http://doi.org/10.35784/iapgos.9125 *received: 31.01.2026 | revised: 07.06.2026 | accepted: 19.06.2026 | available online: 30.06.2026*

KNOWLEDGE MODEL "TAGS ABOUT BATCHES AND CONTAINERS" OF THE ERP SYSTEM "PlasmIS" WITH THE POSSIBILITY OF SELF-IMPROVEMENT USING LOCAL LLM MODELS

Oleh Bisikalo[1], Valerii Starzhynskyi[1], Tetiana Molodetska[1], Nelia Burlaka[2]
[1]Vinnytsia Technical National University, Vinnytsia, Ukraine, [2]Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, Vinnytsia, Ukraine

Abstract. *The main requirements, capabilities and functionality of the "Tags about batch and containers" module of the "PlasmIS" information system are recearched in the article. Mathematical models of the intelligent module including offered database and knowledge base are suggested. The principles of printing actual tags were considered and the method of printing tags from the information system was developed. An approach to automated standardization of technical names of hardware using local large language models (LLM) is considered, which allows to significantly increase the accuracy of processing specialized data. The architecture of interaction of Python scripts with the local Lm studio server is described in detail, including the organization of API requests, the structure of input and output data in JSON format, as well as mechanisms for saving results in the SQLite database.*

Keywords: intelligent module, information system, database, knowledge base, ERP, PlasmIS

MODEL WIEDZY "TAGI DOTYCZĄCE PARTII I POJEMNIKÓW" W SYSTEMIE ERP "PlasmIS" Z MOŻLIWOŚCIĄ SAMODOSKONALENIA Z WYKORZYSTANIEM LOKALNYCH MODELI LLM

Streszczenie. *W artykule omówiono główne wymagania, możliwości i funkcjonalność modułu „Tagi dotyczące partii i pojemników" systemu informatycznego „PlasmIS". Zaproponowano modele matematyczne inteligentnego modułu, w tym proponowaną bazę danych i bazę wiedzy. Rozważono zasady drukowania rzeczywistych etykiet i opracowano metodę drukowania etykiet z systemu informatycznego. Rozważono podejście do automatycznej standaryzacji nazw technicznych sprzętu przy użyciu lokalnych modeli językowych (LLM), co pozwala znacznie zwiększyć dokładność przetwarzania danych specjalistycznych. Szczegółowo opisano architekturę interakcji skryptów Python z lokalnym serwerem Lm studio, w tym organizację żądań API, strukturę danych wejściowych i wyjściowych w formacie JSON, a także mechanizmy zapisywania wyników w bazie danych SQLite.*

Słowa kluczowe: moduł inteligentny, system informacyjny, baza danych, baza wiedzy, ERP, PlasmIS

Introduction

In the over the past 20 years, business applications and programs have evolved from management information systems to enterprise information systems that offer decision support for enterprise resource planning. Enterprise resource planning is an activity that includes procurement, production, project, human resources, finance and provides accurate real-time data to help managers or executives make informed and effective decisions. The purpose of such planning is to organize information flows between all business processes within the enterprise and to effectively manage all its divisions for the entire set of functions and tasks.

The rapid development of information technologies of data storage, communications and processing allows storing and processing all information in cyberspace.

Enterprise Resource Planning (ERP) software attempts to integrate all manufacturing and service departments with their inherent functions into a single intelligent computer system to address the various problems and meet the diverse needs of these departments.

Implementing a single system that meets the needs of different departments (for example, finance, warehouse, HR) is a difficult task, as these departments have individual software that is designed and optimized according to the work of each department. However, if implemented correctly, this integrated approach can be very effective for an enterprise from an economic opinion. Based on a single ERP information system, different departments can easily exchange data and another useful information with each other [8].

Despite all the technological breakthroughs in this direction, there are numbers of tasks that global IT companies are currently working on, for example: scaling databases by size (up to petabytes), automating the design and administration of databases and ERP systems, automatic detection of data trends, structures and anomalies (data mining, data analysis, in particular intelligent, etc.) [10]. The task of creating separate modules within the existing ERP system of the enterprise is urgent, which allows increasing the level of automation of certain types of work and, as a result, to achieve measurable economic advantages of the enterprise as a whole [7].

Data management and decision-making are complex tasks that require considerable attention and responsibility. That's why today's businesses are quickly abandoning legacy solutions for resource planning, sales management, marketing, HR and finance. More than half of enterprises consider ERP as one of the priority sectors for investment. The ERP software market is growing at a rate of 10% annually, and its total share is expected to reach $90 billion by 2028 [5, 15].

"PlasmIS" is a large multi-level ERP system, which was developed on the basis of the requirements of the plants of the group of companies PrJSC "PlasmaTec". This is the largest corporation for the production of raw materials and a full cycle of welding electrodes in Ukraine. In 2011, the "PlasmIS" ERP system was created. Since that time, the system is in the stage of use and constant updating and improvement based on the constant development of production at the factories of PrJSC "PlasmaTec". The operation of this system covers all stages from initial production to transportation of finished products, taking into account production processes at each of the corporation's plants.

The production of electrodes is impossible without the supply and production of special raw materials. When the raw material arrives at the factory or plant, it must be quickly entered into the system with one-time printing of a special tag to mark the container in which the raw material is located. For this, the task was set in the development of the "Tags about batch and containers" module, which will facilitate and automate the process of receiving raw materials at the plant.

The "Tags about batch and containers" module is an automated intelligent unit of the information system related to warehouse accounting. The main requirements that were set for the section are the collection, analysis and storage of individual data, followed by combining them into a comprehensive table for convenient use. In addition to the analysis of these data, special attention was paid to the printing of tags. All collected information is collected using certain technologies in tags that can be conveniently and quickly printed using the interface and special functions – this is the aim of our research.

In parallel with ERP development, the modern industrial environment faces the challenge of unifying technical product

names, particularly for standardized parts such as bolts, nuts, and screws. Multilingual consistency and compliance with technical standards (DIN, ISO, DSTU) are essential for global supply chain integration and automated catalog management. Traditional neural machine translation (NMT) systems, while effective in general contexts, often fail to preserve technical accuracy and structural consistency [1].

Recent advances in large language models (LLM) have opened new opportunities. LLMs can simultaneously perform format standardization and multilingual translation using a unified neural architecture. For example, the LLM-BT approach employs back-translation to automatically verify and stabilize terminology. Furthermore, the deployment of local LLM servers offers data security, autonomy, and cost efficiency for enterprises operating without reliance on cloud APIs [14]. In this regard, our work considers the architecture of an autonomous system based on Python scripts, a local LM Studio server and a SQLite database, which provides a full cycle: format standardization → multilingual translation → storage → validation. An approach is justified and investigated that combines prompt engineering (model prompt generation), format validation logic (DIN/ISO → size → translation) and result quality metrics that allow quantitatively assessing the correctness of the source names.

The purpose of the research is to:

- increase the level of automation of the PlasmaTec enterprise by developing an intelligent module as part of the PlasmIS ERP system;
- ensure the possibility of self-improvement of the Batch and Container Tags module based on the use of local large language models (LLM);
- optimize warehouse operations by reducing their execution time and reducing personnel errors:
- increase the accuracy of standardization of hardware descriptions and multilingual translation of technical terms.

1. Development of automated tag printing models integrated with local LLM-based knowledge systems

Formal relations [2] in the context of the information system, which allow us to consider the information component [3] of the "Tags about packages and containers" module from the point of view of the relational data model, are:

$$RE = \{item, item\_part, barcode, item\_storage, item\_unit\_storage\} \quad (1)$$

where, *Item* – relation for the description of material values; the attributes that make up this relationare: *itm_id* (ID code of a certain material value), *itm_unt* (the basic unit of measurement in which this material value is measured), *itm_mrk* (the mark of the material value), *itm_name* (the name of the material value in Ukrainian), *itm_name_en* (the asset in English), *itm_comment* (notes), *itm_rec_dt* (the date of the last changes to the information on the individual asset), *itm_rec_st* (current status) etc.

$$Item \subset itm\_id \times itm\_itmg \times itm\_unt \times itm\_mrk \times itm\_code \times itm\_inv \times itm\_inv\_b \times itm\_name \times itm\_name\_eng \times itm\_comment \times itm\_rec\_dt \times itm\_rec\_st \times itm\_obi\_ts \quad (2)$$

Item_part – relation for characterizing lots; the attributes that make up this relation are: *itpt_id* (ID code of a batch of a certain material value), *itpt_dpr* (ID code of a material value), *itpt_num* (batch number), *itpt_dt* (date of batch creation), *itpt_st* (current status), *itpt_comment* (comments), *tpt_crtf* (certificate), etc.

$$Item\_part \times itpt\_id\ itpt\_dpr \times itpt\_itm \times itpt\_itm\_equip \times itpt\_ca \times itpt\_crtf \times itpt\_mrk \times itpt\_kind \times itpt\_num \times itpt\_num\_org \times itpt\_dt \times itpt\_st \times itpt\_comment \times itpt\_obi\_ts \quad (3)$$

Barcode – a relationship for characterizing basic data about the batch, container. A tag is formed from this table. The attributes that make up this relation are: *bc_id* (ID code of a certain lot, container), *bc_itpt* (id code of a lot of material value), *bc_grp* (group by lot, container), *bc_count* (quantity/weight), *bc_comment* (comments), *bc_rec_st* (current status), *bc_rec_dt* (record date), *bc_dt* (date), *bc_seal* (seal) etc.

$$Barcode \subset bc\_id \times bc\_itpt \times bc\_ist \times bc\_grp \times bc\_count \times bc\_comment \times bc\_rec\_st \times bc\_rec\_dt \times bc\_obi\_ts \times bc\_ius \times bc\_dt \times bc\_seal \times bc\_st \quad (4)$$

Item_unit_storage – relation for the characteristics of storage units; the attributes that make up this relationship are: *ius_id* (ID code of a specific storage unit), *ius_name* (name of a specific storage unit), *ius_koef* (coefficient of the storage unit), *ius_rec_st* (current status), *ius_rec_dt* (recording date), *ius_comment* (notes), *ius_name_en* (name of a specific storage unit in English) etc.

$$Item\_unit\_storage \subset ius\_id \times ius\_itm \times ius\_unt \times ius\_unt\_rpt \times ius\_mrktr \times ius\_pck \times ius\_name \times ius\_name\_eng \times ius\_coef \times ius\_is\_use\_price \times ius\_unt \times ius\_descript \times ius\_comment \times ius\_rec\_st \times ius\_rec\_dt \times ius\_obi\_ts \quad (5)$$

Item_storage – relation for the characteristics of storage places; the attributes that make up this relation are: *ist_id* (ID code of a specific storage location), *ist_tp* (storage location type), *ist_num* (storage location number), *ist_name* (storage location name), *ist_comment* (comments), *ist_rec_st* (current status), *ist_rec_dt* (record date) etc.

$$Item\_storage \subset ist\_id \times ist\_ist \times ist\_dpr \times ist\_itm \times ist\_plt \times ist\_tp \times ist\_num \times ist\_name \times ist\_weight \times ist\_comment \times ist\_rec\_st \times ist\_rec\_dt \times ist\_obi\_ts \times ist\_pck \quad (6)$$

All tables are linked. The main barcode table has links with other tables: with the *item_part* table there is a (one-to-many) link based on the *bc_itpt* field – from the lot of the storage location, with the *item_storage* table there is a (one-to-many) link based on the *bc_ist* field – item storage id, with the *item_unit_storage* table there is a (one-to-many) relationship by field *bc_ius* – storage unit id.

The item table has relationships with the following tables: with the *item_part* table, a (one-to-many) relationship on the *itpt_itm_equip* field – equipment and a (one-to-many) relationship on the *itpt_itm* field – item id (equipment), with the *item_storagetable* there is a (one-to-many) connection by the *ist_itm* field – the item id (storage location), with the *item_unit_storage* table there is a (one-to-many) connection by the *ius_itm* field – the item id (storage unit).

In Figure 1, the considered relations of the label printing model (1) are presented in the form of a database scheme.

The model of the knowledge base of the "Batch-Container Tags" module defines the logical and functional structure for storing, processing, and updating data related to tag generation and automation within the ERP system.

A tag is a table that is compiled through HTML code and defined through field names. In general, field names change depending on the company [11]. Using the ID of the tape from the barcode table, values are substituted into special variables, which later form the tag. For this, a knowledge base of the production type [12] of the intelligent module "Tags about batch and containers" was developed, which consists of a set of production rules, predicates, functions and operators [4, 9]. In particular:

1. Tags are being printed (function PrintL) the possibility of replacing the factory logo in the upper part of the tag is taken into account. For each enterprise, it is necessary to ensure that the logo of this enterprise is obtained from a separate database. So, if the logo l i any i enterprise in the system, a universal logo is selected for printingl univ ("PlasmaTec companies group"), otherwise a personal logo is selected l i . Products are defined as follows LogoR:

$$\text{If } A\ (l_i) \text{ then } Print\ (l_i) \text{ else } PrintL\ (l_{univ}) \quad (7)$$

where the predicate $A\ (l_i)$ is defined as follows:

$$\exists l_i \mid i \in \{1,2,\dots N\} \rightarrow A(l_i) = True. \quad (8)$$

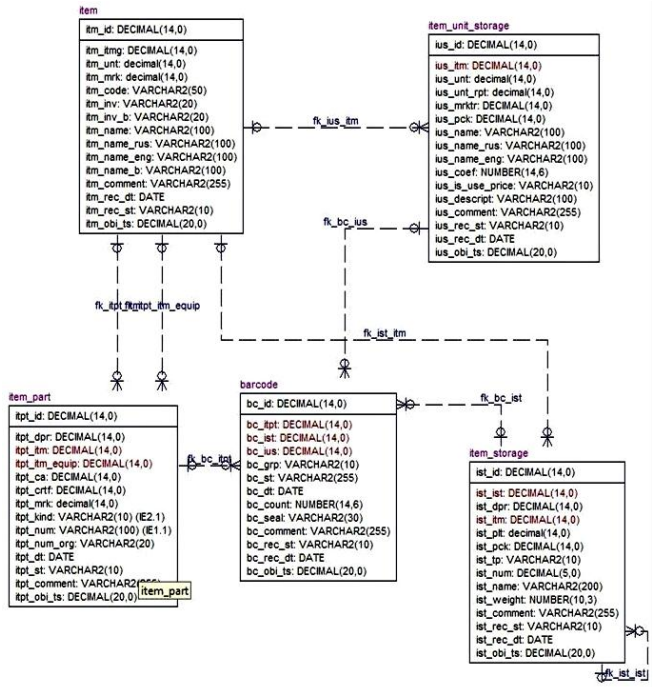

Fig. 1. Representation of label printing model relations in the form of a database schema

2. The system database has a variable named Form. In this variable in case of activating the print function PrintF for j division of the enterprise with M (e.g. drawing area, packing area, warehouse, etc.) the form number is inserted. If there is no form number in the print parameters, then employees have the opportunity to write it down, but in the case of printing without a form number, the value of the variable Fas the field on the tag is not inserted. Products are defined as follows FormaR:

$$\text{If } B\ (f_j) \text{ then } Print\ (f_j) \text{ else } PrintF\ (Null) \quad (9)$$

where the predicate $B\ (f_j)$is defined as follows:

$$\exists f_j \mid j\ \in \{1,2,\dots M\}\ \rightarrow B(f_i) = True \quad (10)$$

3. System users can independently edit the fields in the tag that they need. In order to maintain flexibility, drop-down lists have been developed in the print window: "Do not print" and „Do not insert". Through the „Do not print" menu, users can choose j lines r j from LR table tag, which do not need to be printed through the function PrintRow, and through the "Do not insert" menu, there is an opportunity not to insert values into certain one's j lines with LF table tags by function PrintField. Products are defined as follows RowR and FieldR:

$$\text{If } Row_j \text{ then } PrintRow\ (r_j) \text{else } PrintRow\ (Null) \quad (11)$$

where the predicate is defined as follows:

$$\exists\ Row_j \mid j\ \in \{1,2,\dots LR\}\ \rightarrow\ Row_j\ = True \quad (12)$$

$$\text{If } Field_j \text{ then } PrintField\ (r_j) \text{ else } PrintField\ (Null) \quad (13)$$

where the predicate is defined as follows:

$$\exists\ Field_j \mid j \in \{1,2,\dots LF\}\ \rightarrow\ Field_j\ = True \quad (14)$$

4. For universal printing of the tag on different printers, a method of turning the tag itself on was developed 90°. For this, a separate drop-down list with options was created 0° and 90°; in js-file the method of turning the tag on is laid down 90°. If the option is selected by the user 90° (predicate Turn defined as True), the production rule TurnR tag rotation via function PrintTurn looks like:

$$\text{If } Turn \text{ then } PrintTurn\ (Go) \text{ else } PrintTurn\ (Null) \quad (15)$$

where variable is

$$Go = ".css('transform',`rotate(\${this.value()}deg)`) \quad (16)$$

5. To canter the tag on the paper (move it to the right or left), the Indentation field is provided in the database (*indent*). This repositions the tag horizontally by specifying the number of pixels in dots (*pixel_count*), to which you want to move the tag. Shift operator:

$$O_{pc}\colon (indent) \rightarrow pixel\_count \quad (17)$$

6. One of the most important elements of a tag is a barcode. It is developed through a special function that generates a visual representation of the barcode *BarcodeImage*. The barcode itself is formed from three fields: *id* part material value (*id_part*), id storage of things (*id_goods*) and id barcode (*id_barcode*). Barcode printing operator:

$$O_b\colon (id\_part, id\_goods, id\_barcode) \rightarrow BarcodeImage \quad (18)$$

So, as a result of the study, a model of the knowledge base [2, 3] of the intellectual module "Tags about batch and containers" was developed:

$$KnowledgeBase = < Rule, Pr, Func, Op > \quad (19)$$

consisting of production rules (7), (9), (11), (13), (15):

$$Rule = \{LogoR, FormaR, RowR, FieldR, TurnR\} \quad (20)$$

Predicates (8), (10), (12), (14), (16):

$$Pr = \{A(l_i), B(f_j), Row_j, Field_j, Turn\} \quad (21)$$

functions:

$$Func = \{PrintL, PrintRow, PrintField, PrintTurn\} \quad (22)$$

And operators (17)–(18)

$$Op = \{O_{pc}, O_b\} \quad (23)$$

The use of local LLM models enables the implementation of intelligent knowledge base functions, ensuring autonomous data processing, contextual reasoning, and semantic consistency within the ERP environment.

As warehouse practice shows, one of the most problematic issues in terms of compliance with international standards is the correct filling of such data attributes as *itm_name* and *itm_name_en*. This rather autonomous and narrow task requires automated standardization and multilingual translation of technical names. For example, the task of standardizing hardware – such as bolts, nuts, screws, etc. – is complemented by the need for their multilingual translation to ensure data integrity for the purpose of integration into international supply chains and automation of catalogue replenishment. The proposed structure of the knowledge base "Tags about batches and containers" of the product type allows you to flexibly create and apply the necessary functions based on AI.

Formally, we introduce an additional product Update for the *j*-th warehouse product with N:

$$\text{If } Validate_j \text{ then } PostPrompt(Null) \text{else}$$
$$PostPrompt(itm_{name\,j}, itm\_name\_en_j) \quad (24)$$

which we add to the list (20), while the expression for the knowledge base (19) remains unchanged. The predicate $Validate_j$ is defined as:

$$Validate_j = \begin{cases} True, \text{if } itm_{name\,j} = itm_{name\,j}' \\ \text{or } itm\_name\_en_j = itm\_name\_en_j' \\ False, \text{if } itm_{name\,j} \neq itm_{name\,j}' \\ \text{or } itm\_name\_en_j \neq itm\_name\_en_j' \end{cases} \quad (25)$$

In turn, the $PostPrompt$ function and the values of itm_name_j', $itm\_name\_en_j'$ are determined based on the application of LLM:

$$PostPrompt\colon LLM\left(D_q^V, q\left(itm_{name\,j}, itm_{name_{en\,j}}\right)\right) \rightarrow$$
$$\{itm\_name_j', itm\_name\_en_j'\} \quad (26)$$

where, D_q^V – is a training dataset of $V < N$ queries of size $V < N$, and q – is a query to LLM based on values $itm\_name_j, itm\_name\_en_j$.

2. Principles and results of LLM training on domain data

Training large language models (LLMs) is a key step in ensuring their ability to efficiently solve specific tasks in limited professional areas, such as specific technical terminology and standards. General models trained on large open corpora demonstrate good versatility, but often inadequately handle highly specialized terms and format requirements. Three main approaches are widely used to solve this problem: Retrieval-Augmented Generation (RAG) – to provide access to specialized knowledge in real time; Parameter-Efficient Fine-Tuning (PEFT), for example, LoRA – which allows adapting the model without full retraining; and classical fine-tuning – targeted retraining on domain data[13].

Full retraining requires large-scale computing resources and large amounts of data, which makes it impractical for tasks with a relatively narrow corpus. The RAG approach allows enriching answers with external knowledge bases, but it is more focused on dynamic search and does not guarantee adherence to a clear formatting pattern. In turn, PEFT methods, in particular supervised fine-tuning and LoRA, provide the ability to train the model on small domain data sets, while maintaining the generalization properties of the basic LLM. In the experiments conducted in this study, the PEFT method (using LoRA) was used to adapt the "mistral-7b-instruct-v0.2" model on a corpus of technical names of hardware, which included templates with standards (DIN, ISO, GOST), sizes and translation. More than a hundred prompt-completion examples were collected, where prompt contains a "raw" (usually not entirely accurate) name, and completion – a standardized name with translation. This made it possible to accelerate the retraining, which took only a few hours on typical GPUs, while ensuring a noticeable reduction in the number of errors in name formatting (compared to the base model), without a significant loss of general language abilities.

The principle of retraining (training) of the "mistral-7b-instruct-v0.2" model was carried out through the name standardization method. The main task was to accurately format the name according to the specified template. As planned in the research problem statement, the exchange of input and output data with the model was implemented in the format of JSON requests. In particular, the model receives JSON from objects as input data, with each object containing the keys: *itm_id* (unique record code), *onms_value* (name format template), *itm_name* (name of hardware). Table 1 shows the tasks and answers to the tasks before and after training. The tasks and answers are presented in the format of JSON objects.

Using the Python programming language, a database was created from a single table. A special query selects records from the database, generates a prompt (task) for each and sends a POST request to the local LM Studio server with the model. The server runs the model, which processes the data and returns JSON with the result. Python parses the JSON, obtains standardized names and translations, and updates these values in the database table.

To visualize the solution architecture, a Component/ Deployment Diagram was created, which displays the connection of the local LM Studio environment, the Python client, and the local model server [6]. It is presented in Fig. 1. The Component/Deployment diagram demonstrates the relationship of components, data flows between them, and software deployment locations on the local computer.

This diagram reveals the architecture of the system and its key components, including:

- Python Script (process_items.py) – the main script that starts processing records, forms queries for the model and stores the results in the database.
- Database (items.db) – a local database that stores hardware records with the fields raw_name, ukr_name, eng_name. The Python script accesses it via SQL queries (SELECT, UPDATE).
- LLM Local Server (LM Studio) – a local server that executes the mistral-7b-instruct-v0.2 model. It accepts JSON queries via HTTP POST and returns a JSON response with the fields ukr_name and eng_name.
- Validation Script (validate_items.py) – an additional script that checks the correctness of formatting and compiles statistics on incorrect records.

A significant role in the conducted research was played by the development of the prompts necessary for further training the model. The main task of prompt is to maximally clarify the mistral-7b-instruct-v0.2 model instruction on standardization of hardware names. Example prompt No. 1 is shown in Fig. 2.

In order to streamline the data exchange between the user and the model, a Sequence Diagram was created, which displays the sequence of requests and responses in JSON format. The Sequence Diagram (Fig. 3) demonstrates how the Python client sends input data to the local model server, how the request is processed, and how the response with standardized names and translation is generated.

In order to streamline the data exchange between the user and the model, a Sequence Diagram was created, which displays the sequence of requests and responses in JSON format. The Sequence Diagram (Fig. 3) demonstrates how the Python client sends input data to the local model server, how the request is processed, and how the response with standardized names and translation is generated.

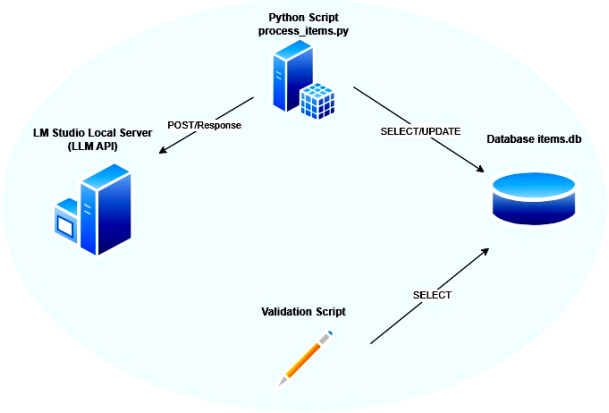

Fig. 1. Component / Deployment Diagram (connection diagram)

Fig. 2. Activity diagram (workflow)

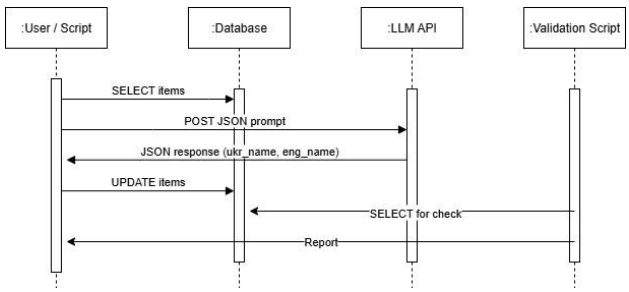

Fig. 3. Sequence diagram (data exchange with the model)

```
prompt = (
    f"Ти – модель для стандартизації назв метизів. Задача стандартизувати назву метиза і зробити переклад її на 2 мови. "
    f"Вхід(приклад): 'Болт оцинкований М10х120 DIN 933'. "
    f"Вихід(приклад): ukr_name: 'Болт DIN 933 М10х120 оцинкований'; eng_name: 'Bolt DIN 933 M10x120 galvanized'. "
    f"Тепер стандартизуй: '{raw_name}' у форматі '<назва виробу> <стандарт> <розмір> <додаткові параметри при наявності>'. "
    f"Відповідь у JSON без додаткових полів, одразу ukr_name, eng_name."
)
```

Fig. 4. Example of the prompt No 1

```
pattern = re.compile(
    r"^(болт|гайка|шайба|цвях|шуруп|саморіз|гвинт|винт|шпилька)\s+"  # назва
    r"(DIN|ГОСТ|ISO)\s*\d+"                                          # стандарт
    r"(?:\s+[A-ZA-Я]?\d+(?:x\d+)*)*"                                 # розмір/параметри (гнучкі)
    r"(.*)?$",                                                       # решта
    re.IGNORECASE
)
```

Fig. 5. Condition for checking the accuracy of a standardized name

```
prompt = (
    f"Ти – модель для стандартизації назв метизів. Задача стандартизувати назву метиза і зробити переклад її на 2 мови. "
    f"Вхід(приклад): 'Болт оцинкований М10х120 DIN 933'. "
    f"Вихід(приклад): ukr_name: 'Болт DIN 933 М10х120 оцинкований'; eng_name: 'Bolt DIN 933 M10x120 galvanized'. "
    f"В конкретному прикладі Болт - Назва виробу(може бути Гайка, Шайба тощо), DIN 933 - Стандарт(може бути ГОСТ та ISO), М10х120 - розмір виробу, оцинкований - інші додатковий параметр"
    f"Тепер стандартизуй: '{raw_name}' у форматі '<назва виробу> <стандарт> <розмір> <додаткові параметри при наявності>'. "
    f"Забезпеч, щоб переклади були точними: "
    f"- ukr_name українською, "
    f"- eng_name англійською. "
    f"Відповідь у JSON без додаткових полів, одразу ukr_name, eng_name."
```

Fig. 6. Example of the prompt No 2

To visualize the sequence of actions and the data processing workflow, an Activity Diagram (Workflow) was created. The Activity Diagram (Workflow) presented in Fig. 3 illustrates the key stages of technical name processing: sending input data to the model, obtaining standardized results, and writing them to the database.

A significant role in the conducted research was played by the development of the prompts necessary for further training the model. The main task of prompt is to maximally clarify the mistral-7b-instruct-v0.2 model instruction on standardization of hardware names. Example prompt No. 1 is shown in Fig. 4.

The accuracy check is implemented by splitting the updated Ukrainian name into sections, which results in a name format and allows you to compare the updated name with the template. The name format has the following form: <name> <standard> <standard number> <size> [options]. The code for checking accuracy using a script is implemented in Python and is shown in Fig. 5.

Based on prompt No 1, the model was tested using 100 records that were entered into the database items. A sample of 10 records is shown in Table 1 for illustration.

Table 1. Comparison of results using prompt No 1

Name of hardware	Updated entry	Record processing time
Bolt M6x20 DIN 933	DIN 933 M6x20 Bolt, Oiled	9.70 s
Screw M10 DIN 934	M10 DIN 934 Nut	7.83 s
Bolt M12x69 DIN 444	M12x69 bolt DIN 444	9.94 s
Screw 8 DIN 934	nut DIN 934 8	10.21 s

The total processing time for 100 records was 998.46 seconds (~17 minutes 4 seconds). The result of checking the correctness of the results through prompt No 1 with a total of 100 records is 67 (67%) incorrect records.

Table 2 shows a comparison of the results with the template executed by the model using prompt No 1. Template: <name> <standard> <standard number> <size> [options].

Table 2. Comparison of results using prompt No 1

No	Updated entry	Notes
1	Bolt DIN 933 M6x20 оцинкований DIN 933 M6x20 Bolt, Oiled	Bolt DIN 933 M6x20 Oiled
2	Screw M10 DIN 934 M10 DIN 934 Nut	Nut DIN 934 M10)
3	Bolt M12x69 DIN 444 M12x69 bolt DIN 444	bolt DIN 444 M12x69)
4	Screw DIN 934 8 nut DIN 934 8	matches the pattern

Prompt No 1 was refined to improve the accuracy of standardization and translation, after which the model was retested, resulting in prompt No 2. An example of prompt No 2 is shown in Fig. 6.

Based on prompt No 2, the model was tested using 100 records that were entered into the items.db database. A sample of 10 records is shown in Table 3 for illustration.

Table 3. Using the prompt No 2

Name of hardware	Updated entry	Record processing time
Bolt M10x40 DIN 912	Bolt DIN 912 M10x40	9.94 s
Bolt M12x85 DIN 603	Bolt DIN 603 M12x85	10.14 s
Screw M12 DIN 934	Screw DIN 934 M12	8.99 s
Washer 12 DIN 125	Washer DIN 12 M12	8.26 s

Total processing time for 100 records: 987.59 seconds (~16 minutes 45 seconds). The result of checking the correctness of the results through prompt No 2 with a total of 100 records is 0 (0.0%) incorrect records. An example of the query execution is shown in Fig. 7.

Table 4 shows a comparison of the results with the template executed by the model using prompt No 2. Template: <name> <standard> <standard number> <size> [options].

Table 4. Comparing results using prompt No 1

No	Updated entry	Notes
1	Bolt DIN 912 M10x40	matches the template
2	Bolt DIN 603 M12x85	matches the template
3	Screw DIN 934 M12	matches the template
4	Bolt ISO 7380 M6x25	matches the template

3. Conclusions

The research presented in this article demonstrates the effectiveness of developing an intelligent module within the ERP system "PlasmIS," namely the "Tags about Batch and Containers" module, which significantly increased the level of automation in the enterprise "PlasmaTec". The implementation of the proposed relational model (1-6) and knowledge base structure (7-26) enabled the optimization of warehouse operations [15], reducing operation time from 2.5 to 4 times and cutting personnel errors by approximately 50%.

The developed knowledge base functions as an online library add-on, organizing data about products and processes into clearly structured topics and subtopics. This approach ensures convenient access, management, and sharing of information within the enterprise.

In addition, the study substantiated the efficiency of local large language models (LLM) for automated standardization and multilingual translation of technical terms. After fine-tuning the Mistral-7B-Instruct model on a domain-specific corpus and designing an optimized prompt (prompt No 2), the accuracy of name processing reached 100%, confirming the validity of the proposed approach for local, autonomous use in applied ERP systems. In contrast, using the unrefined prompt No 1 resulted in 67% incorrect responses, highlighting the crucial role of proper prompt engineering and high-quality training data.

The results prove that integrating local LLM inference with fine-tuning and precise prompt design ensures full compliance with international naming standards (e.g., DIN, ISO, DSTU) and enables reliable multilingual translations. The proposed methodology of incorporating local LLMs into technical data processing workflows is therefore an effective, scalable, and secure solution that can serve as a foundation for future implementations in industrial and analytical systems.

Disclosures

The authors declare no conflicts of interest.

Ethics approval and consent to participate

This study was conducted in accordance with the principles of the Declaration of Helsinki, and in compliance with the International Conference on Harmonization-Good Clinical Practice and local regulatory requirements. Ethical approval was obtained from the Ethics Committee (protocol No 7, 16.05.2024) of the Vinnytsia National Technical University, (Vinnytsia, Ukraine).

References

[1] Ataman, D., Birch, A., Habash, N., Federico, M., Koehn, P., & Cho, K. (2025). Machine Translation in the Era of Large Language Models:A Survey of Historical and Emerging Problems. *Information*, *16*(9), 723. https://doi.org/10.3390/info16090723

[2] Bisikalo, O., Chernenko, D., Danylchuk, O., Kovtun, V., & Romanenko, V. (2020). Information Technology for TTF Optimization of an Information System for Critical Use that Operates in Aggressive Cyber-Physical Space. *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, 323–329. https://doi.org/10.1109/PICST51311.2020.9467997

[3] Bisikalo, O. V., Kovtun, V. V., & Kovtun, O. V. (2020). Modeling of the Estimation of the Time to Failure of the Information System for Critical Use. *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*, 140–143. https://doi.org/10.1109/ACIT49673.2020.9208883

[4] Kang, E.-Y., Choudhary, G., Campusano, M., Kühnrich, M., & Pedersen, A. (2024). *Enhancing Dependability of Industrial Robots: Security and Safety Assessments Based on Model-Driven Engineering*. 2024 11th International Conference on Dependable Systems and Their Applications (DSA), 106–113. https://doi.org/10.1109/DSA63982.2024.00024

[5] Cruz-Torres, W., Alvarez-Risco, A., & Del-Aguila-Arcentales, S. (2021). Impact of Enterprise Resource Planning (ERP) Implementation on Performance of an Education Enterprise: A Structural Equation Modeling (SEM). *Studies in Business and Economics*, *16*(2), 37–52. https://doi.org/10.2478/sbe-2021-0023

[6] Danilczuk, W., & Gola, A. (2020). Computer-aided material demand planning using ERP systems and business intelligence technology. *Applied Computer Science*, *16*(3), 42–55. https://doi.org/10.35784/acs-2020-20

[7] Estefania, T. V., Samir, L., Robert, P., Patrice, D., & Alexandre, M. (2018). The integration of ERP and inter-intra organizational information systems: A Literature Review. *IFAC-PapersOnLine*, *51*(11), 1212–1217. https://doi.org/10.1016/j.ifacol.2018.08.425

[8] Grabski, S. V., Leech, S. A., & Schmidt, P. J. (2011). A Review of ERP Research: A Future Agenda for Accounting Information Systems. *Journal of Information Systems*, *25*(1), 37–78. https://doi.org/10.2308/jis.2011.25.1.37

[9] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[10] Kocsis, D. (2019). A conceptual foundation of design and implementation research in accounting information systems. *International Journal of Accounting Information Systems*, *34*, 100420. https://doi.org/10.1016/j.accinf.2019.06.003

[11] Ofoegbu, C. A. (2016). Enterprise Resource Planning (ERP) adoption in a hybrid service and manufacturing Small and Medium-sized Enterprise (SME): An action case study [Ph.D. thesis]. University of Salford.

[12] Shah, K. (2025, May 16). How to Build Domain Specific LLMs? *Best Artificial Intelligence Blogs*. https://www.projectpro.io/article/domain-specific-llms/1111

[13] Tang, J. (2025, July 1). Fine-Tuning LLMs for Specific Domains: Why, How, and What to Consider. *Medium*. https://medium.com/@jamestang/fine-tuning-llms-for-specific-domains-why-how-and-what-to-consider-8b2fd5781615

[14] Weigang, L., & Brom, P. C. (2025). *LLM-BT-Terms: Back-Translation as a Framework for Terminology Standardization and Dynamic Semantic Embedding* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2506.08174

[15] Zafary, F. (2020). Implementation of business intelligence considering the role of information systems integration and enterprise resource planning. *Journal of Intelligence Studies in Business*, *1*(1). https://doi.org/10.37380/jisib.v1i1.563

Prof. Oleh Bisikalo
e-mail: obisikalo@vntu.edu.ua

Doctor of Engineering Sciences, Head of the Department of Automation & Intelligent Information Technologies, Vinnytsia National Technical University, Ukraine.
Scientific directions: information technology, innovation technologies

https://orcid.org/0000-0002-7607-1943

M.Sc. Valerii Starzhynskyi
e-mail: 3372292@gmail.com

Valerii Starzhynskyi is a postgraduate student of the Department of Automation and Intelligent Information Technologies. Vinnytsia National Technical University. Vinnytsia, Ukraine.
Scientific directions: information technology, innovation technologies.

https://orcid.org/0009-0009-3827-0122

Ph.D. Tetiana Molodetska
e-mail: molodecka@vntu.edu.ua

Ph.D., associated professor, Faculty of Mechanical Engineering and Transport, Vinnytsia National Technical University, Ukraine.
Scientific directions: information technology, innovation technologies.

https://orcid.org/0000-0001-5776-3648

Ph.D. Nelia Burlaka
e-mail: burlaka10@i.ua

Associate professor of the Department of Pedagogy, Professional Education and Management of Educational Institutions of the Vinnytsia Mykhailo Kotsiubynskyi State Pedagogical University, Ukraine.
Scientific directions: information technology, innovation technologies.

https://orcid.org/0000-0002-7424-2657