International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

Development of a Multimodal Bot for Digitizing and Systematizing Handwritten Culinary Recipes Based on OCR and NLP

Anastasiia Mariukhna, Iryna Tvoroshenko, Volodymyr Gorokhovatskyi, Olena Yakovleva
Department of Informatics
Kharkiv National University of Radio Electronics,
Kharkiv, Ukraine
e-mail: anastasiia.mariukhna@nure.ua; iryna.tvoroshenko@nure.ua; volodymyr.horokhovatskyi@nure.ua; olena.yakovleva@nure.ua

Abstract: *This paper addresses the challenge of digitizing and systematizing handwritten culinary recipes, which remain vulnerable to physical deterioration and inaccessible for search or filtering in their original form. The proposed solution is a multimodal bot that implements an automated processing pipeline combining cloud-based optical character recognition, large language model-based text structuring, and storage in a relational database with fuzzy search support. The OCR pipeline includes image preprocessing using OpenCV and semantic validation through a categorized stem-based algorithm to filter non-culinary content. Text structuring is performed by the Gemini 2.5 Flash model with a self-correction mechanism limited to two iterations. The PostgreSQL database employs full-text search via TSVECTOR and trigram-based fuzzy search via pg_trgm to ensure tolerance to OCR errors and regional vocabulary variants. Experimental evaluation on a dataset of 15 handwritten recipes demonstrated that the structuring module achieves an F1-score of 0.941 for ingredient extraction and 0.848 for cooking steps, with 100% accuracy for dish category and cuisine classification, despite an average word error rate of 44.1% at the OCR stage.*

Keywords—large language models; natural language processing; optical character recognition; recipe digitization; Telegram bot

1. INTRODUCTION

The digital transformation is increasingly encompassing various aspects of everyday life. However, the domestic storage of culinary information remains one of the least affected areas. Millions of unique family recipes exist only in handwritten form: in notebooks, notepads, and on loose sheets of paper, making them vulnerable to physical deterioration and inconvenient for daily use. Simply photographing pages does not solve the problem, as images do not support searching, filtering, or editing content. There is a need for an intelligent tool capable of not only reading text from photographs but also understanding its semantic structure and transforming it into a functional digital knowledge base.

Messaging platforms, particularly Telegram, have become a convenient environment for deploying intelligent tools due to their wide accessibility and powerful API for developers. The use of bots allows automating image and text processing tasks without installing additional software, which lowers the entry barrier for users [1, 2].

Multimodality in the context of chatbots refers to the system's ability to perceive and process different types of input data simultaneously – in this case, photographs of handwritten pages and textual instructions for recipe correction. The system must integrate these data into a unified logical structure for subsequent search and analysis.

An analysis of existing solutions within this environment revealed the absence of a comprehensive product for digitizing handwritten culinary recipes. Available tools either provide only optical character recognition without semantic structuring of the text, or work exclusively with pre-existing recipe databases without supporting the digitization of users' own records.

Optical Character Recognition (OCR) technology has come a long way from simple template matching to deep neural networks. Cloud services, such as Google Cloud Vision, provide handwritten text recognition for a wide range of languages, including Ukrainian, without the need to train custom models [3]. However, the quality of handwritten text recognition remains significantly lower compared to printed text, which requires additional image preprocessing stages and subsequent intelligent processing of the results.

For structuring the recognized text, classical Natural Language Processing (NLP) methods, such as regular expressions and rule-based parsers, are ineffective due to the arbitrary structure of handwritten recipes: authors use varying element sequences, non-standard abbreviations, informal vocabulary, and mix languages [4, 5].

Large Language Models (LLMs) demonstrate the ability to structure free-form text through a single textual instruction, allowing the system to be adapted to a specific domain without creating rigid rules for each formatting variant [6, 7].

This work presents a multimodal bot for digitizing and systematizing handwritten culinary recipes. The system implements an automated processing pipeline: image preprocessing using the OpenCV library, text recognition via the Google Cloud Vision API, semantic validation for culinary domain relevance, structuring using the Gemini 2.5 Flash model, and storage in a PostgreSQL relational database with full-text and fuzzy search support [8, 9]. Additionally, an intelligent recipe correction mechanism through natural language text instructions and PDF export have been implemented.

The object of this work is the process of developing a multimodal bot. The goal is the development and experimental evaluation of an automated system for digitizing and systematizing handwritten culinary recipes based on OCR and NLP technologies.

To establish the technical foundation and identify the functional gaps addressed by the proposed system, the following section examines existing solutions in the Telegram bot ecosystem and relevant advances in OCR, NLP, and human-computer interaction research.

2. Related Works

An analysis of existing solutions within the Telegram environment was conducted to identify functional gaps. Four bots of two types were examined: universal OCR services (@ocrbot [10], Argus [11]) and specialized culinary assistants («Culinary Recipes» [12], AI Chef Bot [13]).

Universal OCR bots demonstrate high accuracy on printed text but exhibit critical shortcomings on handwritten input – low recognition accuracy, incorrect interpretation of culinary terms, and loss of document structure.

Specialized culinary bots offer quality visualization and navigation but function as closed systems without support for photo uploading or personal archive creation. A summary comparison is presented in Table 1. None of the analyzed products provides a complete cycle from a photograph of a handwritten recipe to a structured record with search and editing capabilities.

In recent scientific literature, the processing of unstructured handwritten documents holds a prominent position. Traditional OCR systems rely on convolutional and recurrent neural networks, with particular attention to line segmentation and word isolation to minimize errors caused by text skew and irregular spacing [14, 15].

In [15], the authors emphasize that skewed lines and uneven intervals remain the primary sources of errors in classical algorithms. For the Ukrainian language, recognition accuracy is further complicated by the limited volume of available open handwritten datasets, which necessitates image preprocessing methods: shadow removal, grayscale conversion, and contrast enhancement, to reduce error rates.

Table 1: Comparison of functional capabilities of existing Telegram bots

Criterion	OCRBot	Argus	Culinary Recipes	AI Chef Bot
Printed text recognition	Yes	Yes	No	No
Handwritten text recognition	Yes (with errors)	Yes	No	No
Ukrainian language support	Yes (unstable)	Yes	Yes	Yes
Intelligent text correction	No	Yes	No	Yes
Culinary specialization	No	No	Yes	Yes
Automatic recipe structuring	No	No	Yes	Yes
Personal knowledge base creation	No	No	No	No
Multimodal interaction	No	Yes	No	No

In [16], a combination of shadow removal and optimized grayscale conversion reduced the overall OCR error rate to 14.56%, with additional post-processing bringing it down to 13.94%.

Semantic text analysis within the culinary domain was long based on classical NLP libraries and ontology development [4, 5, 17-19]. Such approaches require complex configuration of tokenization, lemmatization, and morphological analysis rules.

In [4], the authors proposed ontologies for linking different names of the same ingredients, enabling the system to recognize that regional variants refer to identical products. However, rule-based approaches prove ineffective when processing informal texts with arbitrary structure, where authors use non-standard abbreviations and individual writing styles.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

The rapid development of transformer architectures and large language models has opened an alternative methodology. In [7], it was demonstrated that pre-trained LLMs can simultaneously perform named entity recognition and relationship extraction, transforming unstructured text into JSON format without creating separate rules for each entity type.

The application of this approach specifically to the culinary domain was investigated in [19], where the authors proposed a methodology for extracting food entities from recipes in a zero-shot regime, without any pre-annotated training data. The results confirmed that LLMs can recognize ingredient names, units of measurement, and culinary actions with promising accuracy even without domain-specific fine-tuning. This finding strongly supports the approach adopted in this work, as handwritten culinary recipes contain analogous entity types.

Research in human-computer interaction substantiates the use of messengers as deployment platforms [20]. Menu-oriented interfaces with inline keyboards provide significantly lower cognitive load compared to free-form dialogues [21]. Asynchronous architecture and delegation of computationally intensive processes to external cloud APIs ensure high system response speed.

Thus, the theoretical analysis and study of existing solutions confirm the relevance of developing a multimodal pipeline that combines image preprocessing, cloud-based OCR, LLM-based text structuring, and long-term data storage in a relational database for building a personal user knowledge base.

3. System Architecture and Methods

Step 1: Overall System Architecture

The implementations are based on Python 3.12, chosen for its mature ecosystem of AI-related libraries and native support for asynchronous programming via async/await constructs [22]. The bot framework Aiogram 3 was selected over the alternative python-telegram-bot due to its exclusive orientation toward asynchronous architecture, which ensures a unified approach without mixing synchronous and asynchronous paradigms – a critical requirement given the chain of external API calls triggered by each user request.

For optical character recognition, Google Cloud Vision API was chosen for its support of handwritten text recognition with Cyrillic language hints and a free tier of 1000 requests per month sufficient for the development phase. Text structuring is delegated to the Gemini 2.5 Flash model, selected for two reasons: the response_mime_type parameter guarantees JSON-only output at the API level, eliminating the need for complex response parsing, and the Flash variant is optimized for execution speed appropriate for field extraction tasks [23].

PostgreSQL was chosen as the database management system for its built-in full-text search via TSVECTOR, fuzzy search via the pg_trgm extension, and JSONB support for semi-structured data – capabilities absent in alternatives such as SQLite or document-oriented databases [24].

The system architecture is built on a modular principle with a clear separation of functional responsibilities between components. Modularity is driven by the integration of several external services, each of which may change its API or be replaced with an alternative solution, so component isolation allows such changes to be made locally.

The system consists of four main components: the Telegram API interaction module, the optical character recognition module, the natural language processing module, and the database module. Coordination between components is handled by a service layer that manages OCR and NLP pipeline invocation, a retry mechanism with exponential backoff for unavailable external services, and logging of results at each stage. The overall system architecture is presented in Fig. 1.

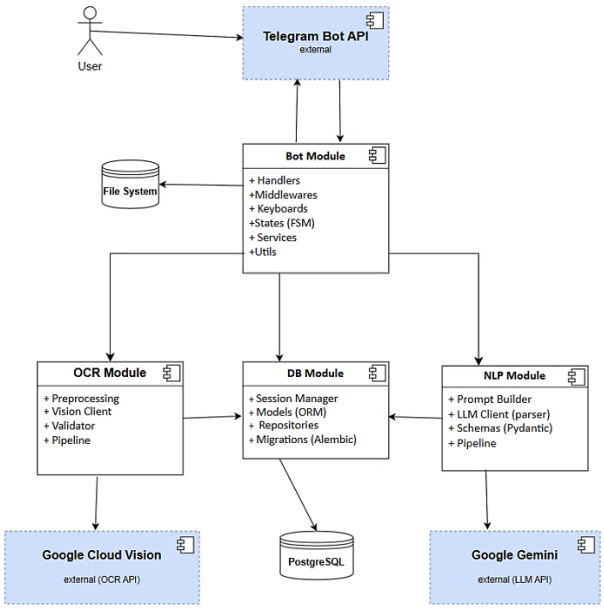

Fig. 1. *Overall architecture of the multimodal recipe digitization system*

The processing pipeline executes automatically as a single process: the user sends a photograph of a handwritten recipe and receives a structured result for review without intermediate interaction steps. The sequence of stages is: image reception → hash computation for duplicate detection → preprocessing → text recognition → semantic validation → structuring → database storage → response generation. Interaction with external services is implemented asynchronously, allowing the bot to continue processing other messages while awaiting responses from cloud APIs.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

Communication with the Telegram Bot API is implemented via the long polling mechanism, where the bot periodically sends requests to Telegram servers to retrieve new messages. This simplifies deployment, as the bot can operate within private networks without additional infrastructure. Navigation is based on interactive menus with inline buttons, which minimizes the need for text input and provides multi-level navigation.

Step 2: Optical Character Recognition Pipeline

The optical character recognition pipeline is implemented as a sequence of three stages: image preprocessing, cloud service recognition, and semantic validation of results.

Before initiating the three-stage recognition pipeline, the system computes a SHA-256 hash of the raw uploaded image bytes and compares it against previously stored hashes for the current user. This constitutes an exact binary match: only byte-identical files are detected as duplicates, whereas re-photographing the same page under different conditions produces a distinct hash and is processed as a new submission. If a match is found, processing is terminated and the user is presented with a link to the existing recipe. This mechanism prevents redundant consumption of Google Cloud Vision and Gemini API quotas when users accidentally resubmit the same image file.

Preprocessing includes three operations implemented using the OpenCV library: conversion to grayscale to reduce data volume, scaling to a width of 1800 pixels while preserving aspect ratio (within the 1024-2048 pixel range recommended by the Google Cloud Vision API documentation), and algorithmic text deskewing. The deskew algorithm determines the rotation angle through the minimum bounding rectangle of text pixels with two protective constraints: skew less than 0.5° is ignored as insignificant, while skew exceeding 15° is rejected as a likely false positive.

Text recognition is performed by the Google Cloud Vision API using the document_text_detection method with the language hint parameter set to “uk” to improve accuracy on Cyrillic text. The average confidence score for the entire image is computed based on the confidence level of each recognized symbol as a diagnostic quality indicator.

Semantic validation consists of two sequential checks. First, the system verifies that the recognized text is non-empty and contains at least five words; otherwise, processing is terminated and the user is advised to submit a higher-quality photograph. Second, the algorithm determines whether the text belongs to the culinary domain through a categorized stem-based analysis. Stems are initial parts of words that cover all morphological forms and are divided into two categories: strong stems (culinary actions and specialized cookware: “smazyt” [to fry], “varyt” [to boil], “kastrul” [pot]) and weak stems (product names and units of measurement: “borosn” [flour], “molok” [milk], “hram” [gram]).

The decision logic is as follows: two or more strong stems unambiguously classify the text as a recipe; one strong stem combined with three weak ones also qualifies; the presence of only weak stems is insufficient for a positive conclusion. The algorithm is shown in Fig. 2. If insufficient markers are found, the system requests user confirmation instead of automatic rejection, which prevents false negatives for atypical recipes.

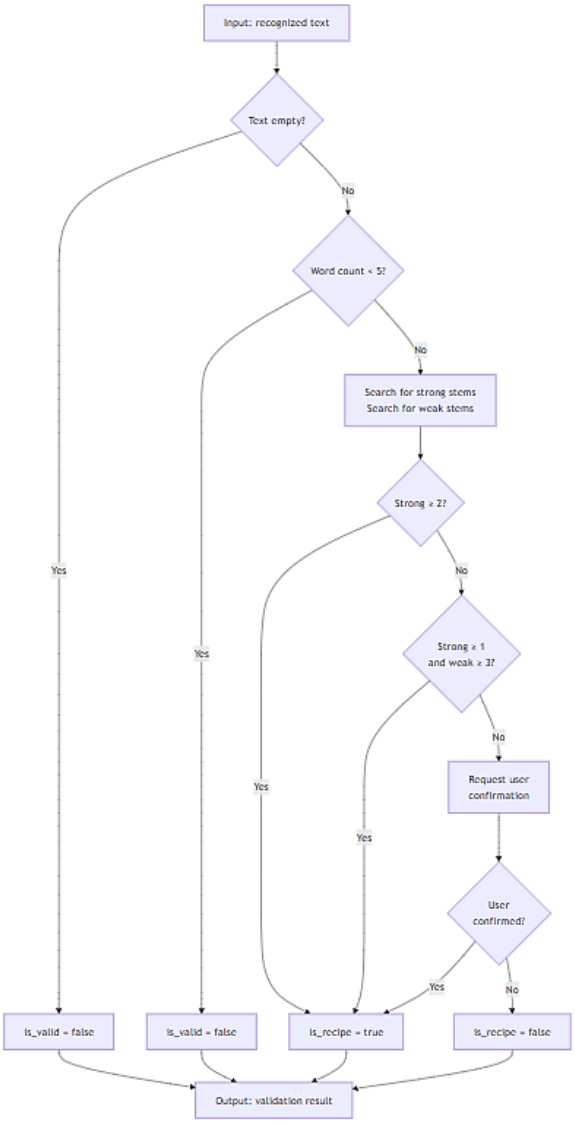

Fig. 2. *Flowchart of the semantic validation algorithm for OCR results*

Upon successful validation, the OCR results: recognized text, confidence score, and detected language, are stored in the database, and the text is automatically passed to the NLP pipeline for structuring. If any stage fails due to external service unavailability, the recipe status is set to FAILED with the error reason preserved, allowing the user to retry later.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

Step 3: Natural Language Processing Pipeline

Text structuring is performed by the Gemini 2.5 Flash model via the Google API. The instruction for the model consists of system directives and the recipe text. The system instruction contains three groups of rules.

The first group covers general structuring rules with two blocks. The extraction block requires the model to use exclusively information from the input text, set null values for undetermined fields, and supplement the ingredient list with items mentioned only in the cooking steps. The inference block allows the model to determine from context the dish category, cuisine type, tags, and difficulty level based on quantitative criteria.

The second group governs ingredient processing: normalization of names to the nominative case singular, separation of quantity and units of measurement, and preservation of non-standard culinary measures (“bunch”, “handful”, “clove”).

The third group defines step processing rules with built-in correction of typical OCR Cyrillic substitutions. The model receives examples of substitutions (“ma” → “та” [and], “Ha” → “На” [on], “sliu” → “oliu” [oil]) and instructions to rely on sentence context for corrections.

The model extracts thirteen semantic components from the text: title, description, ingredients with quantities and units, step-by-step instructions, cooking time, servings, cuisine, difficulty, category, tags, and a confidence score. The temperature parameter is set to 0.1 to minimize output variability while preserving minimal creativity for inference fields. The response_mime_type parameter is set to application/json, guaranteeing that the model returns exclusively JSON output without accompanying text or formatting. The expected response structure is described by the Pydantic schema ParsedRecipe (Fig. 3), which provides automatic type checking, required field validation, and acceptable value range verification.

```python
class ParsedRecipe(BaseModel):  10 usages

    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)",
    )
```

Fig. 3. *Pydantic schema of the ParsedRecipe model*

If the response does not conform to the schema, a self-correction mechanism is initiated: identified errors are transformed into a textual description and passed back to the model. The complete NLP pipeline, including the self-correction loop and user interaction flow, is presented in Fig. 4. The cycle is limited to two iterations based on experimental testing. The second attempt resolves most formatting errors, while a third does not yield significant improvement. Resilience to temporary API failures is ensured by an exponential backoff algorithm with intervals of 2, 4, and 8 seconds. If all retry attempts are exhausted, the recipe status reverts to OCR_DONE, allowing the user to reinitiate structuring later.

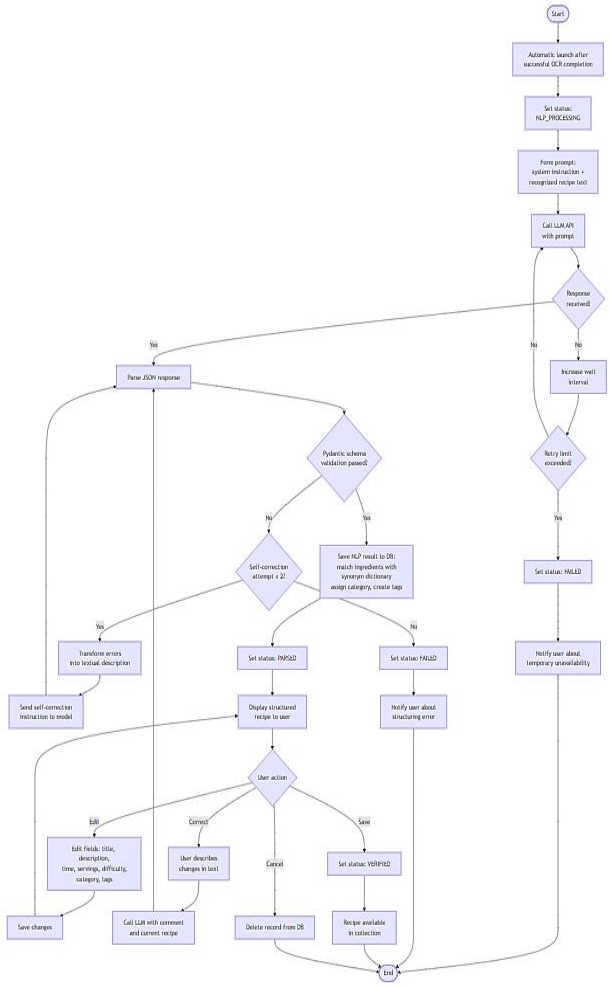

Fig. 4. *Activity diagram of the NLP pipeline for recipe structuring and correction*

Upon successful validation, the structured result is stored in the database: recipe fields are updated, ingredients are matched against the synonym dictionary to ensure unified naming, the category is assigned from the controlled dictionary, and tags are created automatically. The recipe status is then set to PARSED.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

In addition to initial structuring, the pipeline supports correction of already saved recipes through natural language text comments from the user (e.g., "remove the onion and add garlic"). A separate correction instruction requires the model to apply only the requested changes: adding an ingredient extends the list, deleting a step triggers renumbering, and names are normalized to the nominative case. The correction result undergoes the same validation and self-correction as the initial structuring. The previous recipe state is preserved as a separate database record with the user's comment, ensuring a complete change history. The number of corrections is limited to two per recipe to control API usage costs.

Step 4: Database Architecture and Search Strategies

The PostgreSQL relational database is designed with fourteen entities organized into three logical layers (Fig. 5). The user data layer contains user profiles, recipes with attributes, ingredients with quantities and units, and step-by-step instructions. The processing artifacts layer records the results of each pipeline stage: image metadata, OCR results with confidence levels, NLP results with full model responses, and recipe processing status. The reference layer provides normalization through a controlled category dictionary, a free tagging system, and an ingredient synonym dictionary with regional Ukrainian language variants.

All entities use UUID primary keys generated on the application side before database insertion, ensuring independence from the database platform. The recipe_ingredients junction table stores both structured data (numeric quantity, unit of measurement) and the original handwritten text, preserving the authentic wording alongside the normalized representation.

A key architectural principle is raw data immutability: the original image, OCR text, and full LLM response are stored alongside the structured result and are never overwritten. This ensures the possibility of reprocessing when recognition models are improved or instructions are modified. For semi-structured responses from external services, JSONB fields are used, providing storage flexibility without requiring schema modifications when the API format changes.

Data integrity is guaranteed by cascading behavior rules: deleting a recipe automatically removes all dependent records (ON DELETE CASCADE), while deleting a reference dictionary entry is prohibited when references exist (ON DELETE RESTRICT).

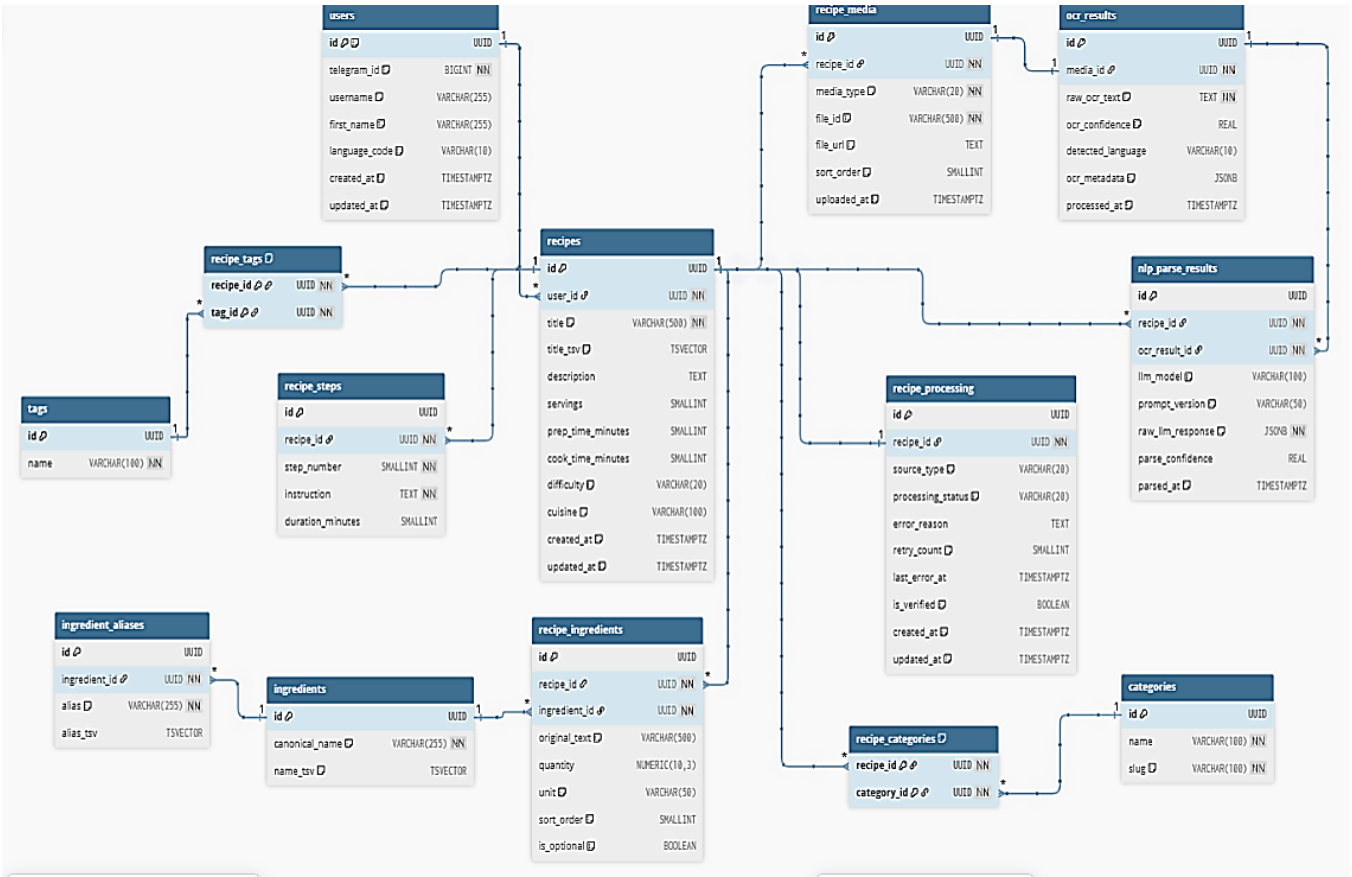

Fig. 5. *Entity-relationship diagram of the database structure*

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 10 Issue 6, June – 2026, Pages: 18-29

The search subsystem combines three indexing strategies. Full-text search by recipe name is implemented via TSVECTOR with a GIN index. Fuzzy search by ingredients uses the pg_trgm extension, which compares strings by matching three-character substrings with a similarity threshold of 0.3. The search is performed simultaneously across canonical ingredient names and their synonyms from the dictionary (e.g., "kartoplya" – "barabolya" – "bulba", all meaning "potato"), providing tolerance for both OCR errors and the variability of Ukrainian culinary vocabulary.

Filtering by categories, tags, and attributes uses standard B-Tree indexes on foreign keys.

Step 5: User Interface and Interaction

With the database layer providing structured storage, full-text indexing, and fuzzy search capabilities, the remaining architectural component is the presentation layer responsible for exposing these functions to the end user. The Telegram messenger platform serves this role, translating database operations into an interactive dialogue managed by a finite state machine. The user interface is implemented through Telegram's native capabilities without developing a separate frontend application. User registration is fully automated: upon the first /start command, the system creates a profile based on the unique Telegram identifier without requiring additional forms or personal data input.

Upon completing the structuring pipeline, the bot displays a formatted recipe preview with an inline keyboard offering four actions: save intelligent correction, manual editing, and cancellation. Each button encodes the action type and recipe identifier in the callback data field. The complete digitization workflow is illustrated in Fig. 6, demonstrating the sequence from photo submission through processing to the final structured preview with action buttons.

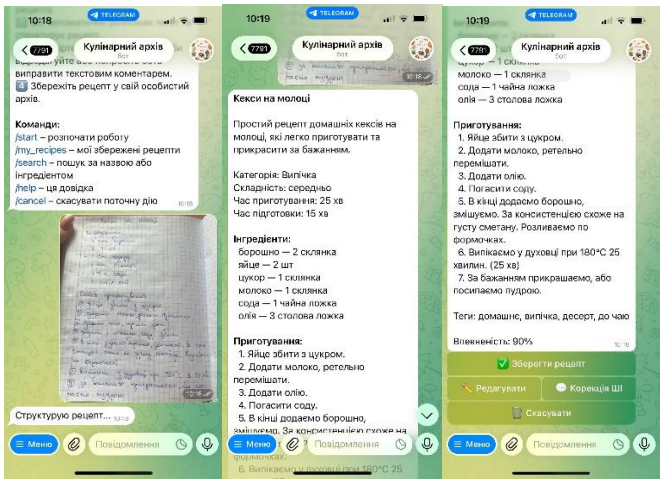

Fig. 6. *Screenshots of the bot application showing the recipe digitization workflow: photo submission, processing status, and structured result with action menu*

Manual editing allows modification of individual fields such as title, description, cooking time, servings, difficulty, category, and tags through dedicated dialogue flows managed by a Finite State Machine (FSM). Fields with constrained value sets are presented as selection menus, while free-text fields accept direct message input with validation.

The recipe collection is accessible via the /my_recipes command with support for filtering by category, tags, cuisine type, and difficulty level. Each filter displays available values with recipe counts before selection. Pagination with navigation buttons handles large collections. Detailed recipe view includes all structured data alongside the option to view the original handwritten photograph, allowing users to verify the structuring result against the source and preserving the sense of authenticity of family records. Recipe export generates a PDF document containing the full recipe with title, ingredients, step-by-step instructions, and metadata. The document is rendered using the ReportLab library with DejaVu Sans font for Cyrillic support and sent directly to the chat as a downloadable file.

The search function, activated by the /search command, transitions the bot into a text input awaiting state and performs a combined query: full-text search by recipe name and fuzzy trigram-based search by ingredients, with results deduplicated at the SQL level. Messages that the system cannot interpret, such as text without an active input state, stickers, documents, or voice messages, trigger an explanatory response directing the user to the /help command.

4. The Testing

Test A: Dataset

A test dataset of 15 photographs of handwritten culinary recipes from different authors was assembled for the experiments. The images varied in handwriting type, lighting conditions, photography quality, and the presence of geometric distortions, approximating real-world usage scenarios. For each photograph, a ground truth text was prepared through manual transcription, serving as the basis for quantitative comparison. Recognition quality was evaluated using two standard metrics: Word Error Rate (WER), defined as the ratio of the minimum number of editing operations (insertions, deletions, substitutions) to the total number of words in the reference text, and Character Error Rate (CER), calculated analogously at the character level [25].

Test B: OCR Module Testing

A comparative analysis of ten image preprocessing pipeline variants was conducted on all 15 images. The investigated methods included grayscale conversion, geometric scaling with deskewing, adaptive thresholding, Otsu global binarization, Contrast Limited Adaptive Histogram Equalization (CLAHE), Gaussian blur combined with binarization, bilateral filtering, non-local means denoising, sharpening, and their comprehensive combination.

The baseline recognition level without preprocessing was WER=46.1% and CER=25.3%. The results are summarized in Table 2.

Table 2: Comparison of preprocessing pipeline variants

Pipeline	WER, %	CER, %	Δ WER	Δ CER
Gray + Resize + Deskew	44.1	23.0	-2.0	-2.3
Gray + FastNlMeans	44.9	22.6	-1.2	-2.7
Grayscale	45.0	24.1	-1.1	-1.2
Baseline (no processing)	46.1	25.3	0.0	0.0
Gray + Bilateral	46.6	23.2	+0.5	-2.1
Gray + Sharpen	46.8	25.7	+0.7	+0.4
All methods combined	52.2	26.8	+6.1	+1.5
Gray + CLAHE	55.9	29.5	+9.8	+4.2
Gray + Otsu	60,0	41.8	+13.9	+16.5
Gray + Gaussian blur + Otsu	68.0	51,2	+21.9	+25.9
Gray + Adaptive thresholding	96.2	92.1	+50.1	+66.8

Binarization methods proved most destructive: adaptive thresholding increased WER to 96.2%, as the conversion to a binary image removes brightness gradients that serve as important features for neural network models of Google Cloud Vision. CLAHE also negatively affected results (WER=55.9%), since excessive contrast enhancement produced artifacts in unevenly lit areas. The combined application of all ten methods simultaneously resulted in WER=52.2%, confirming the hypothesis of conflict between individual preprocessing stages.

The best results were achieved by the minimalist pipeline consisting of grayscale conversion, scaling, and algorithmic deskewing. A detailed per-recipe comparison of this pipeline against baseline recognition is presented in Table 3.

The preprocessing pipeline improved recognition in 7 out of 15 cases (46.7%), worsened it in 6 cases (40.0%), and had no effect in 2 cases (13.3%). The most significant improvement of 16.3% WER was observed for Recipe 10, which had the lowest initial quality, confirming the effectiveness of geometric corrections for images with severe distortions. A notable anomaly was recorded for Recipe 9: WER increased by 11.5% while CER simultaneously decreased by 2.2%. This is explained by the deskew algorithm altering the image geometry, causing the OCR service to segment text into words differently. Such segmentation changes significantly affected WER but had minimal impact on individual character quality. For Recipe 15, results were identical in both cases (WER=14.0%, CER=5.3%), confirming the redundancy of additional preprocessing for high-quality images.

The results demonstrated that Google Cloud Vision API contains powerful built-in adaptive filtering mechanisms, substantially reducing the need for external preprocessing. The application of aggressive binarization methods is counterproductive for neural network-based OCR solutions. However, the minimalist pipeline represents a justified compromise that improves average recognition quality through correction of images with geometric defects. This approach was selected for the final system implementation.

Table 3: Per-recipe comparison: baseline vs. selected pipeline (Gray + Resize + Deskew)

Recipe	WER base, %	WER pipeline, %	Δ WER	CER base, %	CER pipeline, %	Δ CER	Status
Recipe_01	56.6	49.4	-7.2	33.2	20.0	-13.2	Improved
Recipe_02	47.8	52.8	+5.0	32.4	36.1	+3.7	Worsened
Recipe_03	56.2	53.7	-2.5	42.1	38.7	-3.4	Improved
Recipe_04	27.9	36.5	+8.6	19.5	27.8	+8.3	Worsened
Recipe_05	44.5	45.4	+0.9	31.2	30.5	-0.7	Worsened
Recipe_06	58.6	50.0	-8.6	22.7	18.8	-3.9	Improved
Recipe_07	40.0	41.1	+1.1	18.3	21.1	+2.8	Worsened
Recipe_08	48.6	38.9	-9.7	23.2	16.3	-6.9	Improved
Recipe_09	46.2	57.7	+11.5	13.7	11.5	-2.2	Worsened
Recipe_10	72.7	56.4	-16.3	50.8	31.1	-19.7	Improved
Recipe_11	55.9	49.2	-6.7	32.5	31.3	-1.2	Improved
Recipe_12	38.0	30.0	-8.0	14.0	12.1	-1.9	Improved
Recipe_13	46.8	48.9	+2.1	25.2	26.1	+0.9	Worsened
Recipe_14	37.1	37.1	0.0	15.3	18.4	+3.1	No change
Recipe_15	14.0	14.0	0.0	5.3	5.3	0.0	No change
Average	**46.1**	**44.1**	**-2.0**	**25.3**	**23.0**	**-2.3**	

Test C: NLP Module Testing

The NLP module testing evaluated the ability of Gemini 2.5 Flash to transform unstructured text with OCR errors into a structured JSON object. The generated fields were classified into two groups: direct extraction fields (title, ingredients, steps, servings, cooking time), whose values must be determined exclusively from the source text, and contextual inference fields (category, cuisine, difficulty, tags, description), which the model generates based on recipe context. For list-structured extraction fields (ingredients and steps), Precision, Recall, and F1-score metrics were used with fuzzy string matching based on the SequenceMatcher algorithm. Similarity thresholds were set at 0.65 for ingredients and 0.50 for steps, where paraphrasing occurs more frequently [26-30]. For quantitative parameters and simple fields, the Accuracy metric was applied.

The results are presented in Table 4.

Table 4: NLP module testing results

Field	Metric	Value
Ingredient names	F1-score	0.941
Ingredient quantity	Accuracy	93.3%
Units of measurement	Accuracy	72.3%
Cooking steps	F1-score	0.848
Step duration	Accuracy	94.3%
Dish title	Accuracy	100%
Servings	Accuracy	93.3%
Cooking time	Accuracy	73.3%
Preparation time	Accuracy	80.0%
Dish category	Accuracy	100%
Cuisine type	Accuracy	100%
Difficulty level	Accuracy	73.3%

It should be noted that the model received input text with an average WER of 44.1%, meaning nearly every second word contained a recognition error. Despite this, Gemini demonstrated the ability to restore correct names from context: “oirok” was transformed to “ohirok” (cucumber), “bol oroshna” to “boroshno” (flour). This error compensation capability is a key advantage of the architecture combining optical recognition with a large language model.

The highest results were achieved for ingredient names (F1=0.941). Ingredient quantities were determined with 93.3% accuracy, with the model correctly handling integers, fractions, and ranges [31-34]. The weakest result was observed for units of measurement (72.3%), primarily due to ambiguous handwritten abbreviations where the same symbol could denote different units. Cooking steps were identified with F1=0.848, with the lower score compared to ingredients explained by the model occasionally paraphrasing instructions, merging two steps into one, or splitting a complex step into parts [35-41].

Test D: Functional Testing

Functional testing verified all implemented interaction scenarios within the Telegram messenger environment. The complete digitization pipeline processes a photograph and returns a structured result within 10 to 60 seconds. Both correction mechanisms were validated: intelligent correction successfully applies only the requested changes described in natural language, while manual editing correctly modifies individual fields through FSM-managed dialogues.

The fuzzy search demonstrated tolerance to errors: the query “boroshi” successfully found recipes containing “boroshno” (flour), and the synonym dictionary correctly mapped “lavryshka” to recipes listing “lavrovyi lyst” (bay leaf) as an ingredient (Fig. 7). The system correctly handles unsupported input formats, duplicate photo submissions, null search results, and invalid data entry during editing, maintaining database integrity without critical failures.

Fig. 7. *Screenshots demonstrating fuzzy search and synonym-based search functionality*

Collectively, the experimental results confirm the viability of the proposed pipeline across all processing stages. The OCR module achieves a measurable improvement over baseline recognition through minimal preprocessing, the NLP module demonstrates robust error compensation with high extraction accuracy despite elevated word error rates, and functional testing validates the reliability of end-to-end interaction scenarios including search, correction, and export. These findings provide the basis for the conclusions presented below.

5. Conclusion

In this paper, a multimodal bot system for digitizing and structuring handwritten culinary recipes was presented and evaluated.

The developed architecture successfully bridges the gap between legacy paper-based records and modern digital databases by integrating computer vision, relational data storage, and large language models into a unified, asynchronous processing pipeline.

The experimental evaluation demonstrated that while modern cloud-based optical character recognition services like Google Cloud Vision API are highly capable, aggressive image binarization techniques degrade their performance.

A minimalist preprocessing pipeline consisting of grayscale conversion, scaling, and algorithmic deskewing proved to be the optimal compromise, reducing the baseline Word Error Rate to 44.1% and Character Error Rate to 23.0%.

The core contribution of this research lies in demonstrating the exceptional error compensation capabilities of the Gemini 2.5 Flash model acting as a post-OCR semantic layer. Despite receiving text where nearly every second word contained a recognition error, the model successfully applied contextual understanding to reconstruct distorted culinary terms, achieving an F1-score of 0.941 for ingredient names and 0.848 for cooking steps, alongside 100% accuracy in cuisine and category classification. The implementation of Pydantic-based schema validation paired with an automated self-correction mechanism ensured strict JSON output compliance and system resilience.

The PostgreSQL database with full-text search via TSVECTOR and fuzzy trigram-based search via pg_trgm, combined with a regional ingredient synonym dictionary, ensures effective recipe retrieval tolerant to both OCR errors and Ukrainian culinary vocabulary variations. Functional testing confirmed the reliability of all interaction scenarios, including intelligent recipe correction through natural language instructions and synonym-based search capabilities.

The results of this work can be applied to the preservation of family culinary heritage in digital format and the systematization of personal culinary archives.

Future work will focus on expanding the system's capabilities to support multi-page recipe digitization and PDF processing. Additionally, the modular architecture provides a foundation for multilingual adaptation and the integration of functional extensions, such as voice input for real-time dictation and automated grocery list generation based on extracted ingredients. To ensure high availability and scalability for a broader user base, transitioning to a containerized microservices deployment model is planned.

6. Acknowledgment

The papers acknowledge the support of the Department of Informatics, Kharkiv National University of Radio Electronics, Ukraine, in numerous help and support to complete this paper.

The results of this research were obtained under the international research project INITIATE under the grant No. 101136775 HORIZON-WIDERA-2023-ACCESS-03.

7. References

[1] Kalantarov, I. R., & Volkov, D. Y. (2023). Development of a Telegram-bot for automating the educational process, Curr. Probl. Teach. Math. Tech. Univ., no. 10, pp. 53-56.

[2] Panok, V., Shevchenko, A., Nazar, M. M., Starkov, D., Mescheryakov, D. S., & Shevtsov, A. (2025). Methodological principles of educational and psychological chatbot development, Inf. Technol. Learn. Tools, vol. 2, no. 106, pp. 76-93.

[3] Google Cloud. (2026). Vision API: OCR language support. [Online]. Available: https://docs.cloud.google.com/vision/docs/languages.

[4] Ławrynowicz, A., Wróblewska, A., Adrian, W. T., Kulczyński, B., & Gramza-Michałowska, A. (2022). Food recipe ingredient substitution ontology design pattern, Sensors, vol. 22, no. 3, p. 1095.

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

[6] Zhao, W. X. et al. (2023). A survey of large language models. [Online]. Available: https://arxiv.org/abs/2303.18223.

[7] Dagdelen, J. et al. (2024). Structured information extraction from scientific text with large language models, Nat. Commun., vol. 15, no. 1, p. 1418.

[8] OpenCV. (2026). Open Source Computer Vision Library. [Online]. Available: https://opencv.org/.

[9] Bohdan, N., Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, Int. J. Acad. Inf. Syst. Res., vol. 9, no. 10, pp. 7-18.

[10] OCRBot. (2026). Unofficial Telegram OCR robot. [Online]. Available: https://telegram.me/ocrbot.

[11] Argus. (2026). Photo text recognition bot. [Online]. Available: https://t.me/argus_ocr_bot.

[12] Culinary Recipes. (2026). Telegram bot. [Online]. Available: https://t.me/tasty_cooking_bot.

[13] AI Chef Bot. (2026). Telegram bot. [Online]. Available: https://t.me/Best_AI_Chef_Bot.

[14] Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks, Int. J. Acad. Eng. Res., vol. 7, no. 9, pp. 64-72.

[15] Gangania, P., Mishra, S., Garg, S., & Agarwal, S. (2018). Handwriting recognition system using optical character recognition, Int. J. Sci. Res. Comput. Sci. Eng., vol. 6, pp. 18-21.

[16] Christian, I., & Kusuma, G. P. (2023). Improving OCR performance on low-quality image using pre-processing and post-processing methods, Int. J. Eng. Trends Technol., vol. 71, no. 6, pp. 396-405.

[17] SpaCy. (2026). Industrial-strength natural language processing in Python. [Online]. Available: https://spacy.io/.

[18] Pitsilou, V., Papadakis, G., & Skoutas, D. (2024). Using LLMs to extract food entities from cooking recipes, in Proc. IEEE 40th Int. Conf. Data Eng. Workshops (ICDEW), pp. 21-28.

[19] NLTK. (2026). Natural language toolkit. [Online]. Available: https://www.nltk.org/.

[20] Nguyen, Q. N., Sidorova, A., & Torres, R. (2022). User interactions with chatbot interfaces vs. menu-based interfaces: An empirical study, Comput. Hum. Behav., vol. 128, p. 107093.

[21] Bashurov, V., & Safonov, P. (2025). Enhancing university education with AI: a Telegram bot leveraging RAG and external APIs for secure knowledge retrieval, Issues Inf. Syst., vol. 26, no. 3, pp. 413-420.

[22] Aiogram. (2026). Aiogram 3 documentation. [Online]. Available: https://docs.aiogram.dev/en/dev-3.x/.

[23] Google. (2026). Gemini API documentation. [Online]. Available: https://ai.google.dev/gemini-api/docs.

[24] PostgreSQL. (2026). PostgreSQL documentation. [Online]. Available: https://www.postgresql.org/docs/.

[25] Neudecker, C. et al. (2021). A survey of OCR evaluation tools and metrics, in Proc. 6th Int. Workshop Hist. Doc. Imaging Process., pp. 13-18.

[26] Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks, Int. J. Acad. Appl. Res., vol. 7, no. 11, pp. 134-145.

[27] Polat, F., Tiddi, I., & Groth, P. (2025). Testing prompt engineering methods for knowledge extraction from text, Semant. Web, vol. 16, no. 2, SW-243719.

[28] Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images, Int. J. Acad. Appl. Res., vol. 7, no. 9, pp. 57-70.

[29] Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, Int. J. Acad. Inf. Syst. Res., vol. 7, no. 7, pp. 25-36.

[30] Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks, Int. J. Acad. Eng. Res., vol. 7, no. 9, pp. 64-72.

[31] Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., Hudáková, M., & 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.

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

[33] Gorokhovatskyi, V., Tvoroshenko, I. & Yakovleva, O. (2024). Transforming image descriptions as a set of descriptors to construct classification features, Indones. J. Elec. Eng. Comput. Sci., vol. 33, no. 1, pp. 113-125.

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

[35] Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification, IEEE Access, vol. 10, pp. 124738-124746.

[36] Larin, I., Tvoroshenko, I., Gorokhovatskyi, V., & Liubchenko, V. (2025). Research and comparison of facial color normalization methods relative to an etalon image, Int. J. Acad. Appl. Res., vol. 9, no 11, pp. 36-47.

[37] Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, Int. J. Eng. Inf. Syst., vol. 6, no. 2, pp. 40-48.

[38] Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features, Comput. Mater. Contin., vol. 72, no. 1, pp. 1785-1797.

[39] Ayaz, A. M., Gorokhovatskyi, V., Tvoroshenko, I., Vlasenko, N., & Khalid, M. S. (2021). The research of image classification methods based on the introducing cluster representation parameters for the structural description, Int. J. Eng. Trends Technol., vol. 69, no. 10, pp. 186-192.

[40] Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods, Adv. Inf. Syst., vol. 9, no. 1, pp. 5-12.

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