. Встановлено, що після надсилання фотографії бот автоматично виконує повний конвеєр оброблення та протягом 10-60 секунд повертає структурований результат з оцінкою впевненості від великої мовної моделі. Під результатом розташовано вбудовану клавіатуру з варіантами дій: збереження, інтелектуальна корекція, ручне редагування та скасування (рис. 3.20).

Крім того, перевірено можливості редагування отриманого результату. По-перше, інтелектуальна корекція дозволяє описати бажані зміни текстовим коментарем, і модель вносить лише вказані користувачем правки, зберігаючи решту полів без змін (рис. 3.21). По-друге, ручне редагування забезпечує можливість точкової зміни конкретного поля за допомогою скінченного автомату станів. Враховуючи багатокроковість цього процесу, детальну послідовність ручного редагування наведено у додатку К. Отже, користувач має змогу обрати найбільш зручний спосіб коригування залежно від характеру необхідних змін.

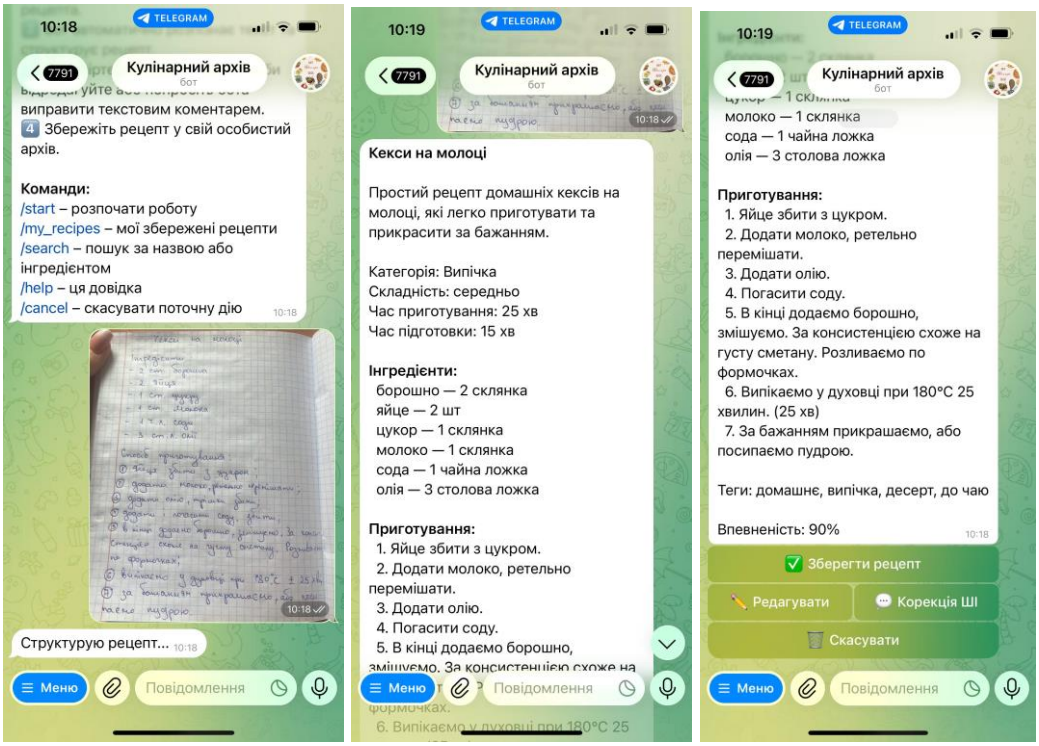


Рисунок 3.20 – Процес оцифрування та результат автоматичної структуризації рецепта

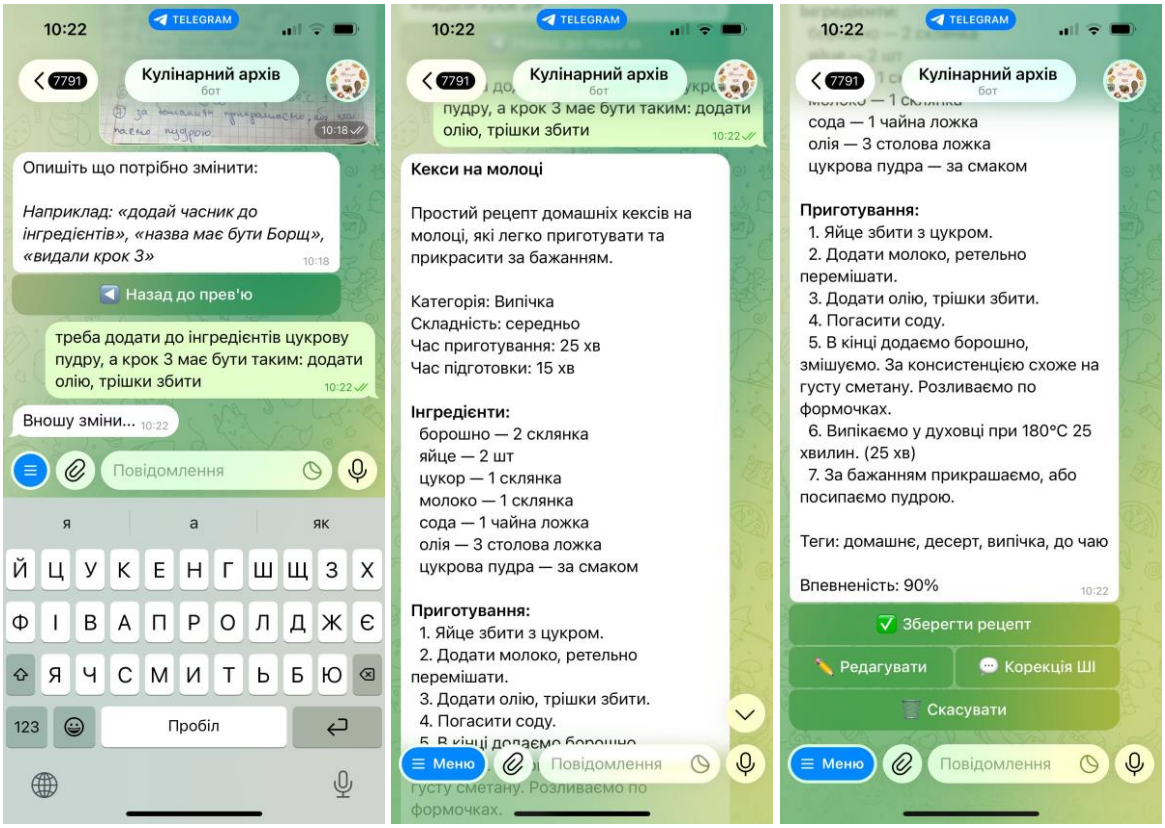


Рисунок 3.21 – Результат виконання інтелектуальної корекції рецепта за текстовим коментарем

Після підтвердження коректності результату структрування користувач зберігає рецепт натисканням відповідної кнопки. Бот фіксує статус рецепта як VERIFIED та повідомляє про успішне додавання запису з підказкою щодо подальших дій. На рисунку Л.1 показано інтерфейс повідомлення та відображення колекції за командою /my_recipes з одним рецептом.

Для перевірки роботи колекції при більшому обсязі даних було оцифровано та збережено 15 рецептів різних категорій. Команда коректно відображає список збережених рецептів із посторінковою навігацією, кнопками переходу між сторінками та лічильником поточної позиції (рис. Л.2).

Протестовано фільтрацію колекції рецептів: користувач може відфільтрувати рецепти за категорією, тегом, типом кухні або рівнем складності. Кожен фільтр спочатку відображає доступні значення із вказівкою кількості рецептів у кожній групі, а після вибору – відфільтрований список рецептів (рис. Л.3). Контекст обраного фільтра зберігається при навігації, завдяки тому, що кнопка «Назад» із перегляду рецепта повертає саме до відфільтрованого списку, а не до загального.

Після вибору конкретного рецепта зі списку відображається його повна повна картка у розширеному форматі з візуальними позначками для часу, порцій, складності та категорій. Під карткою розташовано набір дій: редагування, експорт у PDF, перегляд оригінального фото та видалення з підтвердженням (рис. Л.4).

Також протестовано додаткові можливості роботи зі збереженим записом: експорт рецепта у формат PDF та перегляд оригінальної фотографії рукопису. Встановлено, що бот генерує документ з коректним відображенням кирилиці та надсилає файл безпосердньо у чат. Водночас функція перегляду вихідного зображення поруч із струкрурованим текстом дає змогу візуально оцінити якість розпізнавання та структуризації (рис. Л.5).

Окремо протестовано функцію пошуку за командою /search. Повнотекстовий пошук за назвою знаходить рецепти з урахуванням морфології української мови.

Нечіткий пошук на основі триграм продемонстрував толерантність до помилок. Зокрема, за запитом «бороші» система знаходить рецепти з інгредієнтом «борошно», а коли введено «морков» – рецепти з морквою. Крім того, пошук коректно працює з довідником синонімів інгредієнтів: за запитом «лаврушка» система знайшла два рецепти, в яких інгредієнт вказано як «лавровий лист» (рис. 3.22). Це забезпечує зручність пошуку як за наявності орфографічних неточностей, так і при використанні розмовних варіантів назв інгредієнтів.

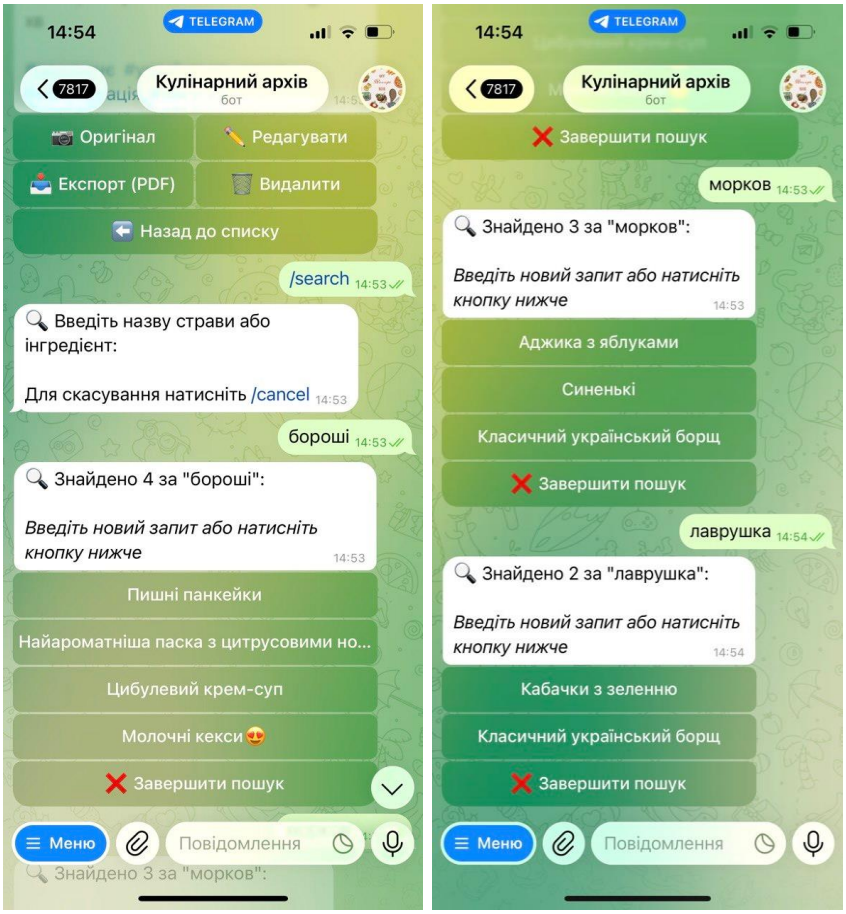


Рисунок 3.22 – Демонстрація нечіткого пошуку та пошуку за синонімами

Важливим етапом тестування стала перевірка стійкості системи до некоректних дій користувача та оброблення граничних випадків. Зокрема, перевірено реакцію бота на надсилання непідтримуваних форматів даних, таких як стікери, голосові повідомлення – у таких випадках система коректно інформує користувача про необхідність надіслати саме фотографію.

Крім того, при надсиланні зображення без кулінарного змісту, успішно спрацьовує семантичний контроль, тоді як при спробі повторного завантаження того самого фото система виявляє дублікат.

Також протестовано поведінку системи у нетипових станах інтерфейсу. Наприклад, перевірено відпрацювання нульової видачі, коли за пошуковим запитом не знайдено жодного рецепта. Окрему увагу приділено перевірці вхідних даних під час роботи FSM у режимі редагування. Зокрема, система успішно перехоплює введення некоректних типів даних, наприклад, коли замість очікуваного додатного числа для кількості порцій користувач вводить текст. Завдяки такій обробці винятків забезпечується цілісність бази даних та безперебійна робота бота без критичних збоїв. Також перевірено безпечне видалення рецепту, яке спочатку запитує згоду на видалення, і лише потім вилучає рецепт з колекції та БД. Демонстрацію оброблення зазначених граничних випадків наведено у додатку М.

Під час тестування для контролю коректності роботи конвеєра використано серверні журнали. Для рецепту «Кекси на молоці» зафіксовано: OCR впевненість 90%, 78 розпізнаних слів; NLP – час відповіді 14,3 с, виділено 6 інгредієнтів та 7 кроків, впевненість 95% (рис. 3.23). Журнали підтвердили коректне проходження всіх етапів оброблення без помилок.

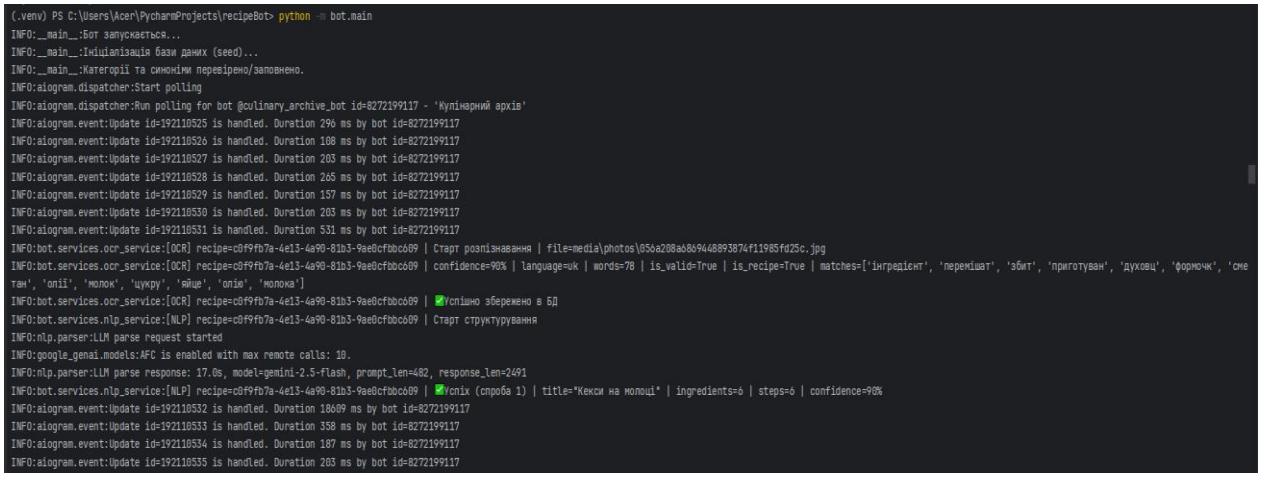


Рисунок 3.23 – Фрагмент серверних журналів

Підсумовуючи результати розділу, слід зазначити наступне.

Модуль оптичного розпізнавання символів забезпечує середнє значення коефіцієнта помилок на рівні слів – 44,1%, проте модуль оброблення природної мови компенсує помилки розпізнавання, забезпечуючи вилучення інгредієнтів з F1-мірою 0,941 та кроків з F1-мірою 0,848. При цьому категорія та кухня визначаються з точністю 100%. Отже, функціональне тестування підтвердило працездатність усіх реалізованих сценаріїв та продемонструвало готовність системи до практичного використання.

3.4 Перспективи подальшої роботи

Результати тестування та поточні архітектурні обмеження системи дозволили визначити напрями подальшого розвитку розробленої системи.

На поточному етапі бот функціонує виключно у локальному середовищі розробника. Для забезпечення постійної доступності необхідно здійснити розгортання системи на віддаленому сервері або хмарній платформі з налаштуванням автоматичного перезапуску, моніторингу стану процесів та захищеного з’єднання з базою даних.

Суттєвим обмеженням є підтримка лише одного зображення на один рецепт. У реальних умовах рукописний рецепт може займати дві або більше сторінок, проте система не передбачає механізму об’єднання кількох фотографій в єдиний текстовий потік. Для усунення цього обмеження необхідно реалізувати режим багатосторінкового введення, в якому користувач послідовно надсилає декілька зображень, а конвеєр виконує розпізнавання кожного окремо та об’єднує результати перед передачею до модуля структуризації. Крім того, на поточному етапі система приймає виключно фото, тоді як додавання підтримки PDF-файлів дозволило б обробляти скановані сторінки кулінарних книг та журналів.

Реалізований алгоритм семантичного контролю побудований на фіксованому переліку стемів, що обмежує його здатність розпізнавати нетипові рецепти – наприклад, страви молекулярної кухні або авторські техніки, які не містять традиційних кулінарних термінів. Перспективним є розширення словника стемів додатковими категоріями та впровадження адаптивного механізму, який поповнює перелік на основі підтверджених користувачем рецептів, що раніше не пройшли автоматичну перевірку.

Окремим напрямком є адаптація системи до багатомовного середовища. У поточній версії реалізації конвеєр оптимізовано виключно для української мови – мовна підказка OCR-сервісу, системна інструкція мовної моделі та словник стемів для автоматичного семантичного контролю орієнтовані на україномовні тексти. Водночас архітектура системи не містить жорстоких мовних обмежень, що створює передумови для підтримки інших мов за умови формування відповідних мовних кофігурацій для кожного з зазначених компонентів.

З метою зменшення залежності від платних хмарних сервісів перспективним є дослідження можливості інтеграції локальних моделей машинного навчання для розпізнавання кириличного рукопису та структурування тексту безпосередньо на сервері застосунку.

Серед функціональних розширень доцільним є розроблення модуля автоматичного формування списку покупок на основі інгредієнтів обраних рецептів, а також реалізація розпізнавання голосових повідомлень для можливості надиктовування рецептів у реальному часі.

У разі збільшення кількості активних користувачів та обсягів даних виникне потреба у впровадженні механізмів кешування для прискорення доступу до часто запитуваної інформації, а також у переході до мікросервісної архітектури з контейнеризацією конвеєрів OCR та NLP для їхнього незалежного масштабування.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено та реалізовано мультимодальний бот в Telegram для оцифрування та систематизації рукописних кулінарних рецептів на основі OCR та NLP. Робота охоплює повний цикл створення застосунку – від аналізу аналогічних рішень до реалізації функціоналу із використанням сучасних інструментів, та вирішено такі завдання:

– проведено комплексний аналіз існуючих мультимодальних ботів у Telegram, що дало можливість виявити ключову прогалину на ринку: відсутність рішення, яке б поєднувало розпізнавання рукописного тексту, інтелектуальну структуризацію та персоналізоване зберігання кулінарних даних;

– здійснено систематичний огляд наукової літератури та сучасних підходів до розпізнавання та структурування кулінарної інформації, що дало можливість обґрунтувати вибір підходу на основі великої мовної моделі замість класичних методів NLP;

– визначено вимоги до мультимодальної взаємодії користувача з системою, що включає оброблення візуальних та текстових даних, та спроєктовано автоматичний конвеєр оброблення рецепту без проміжних кроків взаємодії;

– обрано та налаштовано OCR-конвеєр на основі Google Cloud Vision API з попереднім обробленням зображень засобами OpenCV, а також розроблено алгоритм семантичного контролю з категоризованими маркерами для відсіювання нерелевантних зображень;

– розроблено NLP-модуль на основі великої мовної моделі Gemini для структуризації розпізнаного тексту українською мовою, а також реалізовано механізм інтелектуальної корекції рецептів за допомогою текстових інструкцій природною мовою;

– спроєктовано та реалізовано реляційну базу даних PostgreSQL з чотирнадцяти сутностей з підтримкою повнотекстового пошуку на основі TSVECTOR та нечіткого пошуку на основі pg_trgm з толерантністю до помилок OCR;

– розроблено інтерфейс взаємодії з ботом на основі фреймворку Aiogram з використанням вбудованих клавіатур, механізму скінченного автомата станів та модульної архітектури;

– проведено тестування системи, яке підтвердило її працездатність та стійкість до помилок розпізнавання: на базі моделі Gemini Flash 2.5 забезпечено вилучення інгредієнтів із показником F1 = 0,941 , кроків приготування – з F1 = 0,848, за середнього коефіцієнта помилок розпізнавання слів 44,1%. Точність визначення категорії та кухні страви становила 100%, а функціональне тестування підтвердило коректність роботи всіх сценаріїв взаємодії, нечіткого пошуку та механізмів перевірки даних.

Розглянуто перспективи подальшого розвитку системи, які включають підтримку рецептів на кількох сторінках та PDF-файлів, адаптацію до багатомовного середовища, розгортання на сервері для публічного доступу, а також функціональні розширення – формування списку покупок на основі інгредієнтів обраних рецептів.

Результати кваліфікаційної роботи можуть бути корисними у сфері збереження сімейної кулінарної спадщини у цифровому форматі, систематизації персональних кулінарних архівів та автоматизації побутових рішень.

Огляд сучасних досліджень у галузях комп’ютерного зору, класифікації зображень та інтелектуального оброблення даних підтверджує перспективність подальшого розвитку розробленої системи із залученням нових методів та підходів [43–60].

Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [61].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. OpenCV: Open Source Computer Vision Library. URL: https://opencv.org/ (дата звернення 11.03.2026).

2. Кобилін О.А., Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

3. Google Cloud Vision API. OCR language support. URL: https://docs.cloud.google.com/vision/docs/languages (дата звернення 15.03.2026).

4. Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*, *1*(2), 1-124.

5. Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., ... & Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature communications*, *15*(1), 1418.

6. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

7. OCRBot [image to text] – Unofficial Telegram OCR Robot – Convert images to text. URL: https://telegram.me/ocrbot (дата звернення 27.03.2026).

8. Argus – Бот для розпізнавання тексту на фото / Photo text recognition bot. URL: https://t.me/argus_ocr_bot (дата звернення 27.03.2026).

9. Кулінарні рецепти – Чат-бот містить кулінарні рецепти для приготувань неперевершених страв! URL: https://t.me/tasty_cooking_bot (дата звернення 28.03.2026).

10. AI Chef Bot – Chef-Bot, твій надійний кухонний напарник! URL: https://t.me/Best_AI_Chef_Bot (дата звернення 28.03.2026).

11. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

12. Kalantarov, I. R., & Volkov, D. Y. (2023). Development Of A Telegram-Bot For Automating The Educational Process. *Current Problems Of Teaching Mathematics At Technical University*, *10*, 53-56.

13. Panok, V., Shevchenko, A., Nazar, M. M., Starkov, D., Mescheryakov, D. S., & Shevtsov, A. (2025). Methodological principles of educational and psychological chatbot development. *Information Technologies and Learning Tools*, *2*(106), 76-93.

14. Wahyu, M., & Fitrani, A. S. Application of Telegram Bot for Regional Intranet Network Monitoring System in Government Agencies [Penerapan Bot Telegram untuk Sistem Monitoring Jaringan Daerah di Instansi Pemerintahan].

15. Kjorvezir, D., Najkov, D., Valenčič, E., Jesenko, E., Seljak, B. K., Eftimov, T., & Stojanov, R. (2025, December). Fusing Semantic, Lexical, and Domain Perspectives for Recipe Similarity Estimation. In *2025 IEEE International Conference on Big Data (BigData)* (pp. 6790-6799). IEEE.

16. Ławrynowicz, A., Wróblewska, A., Adrian, W. T., Kulczyński, B., & Gramza-Michałowska, A. (2022). Food recipe ingredient substitution ontology design pattern. *Sensors*, *22*(3), 1095.

17. Tasse, D., & Smith, N. A. (2008). SOUR CREAM: Toward semantic processing of recipes. *Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-LTI-08-005*.

18. Gangania, P., Mishra, S., Garg, S., & Agarwal, S. (2018). Handwriting Recognition System Using Optical Character Recognition. *International Journal of Scientific Research in Computer Science and Engineering*, *6*, 18-21.

19. Paci, H., Minarolli, D., Trandafili, E., & Paturri, S. (2024). Albanian Handwritten Text Recognition using Synthetic Datasets and Pre-Trained Models. *WSEAS Transactions on Information Science and Applications*, *21*, 264-271.

20. Sareen, B., Ahuja, R., & Singh, A. (2024). CNN-based data augmentation for handwritten gurumukhi text recognition. *Multimedia Tools and Applications*, *83*(28), 71035-71053.

21. Kiddon, C., Zettlemoyer, L., & Choi, Y. (2016, November). Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 329-339).

22. SpaCy: Industrial-strength Natural Language Processing in Python. URL: https://spacy.io/ (дата звернення 03.04.2026).

23. NLTK: Natural Language Toolkit. URL: https://www.nltk.org/ (дата звернення 03.04.2026).

24. Straka, M., & Straková, J. (2017, August). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies* (pp. 88-99).

25. Pitsilou, V., Papadakis, G., & Skoutas, D. (2024, May). Using LLMs to extract food entities from cooking recipes. In *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW)* (pp. 21-28). IEEE.

26. Nguyen, Q. N., Sidorova, A., & Torres, R. (2022). User interactions with chatbot interfaces vs. Menu-based interfaces: An empirical study. *Computers in Human Behavior*, *128*, 107093.

27. Shen, W., & Li, S. (2025). Influence of the Use of Emojis by Chatbots on Interaction Satisfaction. *Journal of Marketing Development & Competitiveness*, *19*(2).

28. Neudecker, C., Baierer, K., Gerber, M., Clausner, C., Antonacopoulos, A., & Pletschacher, S. (2021, September). A survey of OCR evaluation tools and metrics. In *Proceedings of the 6th International Workshop on Historical Document Imaging and Processing* (pp. 13-18).

29. Christian, I., & Kusuma, G. P. (2023). Improving OCR performance on low-quality image using pre-processing and post-processing methods. *International Journal of Engineering Trends and Technology*, *71*(6), 396-405.

30. Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., ... & He, L. (2020). A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364*.

31. Dalal, M. K., & Zaveri, M. A. (2011). Automatic text classification: a technical review. *International journal of computer applications*, *28*(2), 37-40.

32. Bashurov, V., & Safonov, P. (2025). Enhancing university education with AI: a Telegram bot leveraging RAG and external APIs for secure knowledge retrieval. *Issues in Information Systems*, *26*(3), 413-420.

33. Souibgui, M. A., & Kessentini, Y. (2020). De-gan: A conditional generative adversarial network for document enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(3), 1180-1191.

34. Velammal, M., Singh, T. I., Patil, N. M., & Pal, S. (2023). MULTIFRAME IMAGE RESTORATION USING GENERATIVE ADVERSARIAL NETWORKS. *ICTACT Journal on Image & Video Processing*, *14*(1).

35. Polat, F., Tiddi, I., & Groth, P. (2025). Testing prompt engineering methods for knowledge extraction from text. *Semantic Web*, *16*(2), SW-243719.

36. General Data Protection Regulation. – Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (дата звернення 15.04.2026).

37. Python Language Reference, version 3.11. URL: https://docs.python.org/3/ (дата звернення 29.04.2026).

38. Aiogram Documentation – Modern and fully asynchronous framework for Telegram Bot API written in Python. URL: https://docs.aiogram.dev/en/v3.28.2/ (дата звернення 29.04.2026).

39. Google Cloud Vision API. Detect handwriting in images. URL: https://docs.cloud.google.com/vision/docs/handwriting (дата звернення 30.04.2026).

40. Google AI for Developers. Gemini API: Structured output. URL: https://ai.google.dev/gemini-api/docs/structured-output (дата звернення 30.04.2026).

41. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/index.html (дата звернення 30.04.2026).

42. SQLite Consortium. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення 30.04.2026).

43. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

44. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

45. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

46. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

47. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

48. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

49. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

50. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134-145.

51. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine*, pp. 25-27.

52. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

53. Gorokhovatskyi V., Chmutov Y., Tvoroshenko I., and Kobylin O. (2025) Reducing computational costs by compressing the structural description in image classification methods, *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

54. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

55. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

56. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

57. Гороховатський В.О., Творошенко І.С. (2025) Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28 вересня 2025 року)*. *Харків: НТУ «ХПІ»*, С. 38-43.

58. Larin I., Tvoroshenko I., Gorokhovatskyi V., and Liubchenko V. (2025) Research and comparison of facial color normalization methods relative to an etalon image, *International Journal of Academic and Applied Research*, 9(11), pp. 36-47.

59. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2026) Feature space quantization and application of metrics in structural methods of image recognition, *IEEE Access*, vol. 14, pp. 17812-17824.

60. Гороховатський В., Творошенко І. (2026) Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія. Харків: ХНУРЕ. 153 с.

61. Марюхна А.В (2026). Аналіз існуючих мультимодальних ботів у Telegram для оцифрування та систематизації рукописних текстів. *Радіоелектроніка і молодь у XXI столітті: матеріали 30-го Міжнародного молодіжного форуму (Харків, 22–24 квітня 2026 р.)*. Харків: ХНУРЕ, 2026. Т. 7. С. 111–113.