Scrum-Guided Development of an Intelligent Payroll Management System with Rule-Based Automation and RAG Chatbot Support

Feras Al-Hawari*, Anoud Alufeishat*, Mohammad Habahbeh*, Ahmad Alfalayleh**
*Digital Transformation Office**, *Finance Department***
German Jordanian University
Amman, 11180, Jordan

firas.alhawari@gju.edu.jo, anoud.alufieshat@gju.edu.jo, mhamad.hab@gmail.com, Ahmad.Alflayleh@gju.edu.jo

***Abstract**:* Managing payroll in large institutions involves complex tasks such as setup, salary calculation, tax filing, and record keeping. Manual spreadsheet administration is inefficient, error-prone, and costly. Digitalizing payroll is essential to enable automation, improve accuracy, transparency, and security. The complexity and evolving requirements of large-scale payroll systems make full upfront planning impractical. Scrum delivers the project in manageable increments, enabling continuous customer involvement and feedback, reducing rework and associated costs compared to Waterfall approaches, where rigid phases limit early visibility and adaptation. Accordingly, we implemented a rule-based payroll system using reusable template–resolver design patterns, supporting configurable payroll, tax, health insurance, and social security rules, employee-specific allowances, automated bulk and individual salary slip generation, retroactive calculations, and built-in audit controls. System reliability was validated using Scrum metrics, including burn-up and defect charts, to track progress and monitor quality. Bulk payroll processing over 19 months confirmed consistent and scalable operation. Building on this system, we introduce a retrieval-enhanced chatbot to assist employees with payroll regulations, tax rules, and health insurance, reducing HR support efforts. Three RAG-based LLMs were evaluated using FAISS and BM25 retrieval with varying embeddings, chunk sizes, and top-k parameters, with responses scored by expert reviewers. Pure LLMs performed worse than RAG models, highlighting the benefits of retrieval augmentation. The best-performing configuration achieved a mean reviewer score of 94.9% on a 50-question, domain-reviewed benchmark, demonstrating high response quality. System robustness was further validated through empirical red-teaming, indicating strong containment under tested adversarial scenarios.

Keywords: *Payroll systems, enterprise information systems, retrieval-augmented generation, large language models, design patterns, red-teaming.*

1. Introduction

One of the basic roles of the finance department at any institution is payroll. Payroll is the process of paying compensation to the employees of an institution periodically and on time. It involves tasks such as employee information input, payroll setup, salary computation, payment distribution, record keeping, report generation, and tax filing. In that respect, it is crucial to ensure the transparency, accuracy, and timeliness of payroll, as these factors positively impact employee morale and trust, which in turn strengthen the institution's reputation. Moreover, in large institutions with hundreds or even thousands of employees, handling salary computation manually or via spreadsheets becomes impractical, error-prone, inefficient, insecure, and costly [1]. To overcome these drawbacks, a computerized payroll system is essential to support complex, automatic, transparent, accurate, secure, and timely payroll operations while ensuring compliance with pay regulations and tax laws.

In that regard, several payroll software modules have been developed in-house at the German Jordanian University (GJU) to digitalize and streamline the various steps involved in the payroll management process. Specifically, the salary computation module calculates employee salaries in compliance with the *Academic Staff Regulation* and *Employees Regulation* at GJU [2], which align closely with those of public universities in Jordan. It is worth noting, however, that the fundamental payroll digitalization concepts presented in this work can be broadly applied to develop payroll systems for universities, industries, and other sectors.

Based on the adopted regulations, salary computation is a complex process that involves determining employee wages, allowances, taxes, social security, health insurance, and deductions. The calculation of certain salary slip items depends on various employee-specific parameters such as employee type (e.g., academic or administrative), grade, grade category, years in grade category, commissions (e.g., dean, chair, or director), family members, and employment status (e.g., currently employed, unpaid leave, or sabbatical leave). Since employee information is usually managed in the human resources system, the payroll modules have been unified with the human resources modules to form a single Human Resources and Payroll Information System (HRPIS) at GJU. This integration eliminates the need to enter employee data twice into separate systems, which reduces the risk of mismatched or erroneous data, prevents potential data breaches during transfer, and enhances overall payroll process efficiency.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Although digital payroll systems exist, the literature provides limited end-to-end accounts of their implementation or the challenges involved. The integration of rule-based payroll management with intelligent support, such as chatbots, remains underexplored. Building large-scale payroll systems requires months of development and cannot be effectively managed using rigid waterfall approaches, which often result in delays, higher costs, and errors. For these reasons, agile methodologies are essential to accommodate evolving requirements, modular development, and continuous user feedback.

Addressing these challenges is critical not only for payroll operations but also for broader intelligent enterprise systems, where knowledge-intensive processes, regulatory compliance, scalability, and user-centricity are essential. To meet these requirements, this study proposes a novel methodological framework (see Fig. 1) that integrates agile development, rule-based payroll automation, and intelligent chatbot support, along with system and chatbot validation, delivering a comprehensive end-to-end solution for knowledge-driven payroll management. The framework is organized around the following objectives:

- **Implement a flexible rule-based payroll engine setup** using knowledge-driven template and resolver design patterns. It integrates regulation tables, configurable health insurance, income tax, and social security rules, as well as employee-specific irregular items. Resolvers compute item-level values while reusable templates automatically link to employees, ensuring accurate, efficient, and scalable payroll management and providing a foundation for automated salary computation.
- **Enable automated payroll processing** to streamline operations for large employee groups through bulk and individual salary slip generation, retroactive salary calculations, and salary tax slip automation. Configurable rules guide all computations, and built-in cross-checks and audit controls maintain accuracy, ensure regulatory compliance, and reduce manual effort, enabling fast, reliable, and scalable payroll management.
- **Adopt a Scrum-based agile methodology** as it is essential for guiding the design and development of rule-based payroll modules and salary computation. Large, complex payroll systems cannot be delivered effectively with rigid waterfall methods, which often cause higher costs and errors due to delayed feedback, inflexible phase sequencing, and limited adaptability to changing requirements. Scrum enables iterative development, stakeholder engagement, and flexibility, ensuring a practical, reliable, and user-centered payroll solution.
- **Validate the payroll system** to ensure accuracy, timeliness, and cost-efficiency, support SLA compliance, and reinforce reliability and scalability. The system is designed to enable fast, precise, and scalable bulk salary slip generation, minimize errors, reduce administrative effort, and provide insights into staff workload and system usage.
- **Integrate a Retrieval-Augmented Generation (RAG) chatbot within the HRPIS** to provide accurate, context-aware support on HR and payroll matters, enhancing operational efficiency, regulatory compliance, and user satisfaction. The chatbot combines a Large Language Model (LLM) with domain-specific knowledge, such as GJU regulations and the Jordanian tax law, overcoming the limitations of pure LLMs and delivering timely, factually grounded, and context-sensitive responses.
- **Validate the chatbot** using an expert-designed question set across three LLMs with hybrid retrieval, two embedding methods, and different chunk sizes. Each configuration is run three times, with answers scored by two independent reviewers. Techniques such as query transformation and prompt engineering are applied to identify the configuration that produces the most accurate, factual, and privacy-preserving responses while minimizing hallucinations. The best-performing configuration is then evaluated on a comprehensive benchmark, and system robustness is further assessed through empirical red-teaming under adversarial conditions.

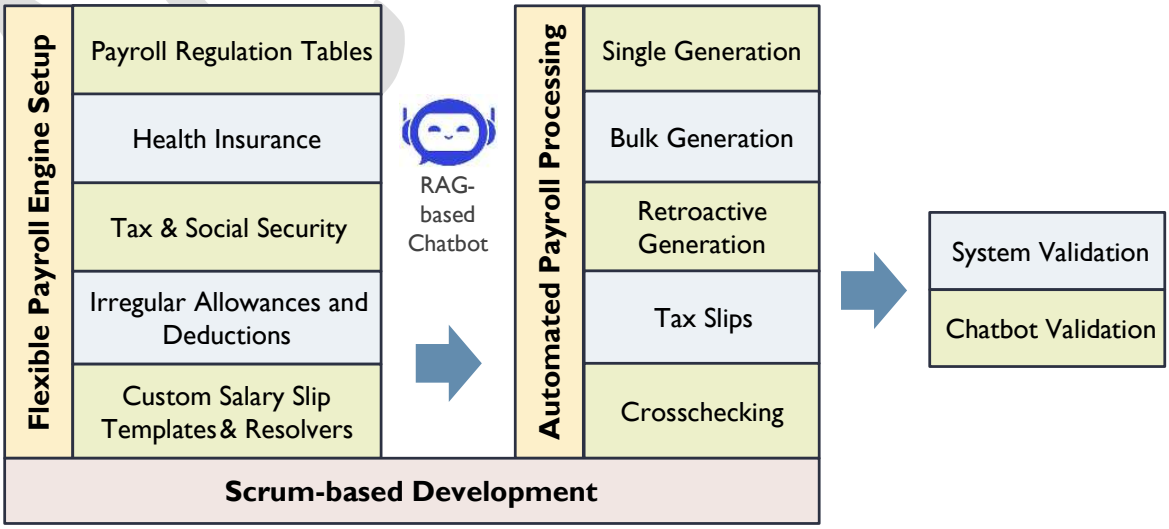


Fig. 1. Methodological framework for payroll digitalization and chatbot support. Illustrates the integration of payroll processing with resolver-based computation, validation, agile development, and chatbot-assisted HR support.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

The remainder of the paper is organized as follows: a review of related literature is presented in Section 2. The Scrum-based agile methodology adopted for the design and development of payroll management modules is described in Section 3. The flexible rule-based payroll setup and customizable salary slip templates are introduced inin Section 4. The three-tier application architecture and the database design are covered in Section 5. The salary computation method is explained in Section 6. Template and resolver design patterns for deterministic, portable, and scalable payroll processing are described in Section 7. The RAG-based chatbot for payroll-specific employee support is outlined in Section 8. A comparison of the introduced framework and features with those from related studies is conducted in Section 9. The payroll system evaluation and chatbot validation methodology, highlighting accuracy, scalability, and efficiency, is presented in Section 10. Finally, conclusions and directions for future work are discussed in Section 11.

2. Literature Review

2.1. Review of Digitalized and Automated Payroll Systems

The study in [3] classified employee financial compensations into fixed pay (e.g., base salary and overtime), variable pay (e.g., incentive bonuses and recognition awards), pay for time (e.g., agency events, holidays, and sick leaves), and income protection programs (e.g., social security and health insurance). The benefits of implementing a web-based payroll application for a minimarket, such as increased accuracy, operational efficiency, and cost savings, were demonstrated in the work in [4]. The importance of providing intelligent payroll management for businesses of all sizes was highlighted in the work in [5]. The study in [6] emphasized the role of robotic process automation in managing repetitive and rule-based tasks within HR and payroll modules, such as recruitment, onboarding, payroll processing, and compliance, which allows staff to focus on more strategic responsibilities. These studies highlight the importance of digitalized payroll, showing how intelligent systems and automation improve accuracy, efficiency, and strategic HR operations, a focus we also adopt in our system.

The study in [7] discussed the analysis, design, and evaluation of a web-based payroll management system for a university. It supports modules such as pay slips, auditing, statistics, and reports. In the study in [8], a three-tier salary management system for colleges was introduced. It offers functions like payroll input, basic settings, payroll audit, and salary payments. The development of employee payroll systems for Indonesian companies, media organizations, and state-owned enterprises was covered in the studies in [9], [10], and [11], respectively. The features and design requirements for a payroll system were further discussed in the work in [12]. The software modules for salary information and management within a human resources system were shown in the study in [13]. Several methodologies for designing payroll information systems, namely waterfall, agile, and scrum, were explored in [14], [15], and [16], respectively. Unlike rigid Waterfall approaches, we adopt Scrum-Agile for its iterative development, modular design, and continuous feedback loops, enabling rapid adaptation to evolving payroll requirements, and provide a detailed description of its implementation.

In the documentation shown in [17], an overview of *RazorpayX Payroll*, a commercial payroll automation software targeting Indian companies of all sizes, was provided. *RazorpayX Payroll* includes features such as salary setup, payroll checklist, salary computation, and one-time payments. Another online payroll solution was introduced in [18], supporting a payroll process based on Canadian regulations with steps to select pay cycle, manage employee earnings, calculate payments, and review payments. Whereas the *ZenHR Payroll* module [19] addresses many of the local tax laws in the Middle East and North Africa (MENA) region, allowing the management of financial transactions, overtimes, part time transactions, and salaries. It is also integrated with an HR system and facilitates the management of social security as well as health insurance deductions. These studies highlight features and regional adaptations of commercial payroll solutions, such as salary computation, tax and social security management, compliance, and HR integration. Our system is also tailored for governmental universities in Jordan.

2.2. AI and LLMs in Payroll and HR Management

Recent work on AI in compensation and payroll systems highlights its role in automating tasks, improving fairness, and enabling adaptive, data-driven decision-making in HR processes [20, 21]. However, systematic reviews indicate that AI applications in HR information systems remain fragmented, with limited focus on employee compensation and benefits, highlighting clear research opportunities [22]. Building on these developments, the study in [23] advances payroll research beyond basic automation toward intelligent, learning-based systems that enhance accuracy, compliance, and efficiency. The study outlines AI-enabled payroll architectures and shows how techniques such as regression, classification, anomaly detection, NLP, and reinforcement learning support predictive analytics, tax computation, error detection, and decision-making in compensation policies.

The study in [24] examined the advantages and challenges of AI chatbots in HR management, highlighting their ability to streamline routine tasks, support decision-making, and improve efficiency, while noting limitations such as limited contextual understanding, emotional intelligence, and privacy concerns. Work in [25] showed that chatbots enhance employee engagement, assistance, productivity, and satisfaction through timely, tailored responses, but also emphasized challenges in system integration, data protection, and user acceptance that require planning, security measures, training, and ongoing monitoring.

The study in [26] reviewed key retrieval approaches, including classical sparse methods such as TF-IDF and BM25 [27], which rely on keyword matching, and dense retrieval, which uses vector embeddings for semantic similarity. It also covered recent strategies

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

to improve retrieval effectiveness. Hybrid retrieval combines sparse and dense models, query expansion and rewriting enrich queries with synonyms, related terms, or contextual information, and multi-stage retrieval with reranking first selects candidate documents with a fast retriever and then reorders them with a stronger re-ranker for higher accuracy. The study highlighted the importance of text structuring to convert unstructured information into organized knowledge and introduced the Retrieval-and-Structuring (RAS) paradigm, integrating information retrieval, knowledge graphs, and large language models into a unified framework.

In the work in [28], LLMs were applied to HR support tasks to address employee inquiries. The development of an HR chatbot involved a human-in-the-loop approach for dataset curation, verification of articles, and tailoring prompts for question-answering. The generated responses were evaluated for truthfulness, relevance, usability, and readability to ensure high-quality outputs.

The study in [29] focused on developing and evaluating a domain-specific AI chatbot for international student admissions using LLM-based RAG pipelines. Multiple pipeline configurations were compared, combining retrieval methods (Dense and Hybrid), chunking strategies (Semantic, Recursive), and both open-source and commercial LLMs. Dual evaluation datasets of LLM-generated and human-tagged QA sets were used to assess answer relevancy, faithfulness, context precision, and recall, along with heuristic NLP metrics. Latency analysis across RAG stages was also conducted to evaluate deployment feasibility in real-world educational settings.

The work in [30] introduced *RankArena*, a platform for evaluating retrieval pipelines, re-rankers, and RAG systems using human and LLM feedback. It supports re-ranker comparison through pairwise battles, manual annotation with quality labels and movement metrics, LLM judgment via GPT-based voting, re-ranker leaderboards, and RAG output evaluation. It also enables dataset collection of aligned human and LLM preferences to train reward models and fine-tune LLM judges for comparing retrieval or RAG systems.

The study in [31] highlighted key LLM vulnerabilities hindering trustworthy deployment, including hallucination, prompt injection, and jailbreaks. Hallucinations, from factual errors to contradictions, can be mitigated with fact-checking pipelines and external grounding, while prompt injection is managed with input/output guardrails and strong system prompts. Jailbreak attempts, including role-playing and multi-turn manipulation, are contained through filtering, continuous monitoring, and authentication.

The work in [32] introduced a framework to help organizations strengthen governance, monitoring, and mitigation of risks from generative AI. It includes an evolving dataset of prompt-injection attacks and a dynamic template for sensitive topics like violence, abuse, and malicious content, supporting proactive robustness assessment and informed risk mitigation.

2.3. Comparison of Proposed and Existing Systems

The proposed system is evaluated in Section 9 against related works. In brief, existing payroll systems lack comprehensive methodologies for planning, design, implementation, and deployment, and offer limited AI or chatbot support. Our system combines rule-based intelligence with a RAG-based chatbot for payroll and tax regulations, using Scrum-Agile for iterative development, modular design, and rapid adaptation. We also discuss system details, including requirements, UI, database, implementation, automation, architecture, security, availability, testing, and generalization. Ablation studies on the chatbot system explored different LLMs, retrieval configurations, embeddings, query transformations, chunk sizes, and prompt-engineering strategies with expert review. These experiments optimized accuracy, reduced hallucinations, and improved transparency. Overall, the RAG-based approach outperforms pure LLM generation, delivering reliable, context-grounded responses.

3. Scrum-Based Agile Methodology for Payroll Information System Development

This study adopts a Scrum-based agile methodology [33] to guide the development and evaluation of the payroll management modules, as illustrated in Fig. 2. Scrum is chosen because traditional models such as Waterfall are rigid and delay changes until the end of the process, which can be costly, particularly in large projects where requirements and details are difficult to determine upfront and often emerge progressively. As an agile framework, Scrum provides the flexibility needed to accommodate evolving requirements throughout development. It emphasizes clearly defined roles, collaborative teamwork, structured planning, active customer involvement, rapid development cycles, continuous review, and ongoing adaptation. By using an incremental and iterative approach, Scrum breaks down large systems into small, manageable increments implemented in time-boxed sprints, typically lasting 2 to 4 weeks to ensure delivery of functional components for early evaluation. At each sprint, completed features are presented to stakeholders for feedback or approval.

The Scrum Team comprises three groups (see Fig. 2): the Scrum Master (SM), the Development Team (TEAM), and the Product Owner (PO). The SM oversees the process by organizing meetings, fostering collaboration, coaching the TEAM, guiding planning, monitoring progress, and facilitating reviews. The TEAM handles project development and support, including programming, software architecture, database design, quality assurance, system administration, and research, as a self-organizing unit to deliver the product increment. The PO is a single individual representing stakeholders, conveying business requirements, and ensuring the TEAM delivers value. In this case, the development team leader also serves as SM, while the Assistant to the President for Administrative Affairs serves as PO, representing HR and Finance and consolidating stakeholder input.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

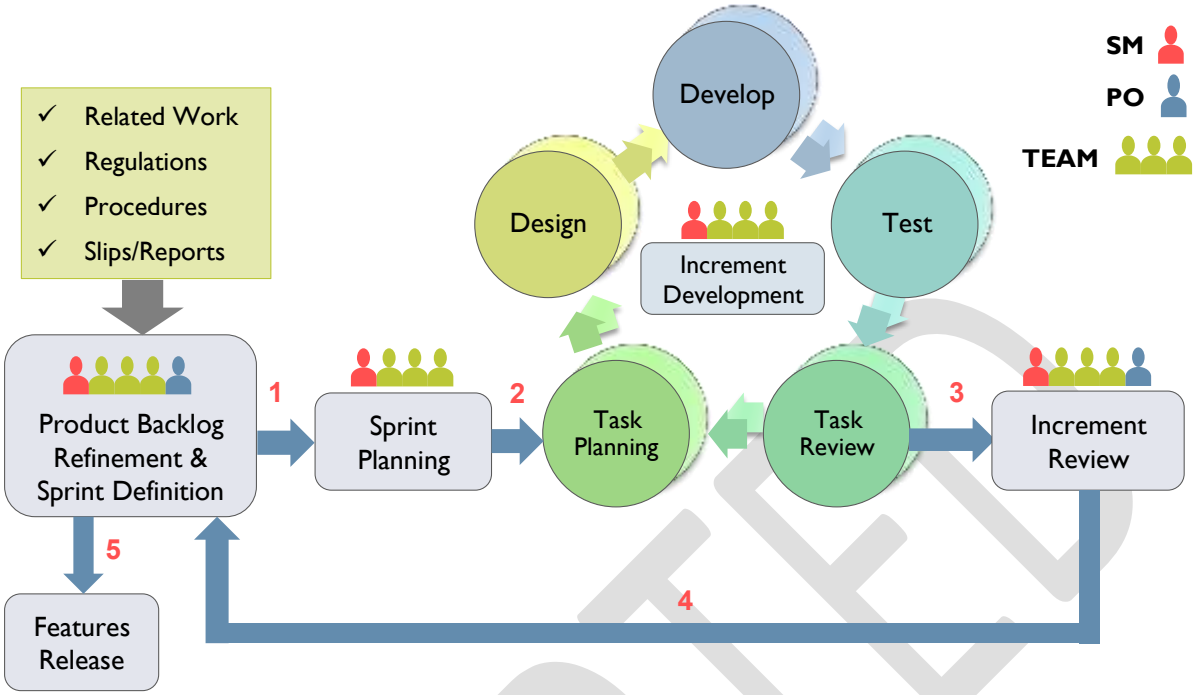


Fig. 2. Scrum-based agile methodology used to develop software modules for employee payroll management.

Based on Fig. 2, the primary stages of the innovative scrum-based methodology for digitalizing the payroll management process are as follows:

A. *Product Backlog Refinement and Sprint Planning*: The SM facilitates regular meetings with the TEAM and PO to refine the *Product Backlog* (an ordered and evolving list of all work needed to build the product), define sprint increments, and support planning adjustments, while the PO maintains backlog prioritization. Given the complexity of this project, requirements are specified gradually as manageable and interrelated increments. An increment represents one or more usable software features developed within a sprint. The objectives for each increment are based on PO stories, PO priorities, research outcomes, regulations, standard operating procedures, forms, and dependencies on other increments. These meetings reconvene after each sprint to review completed increments and plan the next iteration.

Accordingly, the primary user roles and software features identified to manage employee payroll are summarized in the use-case diagram shown in Fig. 3. The *HR Admin* role is needed to manage (i.e., add, view, edit, delete) HR-specific data, including general HR setup (e.g., university structure, university entities, employee attributes, commission types, and job titles) and payroll-related HR setup (e.g., tax laws, health insurance contracts, and social security contributions). This role also handles employee details like ID, personal and contact information, employment status, type, grade, promotions, commissions, certificates, family members, work hours, health insurance subscription, vacations, and leaves. On the other hand, the *Payroll Admin* user role focuses on payroll centric features like payroll setup On the other hand, the *Payroll Admin* user role focuses on payroll features such as payroll setup, salary computation, income tax slip generation, and payroll reports.

Although this work mainly focuses on the specification, design, and development of employee payroll management modules rather than the HR modules, it is important to note that the payroll modules have direct access to the HR database tables needed to implement the desired functionalities. Moreover, employees with a *Payroll Admin* role have view access to all HR reports to support informed decision-making.

B. *Sprint Planning*: The SM facilitates meetings with the TEAM to plan the current sprint based on the known features for the increment. Each feature is broken down into tasks, and the self-organizing TEAM collaboratively decides how to execute them.

C. *Increment Development*: For each task, the TEAM carries out full planning, design, implementation, and testing. Tasks may be worked on in parallel by different TEAM members. Once all tasks for the features meet the *Definition of Done*, the increment is considered complete. The TEAM holds Daily Scrums to track progress, resolve challenges, and adjust plans as needed, with the SM facilitating the process.

D. *Increment Review*: The TEAM demonstrates the completed features to the PO for evaluation. Any feedback or requested changes from the PO are added to the *Product Backlog* and addressed in subsequent *Product Backlog Refinement and Sprint Planning* meetings. The TEAM and PO then review the *Product Backlog* to plan the next sprint, with the SM facilitating the process.

E. *Features Release*: Completed software modules are tested and released to stakeholders or made available for use and evaluation once they meet the *Definition of Done*. Feedback is collected to guide future increments.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

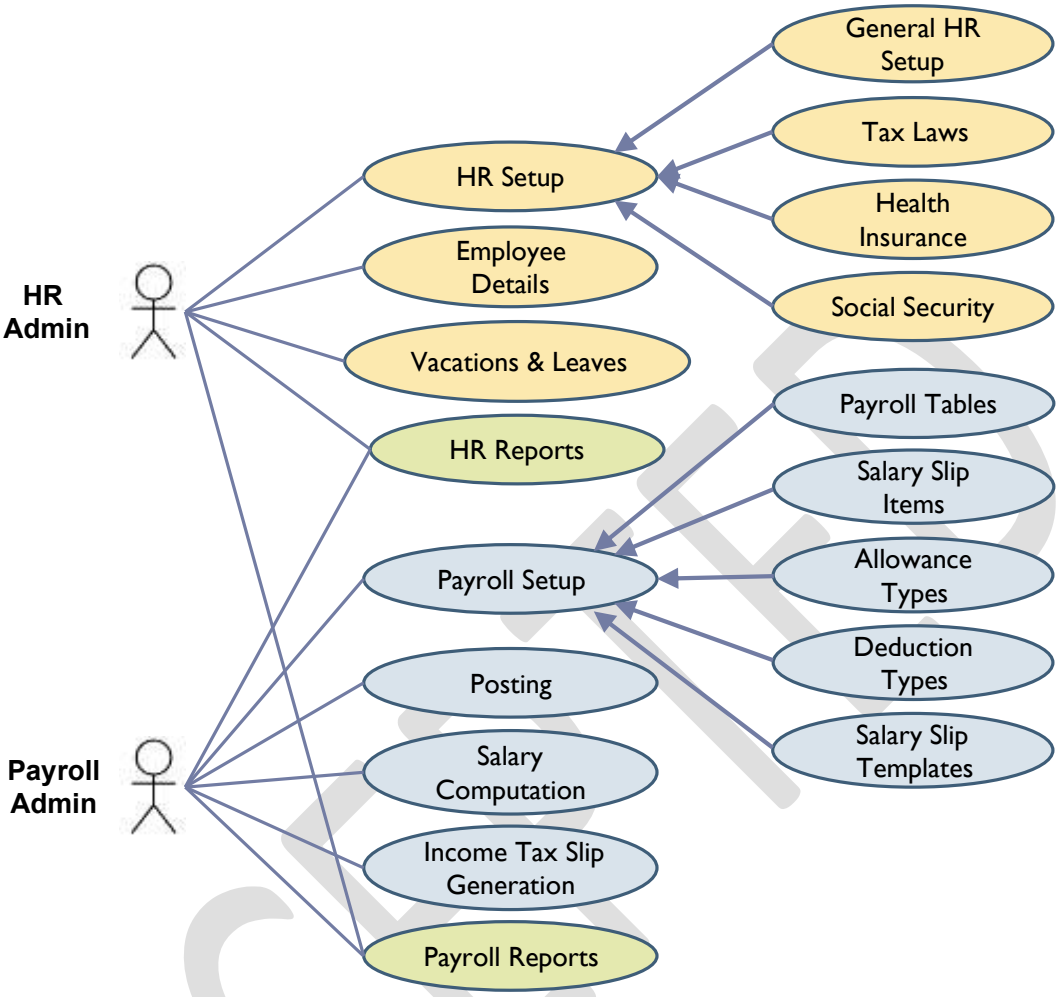


Fig. 3. Use-case diagram showing the main software features needed to support payroll management in the HRPIS.

4. Rule-Based Payroll Setup

To understand the setup requirements needed to compute the salaries of the employees at GJU, let us begin by exploring the structure of the salary slip template shown in Fig. 4. In this context, a salary slip template specifies how to compute the salary of each employee linked to it. The template consists of a salary details list and a salary deductions list. The salary details list contains the allowances that could be paid to an employee, whereas the salary deductions list covers the possible deductions from the employee payment. Hence, the net total salary amount can be computed by subtracting the total deductions from the total allowances. The items required to build salary slip templates can be managed using the salary slip items management screen shown in Fig. 5. Accordingly, each item has several attributes, such as name, type (i.e. either allowance or deduction), taxable (i.e. a flag to indicate whether to consider the item amount in the tax amount computation), include in social security (i.e. a flag to indicate whether to consider the item amount in the social security contribution computation), and linkable (i.e. a flag to link an item to irregular allowance or deduction types for the purpose of retrieving values). Specifically, supporting salary slip item attributes is essential to enable the automatic calculation of each item's amount and the overall net salary. Additionally, the setup data required to compute each salary slip item can vary depending on the item's name and attributes. For example, calculating the basic salary, income tax, and health insurance amounts depends respectively on payroll regulations, tax laws, and health insurance contracts.

In that respect, the following sections discuss the various setup methods used to define the data required for computing the amount of each item in a salary slip. In addition, the features that support customized salary slip templates, which are used as references for calculating salaries of employees with different types and classifications, are also presented.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Academic Employee Template	
Salary Details	Salary Deductions
Basic Salary	Income Tax
Administration Allowance	Income Tax Overtime
University Allowance	Social Security
Transportation Allowance	Saving Fund 5% from Basic Salary
Personal Allowance	Health Insurance
Family Allowance	Cancer Care Insurance
Rare Majors Allowance	Telephones Bills
GJU Establishment Allowance	Other Deductions
Speciality Allowance 135% from Basic Salary	Social Committee Fees
Misc Differences	
Over Time	
Extra Allowances	

Fig. 4. Salary slip template structure. Defines employee salary computation using allowances and deductions.

Salary Slip Items Information

(1 of 5) 1 2 3 4 5 10

Name (EN)	Name (AR)	Type	Taxable	Include in Social Security	Overtime	Linkable	Active	Edit
Basic Salary	راتب اساسي	Allowance	Yes	Yes	No	No	Yes	
Administration Allowance	علاوة ادارة	Allowance	Yes	Yes	No	No	Yes	
University Allowance	علاوة جامعة	Allowance	Yes	Yes	No	No	Yes	
Transportation Allowance	علاوة تنقل	Allowance	Yes	Yes	No	No	Yes	
Personal Allowance	علاوة شخصية	Allowance	Yes	Yes	No	No	Yes	
Family Allowance	علاوة عائلة	Allowance	Yes	Yes	No	No	Yes	
Rare Majors Allowance	علاوة تخصصات نادرة	Allowance	Yes	Yes	No	No	Yes	
GJU Establishment Allowance	علاوة تاسيس الجامعة الالمانية	Allowance	Yes	Yes	No	No	Yes	
Speciality Allowance 135% from Basic Salary	علاوة اختصاص 135% من الراتب الاساسي	Allowance	Yes	Yes	No	No	Yes	
Extra Allowances	مكافات اضافية	Allowance	Yes	No	No	Yes	Yes	
Social Committee Fees	لجنة اجتماعية	Deduction	No	No	No	No	Yes	
Other Deductions	اقتطاعات اخرى	Deduction	No	No	No	Yes	Yes	

+ Add Print Export

Fig. 5. Salary slip items management screen. Manages salary items and their attributes (e.g., type, taxable status, and social security inclusion) to support automated salary calculation.

4.1. Defining Payroll Regulation Tables

The amount of most items on the salary slip can be retrieved from related lookup tables, which are populated based on the payroll regulations of GJU. For reference, Table 1 illustrates the employee categories at GJU along with their primary attributes, which are essential for determining various salary slip items. As shown in Table 2, the basic salary of an academic employee depends on the employee's academic grade, grade category, number of years within the grade category, and the salary slip's effective date. For example, the basic salary in March 2025 for a professor in grade category A in their first year within that category is 600 Jordanian Dinars (JD). Furthermore, the administration allowance is determined based on the employee's commission (e.g., dean, chair, or director) and salary slip date, as indicated in Table 3. Moreover, the university allowance depends on the employee's grade and salary slip date, as outlined in Table 4. In addition, the transportation allowance solely depends on the effective salary slip date, as shown in Table 5. Also, the specialty allowance is calculated as 135% of the basic salary, which in the above example amounts to 810 JD, as stated in Table 6.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Particularly, the HRPIS supports the management of lookup tables for all salary slip items via the *Payroll Setup* menu, as shown in Fig. 6. For instance, clicking the *Basic Salaries* button in this menu allows users to manage the basic salaries tables for both academic and administrative staff. Specifically, the screen used to configure the lookup table for the university allowance of academic employees is illustrated in Fig. 7.

It is important to note that the values used in the examples and lookup tables throughout this paper are not real, as they have been modified for privacy reasons. Additionally, most of the lookup tables have been shortened to address space limitations, though this has been done without compromising the data necessary to illustrate the core concepts. Besides, all amounts in the lookup tables are associated with specific periods to allow for value changes over time. For example, as shown in Table 5, this setup enabled an increase of 10 JD in the transportation allowance starting from July 2021.

Table 1 Lookup table for employee categories. Defines categories by type, grade, grade category, and years of service, used to determine applicable salary scales.

Employee Categories			
Employee Type	Grade	Grade Category	Years
Academic	Professor	A	1-31
	Associate Professor	A	1-5
		B	1-5
	Assistant Professor	A	1-5
		B	1-5
Administrative	First Grade	A	1-16
		B	1-5
	Second Grade	A	1-5
		B	1-5
	Third Grade	A	1-5
		B	1-5
	Fourth Grade	A	1-5
		B	1-5

Table 2 Basic salary lookup. Defines basic salary based on academic grade, grade category, years within category, and effective date.

Basic Salary					
Academic Grade	Grade Categ.	Year	Amount (JD)	From Date	To Date
Professor	A	2	630	June 1, 2005	
		1	600	June 1, 2005	
Associate Professor	A	2	520	June 1, 2005	
		1	500	June 1, 2005	
	B	2	420	June 1, 2005	
		1	400	June 1, 2005	
Assistant Professor	A	2	360	June 1, 2005	
		1	350	June 1, 2005	
	B	2	310	June 1, 2005	
		1	300	June 1, 2005	

Table 3 Administration allowance lookup. Specifies administration allowance based on employee commission and salary slip date.

Administration Allowance			
Commission	Amount (JD)	From Date	To Date
Dean	300	June 1, 2005	
Chair	100	June 1, 2005	
Director	100	June 1, 2005	

Table 4 University allowance lookup. Defines university allowance based on employee grade and salary slip date.

University Allowance			
Academic Grade	Amount (JD)	From Date	To Date
Professor	800	July 1, 2021	
Associate Professor	600	July 1, 2021	
Assistant Professor	400	July 1, 2021	
Professor	700	June 1, 2005	June 30, 2021
Associate Professor	500	June 1, 2005	June 30, 2021
Assistant Professor	300	June 1, 2005	June 30, 2021

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 5 Transportation allowance lookup. Specifies transportation allowance based on the salary slip effective date.

Transportation Allowance		
Amount (JD)	From Date	To Date
100	July 1, 2021	
90	June 1, 2005	June 30, 2021

Table 6 Specialty allowance calculation. Defines specialty allowance as a percentage of the basic salary.

Specialty Allowance		
Percentage from Basic Salary	From Date	To Date
135	June 1, 2005	

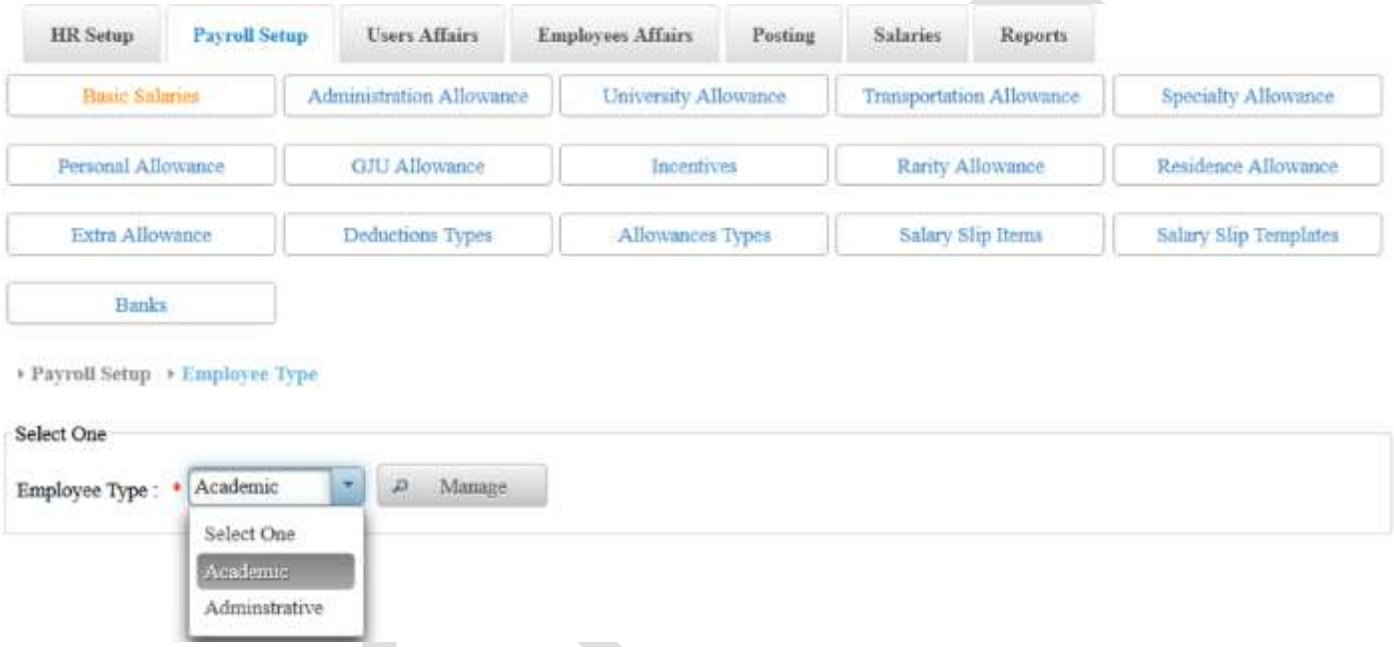


Fig. 6. Payroll setup menu. Provides access to general payroll configuration and salary-related data, with the basic salaries setup entry point highlighted.

Academic University Allowances			
Academic Grade	Amount	From Date	To Date
Professor	88	2021-10-01	
Associate Professor	65	2021-10-01	
Assistant Professor	49	2021-10-01	
Industrial Professor	41	2021-10-01	
Instructor	31	2021-10-01	
Assistant Instructor	26	2021-10-01	
Professor	86	2021-09-01	2021-09-30
Associate Professor	64	2021-09-01	2021-09-30
Assistant Professor	48	2021-09-01	2021-09-30
Industrial Professor	39	2021-09-01	2021-09-30
Instructor	29	2021-09-01	2021-09-30
Assistant Instructor	24	2021-09-01	2021-09-30

Fig. 7. University allowance lookup table management screen. Displays academic university allowance values and their effective dates, with amounts expressed in Jordanian Dinars (JD).

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

4.2. *Defining Health Insurance Contracts*

To facilitate the automatic computation of the health insurance amount in a salary slip, the HRPIS includes features for managing health insurance company information and contract details (see Fig. 8 and Fig. 9), as well as recording the employee and family member subscriptions to health insurance (see Fig. 9 and Fig. 10). As shown in Fig. 8, a health insurance contract is linked to a provider and includes a contract number, maximum coverage amount, and an effective period. It also defines specific age ranges (e.g., 0-17, 18-65, and 66-90), with each range potentially applicable to the employee, their family members, or parents. Moreover, as illustrated in Fig. 9, a contract may offer multiple coverage levels, such as *VIP Class*, *First Class*, and *Second Class*. Besides, each contract level has issuance fees and basic fees that vary based on the age range of the subscribers. Further, each coverage level within a contract includes a table of contribution percentages, where the shares paid by the employee and the university vary based on the subscriber's relationship to the employee. Lastly, the employee and their family members can be linked to the same health insurance contract and contract level, as shown in Fig. 10.

4.3. *Defining Tax Laws and Social Security Contributions*

The HRPIS supports the definition of a tax law with its details, as shown in Fig. 11, to allow the automatic computation of income tax in salary slips. In that regard, the main components of the Jordanian tax law are the exemption rules and the tax brackets tables. Specifically, the exemptions table lists the annual non-taxable income thresholds for individuals filing taxes either separately or jointly. The brackets table defines income ranges subject to specific tax rates based on cumulative taxation, after accounting for tax exemptions. Accordingly, this information, along with the *Taxable* flag (see Fig. 5) linked to each salary slip allowance, is used to calculate the total tax liability in a salary slip. For example, if a couple filing taxes jointly have a total yearly income of 28,000 JD, they are eligible for an 18,000 JD tax exemption, leaving a taxable income of 10,000 JD. They will be taxed 250 JD (5% tax rate) on the first 5,000 JD and 500 JD (10% tax rate) on the remaining 5,000 JD, resulting in a total yearly income tax of 750 JD.

Furthermore, as illustrated in Fig. 12, the HRPIS allows specifying the social security contribution percentages for both employees and the university, differentiated by risky and non-risky jobs, to facilitate the calculation of social security contributions in the salary slip. Hence, this data, along with the *Include in Social Security* flag (see Fig. 5) linked to each salary-slip allowance, is used to calculate the total social security amount for each month.

Fig. 8. Health insurance contract management screen. Provides an entry point for managing contracts and accessing related age ranges and contract levels.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Fig. 9. Health insurance contract levels management screens. Manage contract levels along with their associated contribution and fee tables.

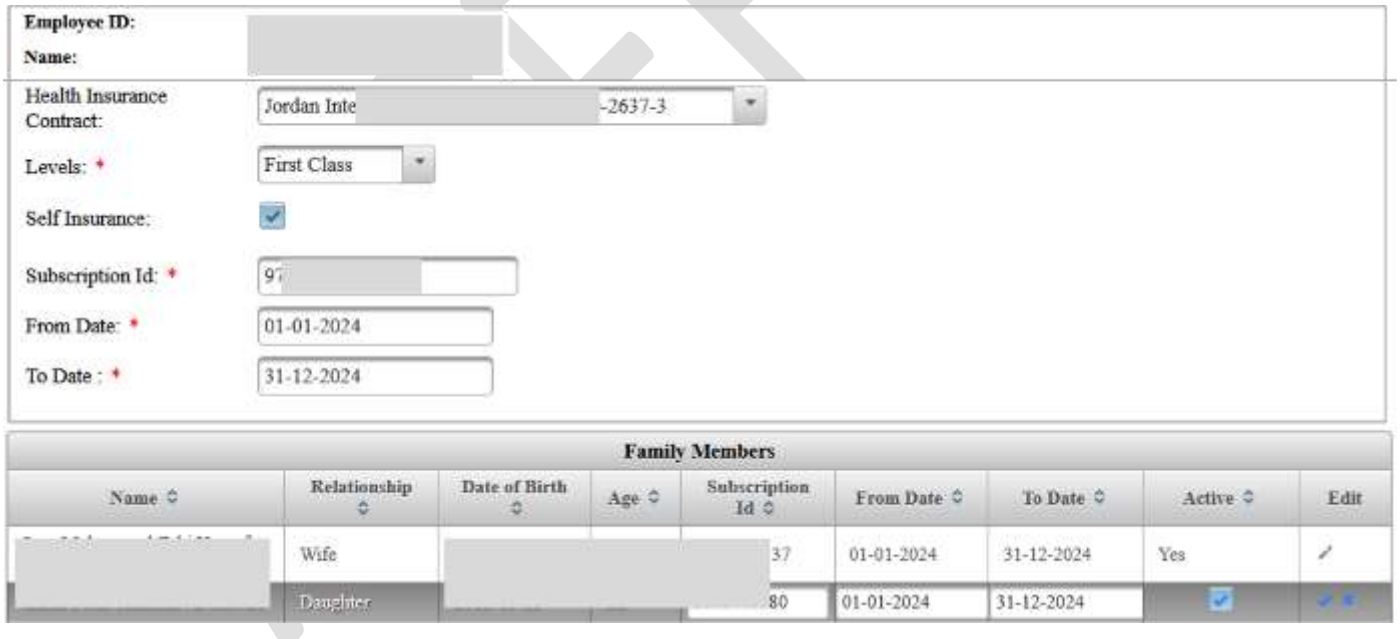


Fig. 10. Health insurance subscription management screen. Links employees and their family members to specific health insurance contracts and coverage levels.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

HR Setup › Tax Laws › Tax Law Details

Tax Law Information

Name:	Income Tax Law (38) for the year 2018	**Active:**	No
Non Resident Overtime Rate:	25	**Resident Overtime Rate:**	25
Start Year:	2020	**End Year:**	2020

Exemptions

ID	Amount (JD)	Edit
1	9000	
2	18000	
3	23000	

Brackets

ID	From Amount	To Amount	Rate	Edit
1	1	5000	5	
2	5001	10000	10	
3	10001	15000	15	
4	15001	20000	20	
5	20001	1000000	25	
6	1000001	20000000	30	

Fig. 11. Tax law configuration screen. Defines tax exemptions and income tax brackets, with amounts in Jordanian Dinars (JD), for automatic income tax calculation.

HR Setup › Social Security

Social Security Contributions

Job Risk	Employee Percentage	GJU Percentage
Risky	6.5	14.0
Non Risky	6.5	13.0

Fig. 12. Social security configuration screen. Specifies employee and university contribution percentages based on job risk level for calculating social security contributions.

4.4. Defining Irregular Allowance and Deduction Types

Some salary slip items, like the *Extra Allowances* and *Other Deductions* shown in Fig. 4, are associated with irregular allowance or deduction types. Typically, an allowance or deduction is considered irregular if it does not apply to all employees, is not disbursed monthly, and is unsuitable for automatic computation. For example, a reward for participating in a university service committee is disbursed only to the relevant employees and only during the months in which the service was delivered. Similarly, telephone bill amounts are provided monthly by the service provider to the finance department and then deducted from the salaries of the subscribing employees. Therefore, it is necessary to provide an easy-to-use mechanism in the HRPIS to incorporate the values of such allowances or deductions into the relevant employees' pay slips when they are due.

In that regard, the HRPIS supports the management of irregular allowance and deduction types via the *Payroll Setup* tab, as shown in Fig. 6, while the screen for managing irregular allowance types is illustrated in Fig. 13. Specifically, the irregular *Programming Reward* allowance type, shown in Fig. 13, was defined and linked to the *Extra Allowances* salary slip item using the screen shown in Fig. 14. Importantly, each irregular allowance or deduction type must be associated with a linkable salary slip item (e.g., the *Extra Allowances* item shown in Fig. 14) to ensure its value is accurately retrieved and reflected in the relevant salary slip. It is also worth noting that multiple irregular allowance or deduction types can be associated with one linkable salary slip item. In such cases, the amount of the item will equal the sum of the values of the irregular types linked to it. This one-to-many grouping leads to a more concise salary slip, as displaying a single linkable item eliminates the need to list each associated allowance or deduction individually. However, the detailed breakdown of the grouped allowances or deductions remains accessible in the profile of the relevant employee for tracking purposes.

Once an irregular allowance or deduction type is introduced, the HRPIS allows the addition of corresponding definition entries that specify the applicable values and time periods for the relevant employees, as shown in Fig. 15. Accordingly, the HRPIS can generate (post) monthly values from these definitions (see Fig. 16) and link them to the relevant employees. This enables the system

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

to incorporate the posted allowance/deduction amounts when computing salaries for the corresponding month. Suitably, the system also supports reusing a definition across multiple months when its specified period spans several salary cycles. For example, the *Programming Reward* definition (see Fig. 15) has a start-date and an open-ended end date, resulting in a 500 JD allowance being recorded for the relevant employee each month, beginning in September 2021, upon posting the allowances. Nevertheless, certain allowances and deductions, such as the telephone bills that get issued by an external entity based on employee service usage, do not have specific definitions. To accommodate such cases, the system allows the monthly import of Excel files containing the corresponding amounts for relevant employees, as shown in Fig. 17.

In that regard, based on the example shown in Fig. 15, assume that four irregular allowance definitions with different values and periods have been linked to an employee whose ID is 326. Suitably, the screens shown in Fig. 16 can be used to automatically post the allowance values for February 2014 to the relevant employees, including the employee with ID 326. It is worth noting that the system allows for reviewing and verifying the resulting values prior to saving them, as illustrated in Fig. 16. Accordingly, a *Receiving Committee Reward* of 10 JD and a *Programming Reward* of 500 JD will be scheduled for disbursement to that employee in the February salary slip. Since the two allowance types are linked to the *Extra Allowances* salary slip item (see Fig. 13), a total amount of 510 JD will be disbursed to the employee under the *Extra Allowances* item in the February salary slip.

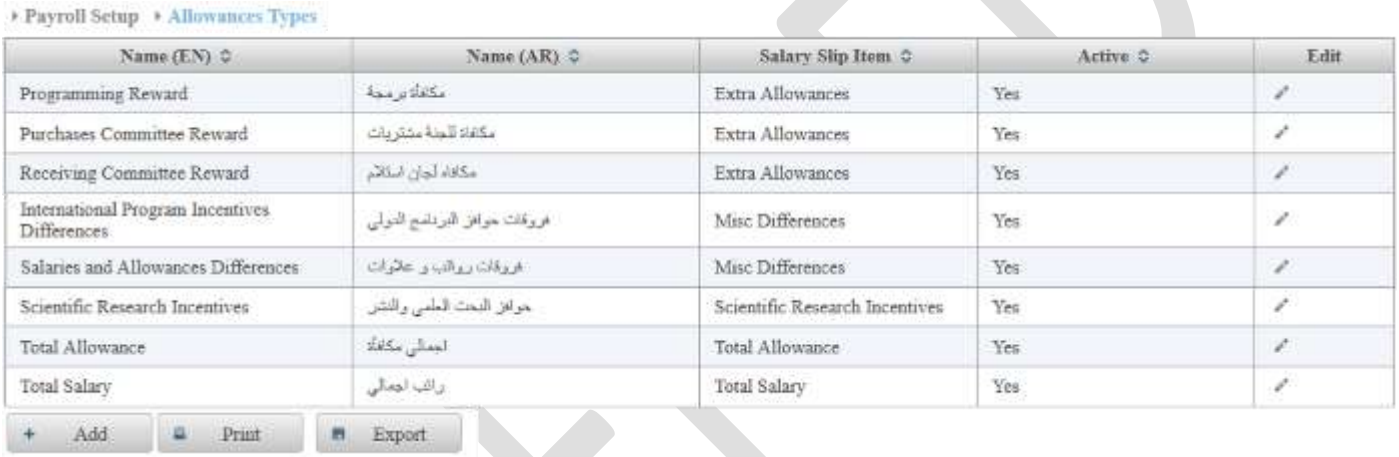


Fig. 13. Irregular allowance types management screen. Manages irregular allowance types and their configuration within the payroll system.

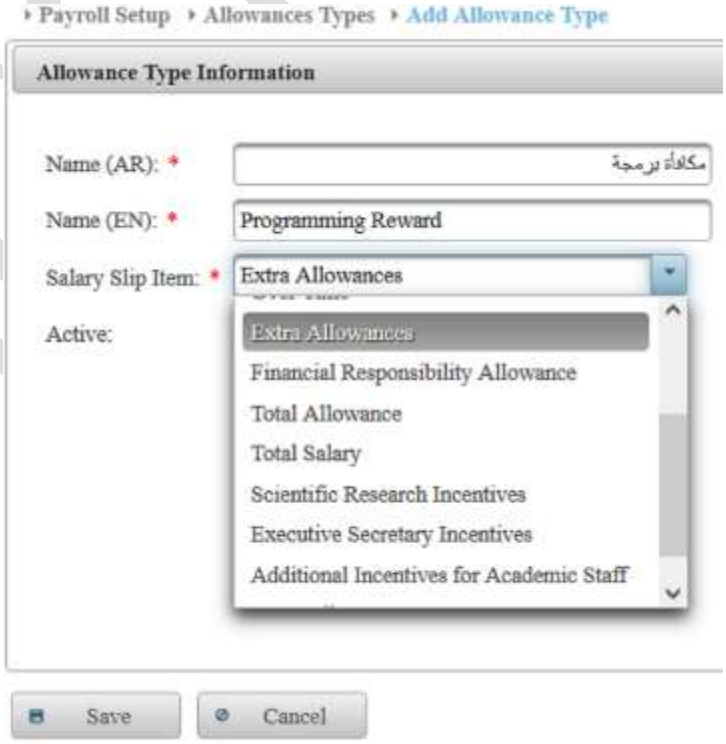


Fig. 14. Add irregular allowance type screen. Allows adding a new irregular allowance type and linking it to its corresponding salary slip item.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

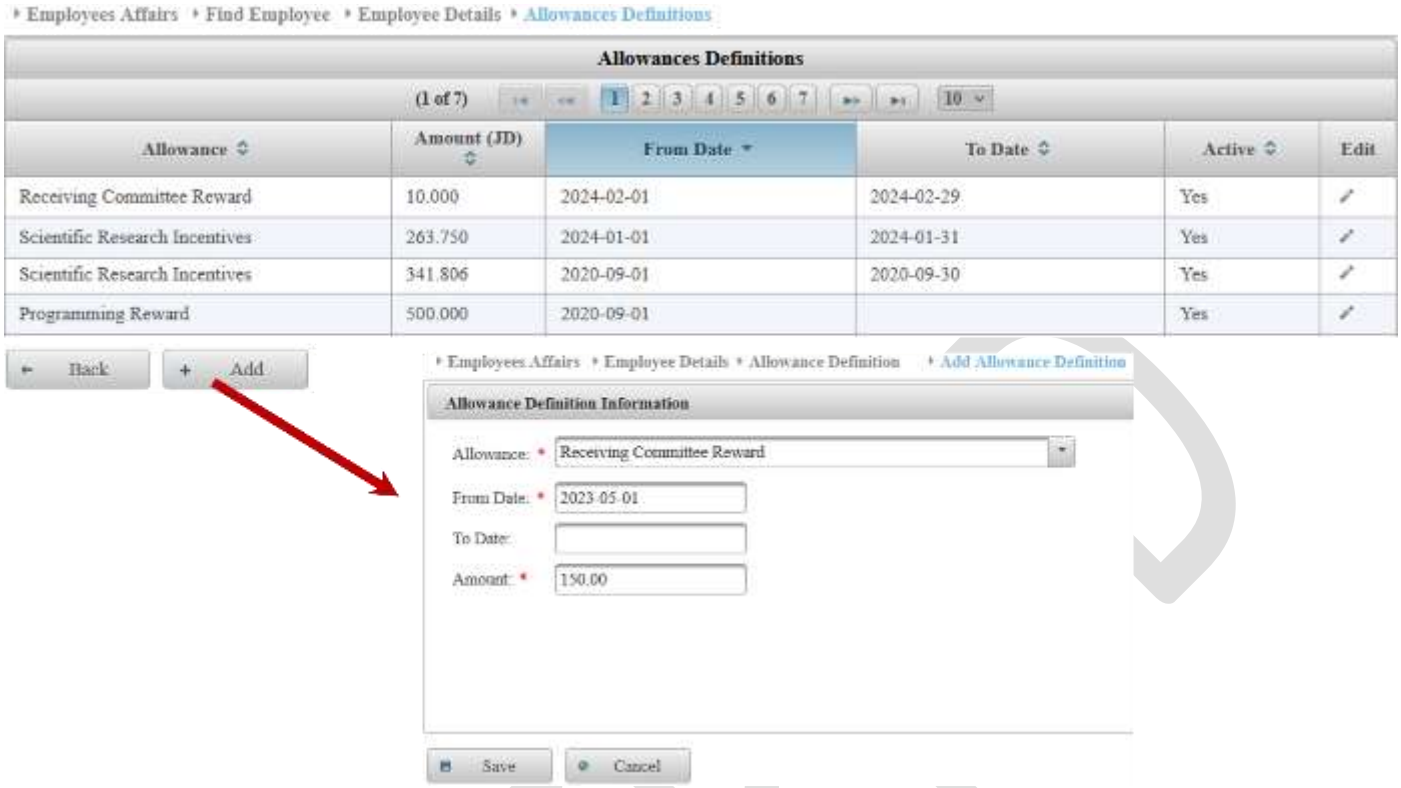


Fig. 15. Irregular allowance definition screen. Defines allowance values, expressed in Jordanian Dinars (JD), and their applicable time periods for specific employee.

Fig. 16. Allowance posting screens. Generate and post monthly allowance and deduction values based on defined entries.

Fig. 17. Telephone bill charges import screen. Supports importing monthly telephone bill charges for relevant employees from an Excel spreadsheet.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

4.5. Defining Custom Salary Slip Templates

The existence of different types of employees at a university requires the issuance of salary slips with different details based on the employee type. Nevertheless, defining the details of a salary slip on an individual basis becomes time consuming and impractical when there are many employees. Therefore, this work proposes utilizing a limited number of salary slip templates to associate each of them with a group of employees having the same salary details, and then repeatedly use those templates as a blueprint to generate the salary slips of the related employees.

The currently supported salary slip templates in the HRPIS are shown in Table 7, in which there are four unique templates that can be automatically associated to employees based on their type (i.e. academic or administrative) and classification (i.e. detailed contract, total contract, or reward) when their salaries are computed. Specifically, the screen used to manage salary slip templates in the HRPIS is shown in Fig. 18, while Fig. 3 presents the template for academic employees. The main steps to define a salary slip template, summarized in Fig. 19, are as follows:

- Add the needed salary slip items with their attributes through the screen shown in Fig. 5.
- Define the needed irregular allowance types and link them to the related salary slip items using the screen shown in Fig. 13.
- Specify the required irregular deduction types and link them to the relevant salary slip items.
- Add a new salary slip template via the screen in Fig. 18.
- Add items to the template, linking each to a specific salary-slip item and assigning an order to control its placement in the slip.

Table 7 Salary slip templates. Lists available salary slip templates and their association with employee type and classification.

Salary Slip Templates		
Employee Type	**Employee Classification**	**Salary Slip Template**
Academic	Detailed Contract	Academic Employee Template
	Total Contract	Total Salary Template
	Reward	Total Reward Template
Administrative	Detailed Contract	Administrative Employee Template
	Total Contract	Total Salary Template
	Reward	Total Reward Template

Fig. 18. Salary slip templates management screen. Allows managing and configuring salary slip templates within the HRPIS.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

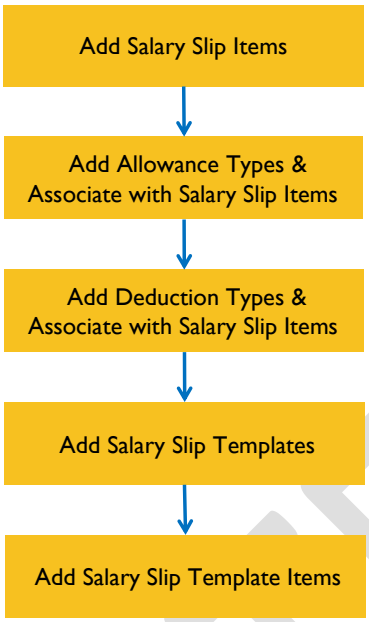


Fig. 19. Salary slip template definition workflow. Summarizes the main steps for defining a salary slip template.

5. Three-Tier Web Application Architecture

As illustrated in Fig. 20, the HRPIS is a Java EE [34] web application that is developed based on a three-tier architecture consisting of the following tiers: client, web, and data. In the client tier, users can utilize web browsers on computers or phones to access and use the application. In the web tier, the application modules are hosted and executed in a web application server, and implemented based on the design patterns and methods discussed in the work in [35, 36]. The application's software components use the Java Database Connection (JDBC) APIs [37] to access the needed tables in the Oracle database that is managed in the data tier. Additionally, GJU employees can access their HR and payroll information through the university portal, MyGJU [38, 39], which is hosted in the web tier and retrieves data from the HR and Payroll database tables.

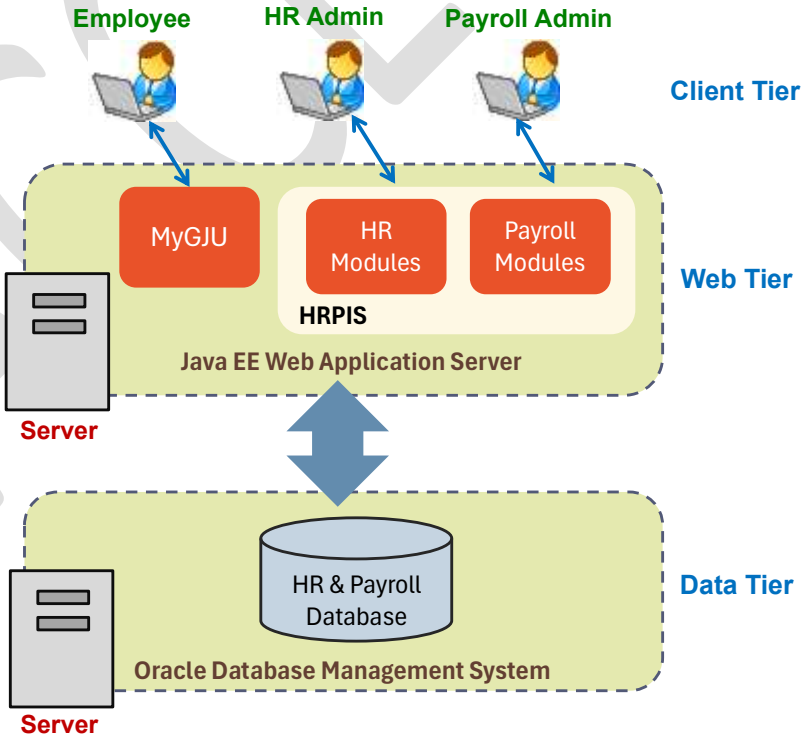


Fig. 20. HRPIS system architecture. Illustrates the three-tier architecture of the HRPIS, including client, web, and data tiers.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

5.1. *Database Design*

The database tables related to specifying tax laws, health insurance contracts, and regulated allowances are illustrated in the ER diagram in Fig. 21. The definitions of the tax laws are stored in the TAX_LAW table, whereas the bracket and exemption data linked to each tax law are saved in the TAX_LAW_BRACKET and TAX_LAW_EXEMPTION tables, respectively. Moreover, the health insurance companies information are kept in the HEALTH_INS_COMPANY table, where each company can have several contracts defined in the HEALTH_INS_CONTRACT table. Contract specifics, such as levels and age ranges, are saved in the LEVEL and AGE_RANGE tables, respectively. While the data for the fees and contributions of each contract level are inserted in the LEVEL_FEE and LEVEL_CONTRIBUTION tables, respectively. Data for each regulated allowance is saved in a separate table. For example, the basic salary and the university allowance for the academic employees are saved in the ACA_BASIC_SALARY and ACA_UNIV_ALLOWANCE tables, respectively.

Additionally, the database tables used to define irregular allowances, salary slip templates, and employee salary slips are illustrated in the ER diagram in Fig. 22. The salary slip items are saved in the SALARY_SLIP_ITEM table. Meanwhile, irregular allowance and deduction types reside in the ALLOWANCE TYPE and DEDUCTION_TYPE tables, respectively, noting that each of these types must be linked to a salary slip item. Moreover, employee-specific allowance and deduction definitions are saved in the EMP_ALLOWANCE_DEF and EMP_DEDUCTION_DEF tables, respectively. The posted monthly values based on the allowance and deduction definitions for the related employees are inserted in the EMP_MONTHLY_ALLOW and EMP_ MONTHLY_DED tables, respectively. Salary slip templates are defined in the SALARY_SLIP_TEMPLATE table, while the items in each template are inserted in the SALARY_SLIP_TEMPLATE_ITEM table. Finaly, header data for the employee salary slips are saved in the EMP_SALARY_SLIP_MAST table, and the items in each salary slip are stored in the EMP_SALARY_SLIP_DET table.

To clarify some concepts, we will present a few examples to illustrate how selected database tables can be populated with information. Based on the data in the tables shown in Fig. 23, two irregular allowance types, namely *Programming Reward* and *Committee Reward*, were defined and linked to the *Extra Allowances* row in the SALARY_SLIP_ITEM table (via its SSI_PK foreign key of value 1). Similarly, three irregular deduction types, namely *Nursery Subscription, Petty Cash,* and *Telephone Bills*, were defined and linked to SALARY_SLIP_ITEM rows via SSI_PK foreign keys of values 2, 2, and 3, respectively. Furthermore, as shown in Fig. 24, two allowance definitions for employee EMP_ID 100 and one for EMP_ID 102 were inserted into the EMP_ALLOWANCE_DEF table. Accordingly, when the allowances are posted on March 2025, EMP_ID 100 will have a 500 JD *Programming Reward* allowance and a 150 JD *Committee Reward* allowance recorded in the EMP_MONTHLY_ALLOW table, as both related definitions are active in that month. Similarly, EMP_ID 102 will have a 150 JD *Committee Reward* allowance inserted in the EMP_MONTHLY_ALLOW table. Thus, when the March 2025 salaries are computed, EMP_ID 100 and EMP_ID 102 would receive 650 JD and 150 JD, respectively, against the *Extra Allowances* item in their salary slips.

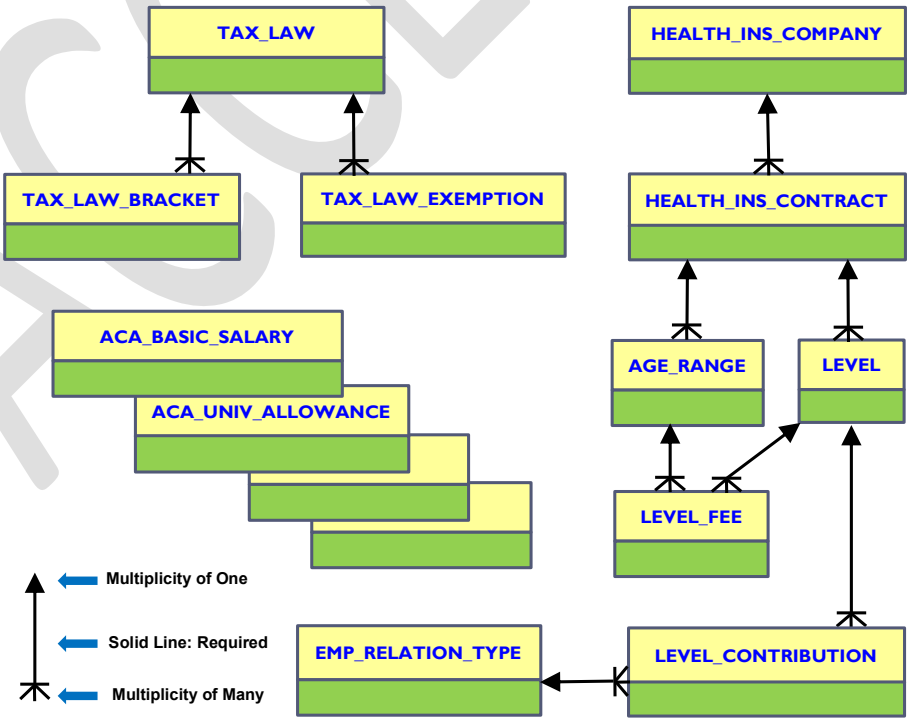


Fig. 21. ER diagram for tax, health insurance, and regulated allowances. Illustrates database tables and relationships for tax laws, health insurance contracts, and regulated allowances.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

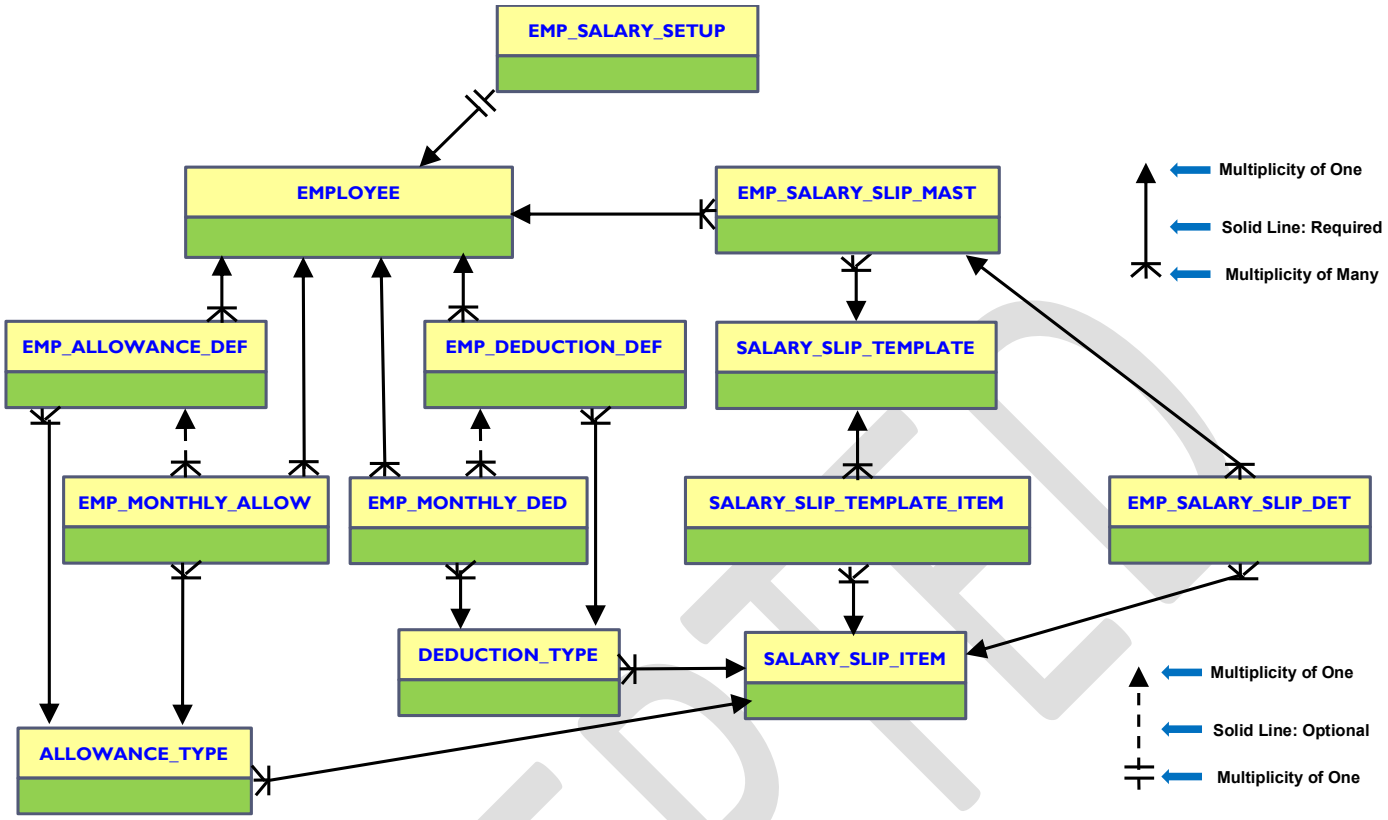


Fig. 22. ER diagram for irregular allowances, deductions, and salary slips. Illustrates database tables and relationships for irregular allowances, deductions, salary slip templates, and employee salary slips.

ALLOWANCE_TYPE

AT_PK	NAME	SSI_PK
1	Programming Reward	1
2	Committee Reward	1

DEDUCTION_TYPE

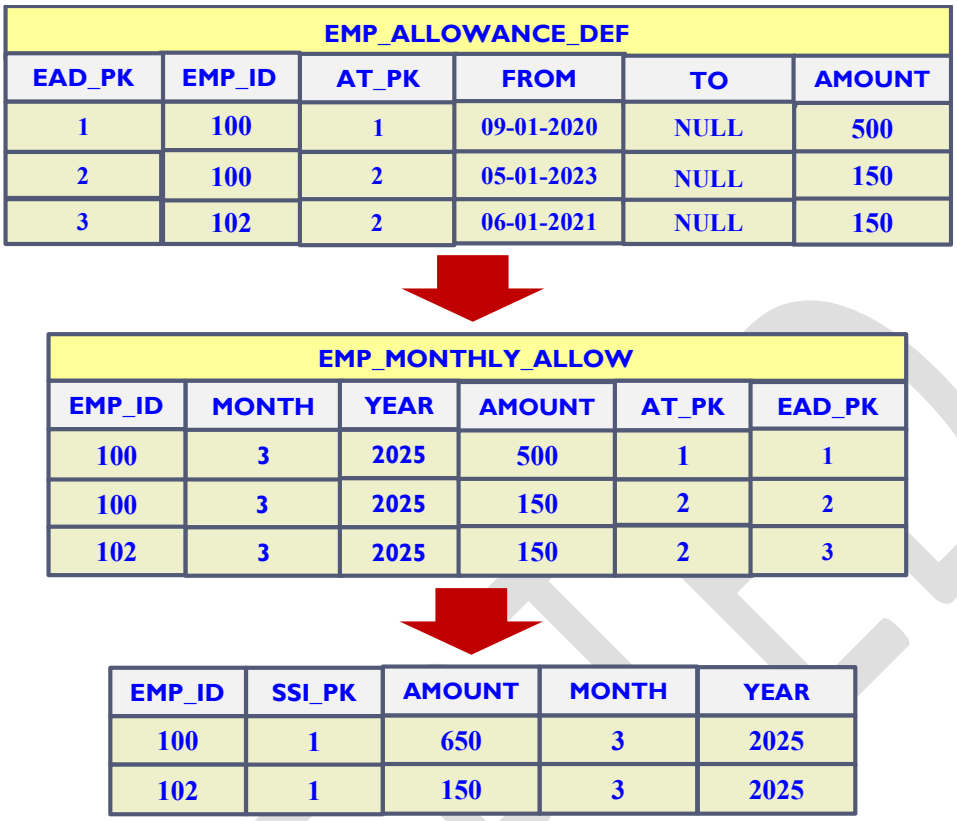
DT_PK	NAME	SSI_PK
1	Nursery Subscription	2
2	Petty Cash	2
3	Telephones Bills	3

SALARY_SLIP_ITEM

SSI_PK	NAME
1	Extra Allowances
2	Other Deductions
3	Telephones Bills

Fig. 23. Irregular allowance and deduction types data example. Shows sample data linking irregular allowance and deduction types to salary slip items.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301



EMP_ALLOWANCE_DEF

EAD_PK	EMP_ID	AT_PK	FROM	TO	AMOUNT
1	100	1	09-01-2020	NULL	500
2	100	2	05-01-2023	NULL	150
3	102	2	06-01-2021	NULL	150

EMP_MONTHLY_ALLOW

EMP_ID	MONTH	YEAR	AMOUNT	AT_PK	EAD_PK
100	3	2025	500	1	1
100	3	2025	150	2	2
102	3	2025	150	2	3

EMP_ID	SSI_PK	AMOUNT	MONTH	YEAR
100	1	650	3	2025
102	1	150	3	2025

Fig. 24. Employee allowance definitions and posting example. Shows sample data for employee allowance definitions and their resulting monthly posted values (in Jordanian Dinars, JD).

5.2. HRPIS Security, Redundancy, and Access Controls

The HRPIS is deployed on a secure, virtualized infrastructure designed for high availability, fault tolerance, and data integrity. The application and database run on separate virtual machines hosted on distinct physical servers, with backups replicated twice daily across geographically distributed storage systems. Annual archival backups are encrypted and stored on tapes in multiple secure locations under independent administrative control, with access limited to authorized personnel, following enterprise best practices. Physical servers, network switches, and storage controllers are fully redundant to ensure uninterrupted service [40]. The system operates within an internal VPN protected by a firewall, and all external access requires a secure, authenticated connection. Credentials are managed via *Microsoft Active Directory* [41], which stores hashed passwords and enforces centralized access control.

Access and operations are strictly role-based. IT administrators manage backups, database administrators handle the database, and HR and payroll functions are restricted to their respective roles. This separation prevents unauthorized access outside each domain, while mutual oversight across teams reinforces strong internal controls.

This multi-layered, distributed, and role-segregated architecture eliminates single points of failure, protects against unauthorized modifications, and ensures compliance with institutional and regulatory standards for confidentiality, retention, and auditability. While blockchain-based HR systems [42] rely on smart contracts and immutable ledgers for processes like payroll, HRPIS achieves comparable integrity through redundancy, and multi-admin oversight, with the additional benefit of rapid operational recovery. Blockchain remains superior for preventing undetectable tampering.

5.3. Operational Integrity and Transaction Auditing

To ensure institutional accountability, a background logging mechanism captures all sensitive operations across the system's screen flows. These data are exposed to management via a dedicated *Operations Log Report*, providing a transparent audit trail that identifies the user and timestamp for every critical execution. By converting raw system logs into actionable reports, the framework ensures the privacy, accuracy, and legality of records while enabling management to monitor staff performance, detect potential data breaches, and rapidly trace the source of data errors.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

6. Salary Computation Method

The HRPIS streamlines payroll processing by enabling the bulk generation of employee salary slips in any given month via the screen shown in Fig. 25. After salaries are computed, the system allows a *Payroll Admin* to verify calculated values before official posting. Once posted, the salary of an employee is considered ready for bank transfer. Additionally, the system provides flexibility by permitting salary computation for individual employees either monthly or retroactively for previous months through the screen shown in Fig. 26.

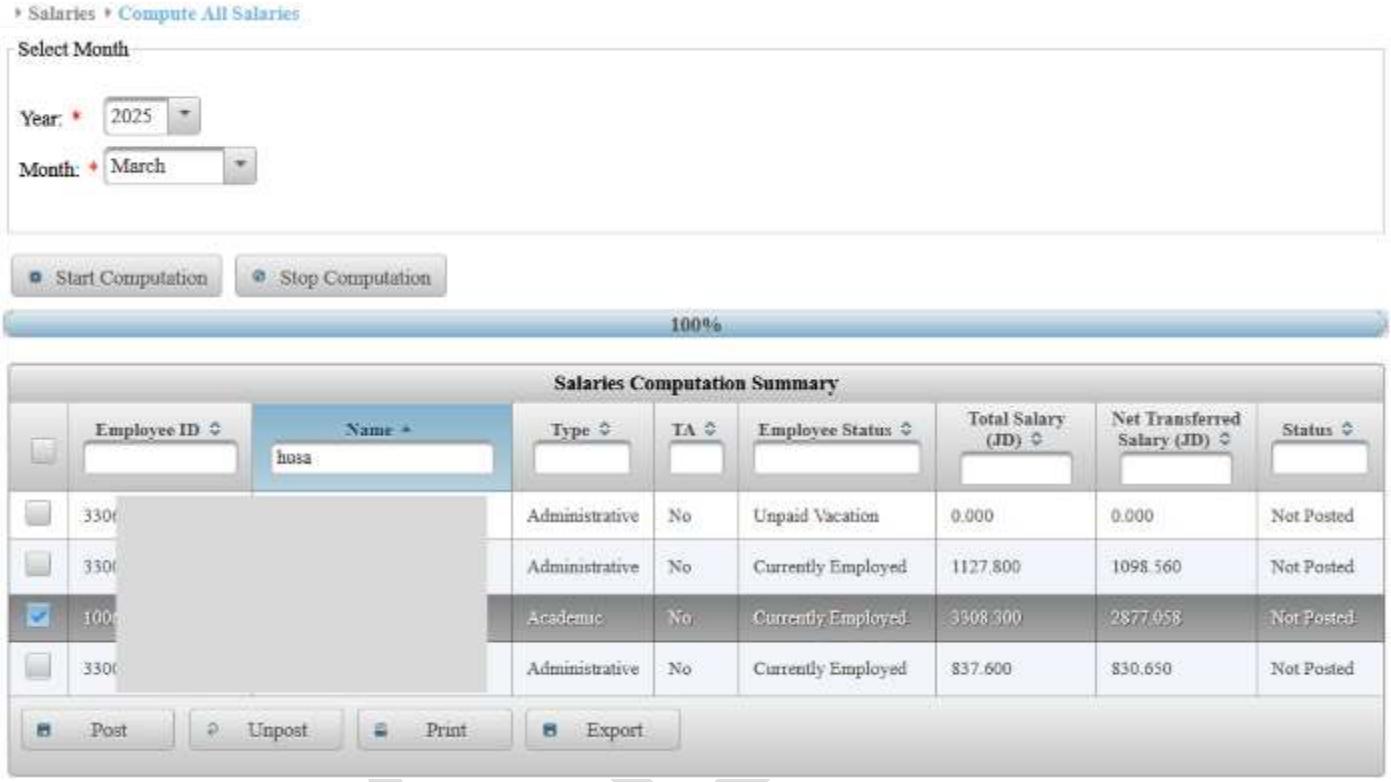


Fig. 25. Bulk salary slip generation screen. Enables computing and verifying employee salaries for a given month before posting.

In this context, the pseudocode for the salary computation loop used to generate monthly salary slips (see Fig. 27) for all employees is presented in Fig. 28. Accordingly, after entering the employee loop starting at line 1, the necessary variables for the current employee are initialized between lines 2 and 7. Next, the amounts of the allowance-type salary slip items (e.g., *Basic Salary* and *Extra Allowances*) defined in the salary slip template are computed for the current employee within the *foreach* loop that starts at line 8 and ends at line 14. Moreover, if the salary slip item is marked as taxable (as checked at line 12) or designated to be included in the social security contributions (as checked at line 13), then the computed *allowanceAmount* value will be added to the *totalTaxedIncome* value at line 12 and *totalIncomeIncludedInSocialSec* value at line 13, respectively. Following this, the amounts of the deduction-type salary slip items (e.g., *Income Tax* and *Health Insurance*) defined in the salary slip template for the employee are computed in the *foreach* loop that begins at line 15 and ends at line 18. Finally, the net salary for bank transfer is computed at line 19 by subtracting *totalDeductions* from the *totalAllowances*.

Specifically, Fig. 29 presents the SQL statement used to retrieve the university allowance amount for an academic employee from the ACA_UNIV_ALLOWANCE database table (see Fig. 21), which contains the information shown in Table 4. It is worth noting that the values of other allowances discussed in Subsection 4.1 can be queried in a similar manner. The SQL statement shown in Fig. 30 retrieves the total amount of all irregular allowance types linked to the *Extra Allowances* salary slip item for the employee with ID 100 for the month of March 2025. Furthermore, using the *totalAllowances* amount from Fig. 28 along with the employee's yearly tax exemption, the monthly income tax can be calculated using the *computeMonthlyTax* method shown in Fig. 31. It should be noted that the yearly tax exemption amount can be retrieved from the TAX_LAW_EXEMPTION database table (see Fig. 21 and Fig. 10), based on the employee's tax filing status. In turn, the *monthlyIncome* (i.e. *totalAllowances*) will be broken down into different values, based on the tax law brackets (see Fig. 11), at lines 7 to 11 in the pseudo code shown in Fig. 31. Each of these values is then multiplied by its corresponding tax rate to compute the monthly *taxAmount*, following the formula shown at line 12 in Fig. 31.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Fig. 26. Individual salary computation screen. Supports computing salaries for individual employees for current or previous months.

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Finance Department
Payroll Section
Salary Slip for March/2025

Employee ID:		**Employee Type:**	Administrative
Employee Name:		**Employee Classification:**	Total Contract
Work Place:		**Employee Grade:**	N.A.
Bank to Transfer to:		**Employee Grade Category:**	N.A.
IBAN:		**Employment Book Date:**	

Salary Details in Jordanian Dinars

Salary Details	Amount in JDs	Salary Deductions	Amount in JDs
Total Salary	1240.000	Income Tax	43.287
Administration Allowance	0.000	Income Tax Overtime	0.000
International Program Incentives	0.000	Social Security	79.820
International Program Incentives 50% from Administration Allowance	0.000	Telephones Bills	12.408
Health Insurance Refunds	0.000	Health Insurance	35.220
Misc Differences	0.000	Saving Fund 5% from Basic Salary	0.000
Over Time	0.000	Cancer Care Insurance	12.000
Extra Allowances	151.200	Other Deductions	18.000
		Social Committee Fees	0.000
		Employees Association Fees	0.000
		Income Tax Extra Allowances	0.000
Total:	1391.200	Total:	200.735
Net Transferred Salary:	1190.465		

Signature and Stamp of Payroll Section:

Back Print

Fig. 27. Sample salary slip. Shows an example of a generated salary slip for an employee, including allowances, deductions, aggregated bases, and the resulting net salary for the selected pay period.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

```
1   foreach employee in employees do
2       totalAllowances = 0.0;
3       totalDeductions = 0.0;
4       totalTaxedIncome = 0.0;
5       totalIncomeIncludedInSocialSec = 0.0;
6       netSalary = 0.0;

7       - build salarySlipTemplate object based on employee type

8       foreach allowance in salarySlipTemplate.allowances do
9           - invoke proper method to compute allowanceAmount
10          allowance.setAmount(allowanceAmount);
11          totalAllowances += allowanceAmount;
12          totalTaxedIncome += (allowance.isTaxable()) ? allowanceAmount : 0.0;
13          totalIncomeIncludedInSocialSec += (allowance.isIncludeInSocialSec()) ? allowanceAmount : 0.0;
14      end // foreach allowance

15      foreach deduction in salarySlipTemplate.deductions do
16          - invoke proper method to compute deductionAmount
17          deduction.setAmount(deductionAmount);
18      end // foreach deduction

19      netSalary = totalAllowances – totalDeductions;
20  end // foreach employee
```

Fig. 28. Salary computation pseudocode. Presents the pseudocode for computing allowances, deductions, and net salary for employees.

```
SELECT AMOUNT
FROM ACA_UNIV_ALLOWANCE
WHERE EMP_ACA_GRADE_ID=6 -- Professor
AND TO_DATE('01-Mar-2025', 'DD-MM-YYYY')
BETWEEN FROM_DATE AND NVL(TO_DATE, TO_DATE('01-01-2221', 'DD-MM-YYYY'));
```

Fig. 29. University allowance SQL query. Retrieves the university allowance for a professor in March 2025 from the database.

```
SELECT SUM(AMOUNT) AS ALLOWANCE_AMOUNT
FROM EMP_MONTHLY_ALLOW
WHERE EMPLOYEE_ID       =100
AND MONTH               =3
AND YEAR                =2025
AND ALLOWANCE_TYPE_ID IN
(SELECT ALLOWANCE_TYPE_ID FROM ALLOWANCE_TYPE
 WHERE SALARY_SLIP_ITEM_ID=41); -- Extra Allowances
```

Fig. 30. SQL statement used to compute the total amount of the irregular allowances that are linked to the *Extra Allowances* salary-slip item.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

```
computeMonthlyTax(monthlyIncome, taxExemptionAmount)
begin
    yearlyIncome = monthlyIncome * 12.0
    taxableAmount = max(yearlyIncome - taxExemptionAmount, 0.0)
    taxAmount = 0.0

    if taxableAmount > 0.0 then
        amount25 = max(taxableAmount - 20000.0, 0.0)
        amount20 = max(taxableAmount - amount25 - 15000.0, 0.0)
        amount15 = max(taxableAmount - amount25 - amount20 - 10000.0, 0.0)
        amount10 = max(taxableAmount - amount25 - amount20 - amount15 - 5000.0, 0.0)
        amount5  = max(taxableAmount - amount25 - amount20 - amount15 - amount10, 0.0)

        taxAmount = (0.05*amount5 + 0.1*amount10 + 0.15*amount15 + 0.2*amount20 + 0.25*amount25) / 12
    end

    return taxAmount
end
```

Fig. 31. Monthly tax computation pseudocode. Shows the pseudocode for the *computeMonthlyTax* method, illustrating the calculation of income tax from the taxable base using applicable exemptions and tax brackets.

7. Knowledge-Driven, Generalizable Payroll Design Patterns

Payroll computation in HRPIS is organized around reusable design patterns that encode payroll knowledge through structured data and deterministic computation steps. This design separates elements that vary across institutions, such as regulatory parameters, contractual values, and employee attributes, from stable operational elements, including computation loops, aggregation mechanisms, and verification logic.

The template-driven architecture provides structural blueprints for salary slips, where each template contains an ordered set of items linked to salary slip items representing allowances or deductions. Each salary slip item is annotated with control flags, such as taxable or inclusion in social security, and is associated with a resolver, a modular plug-and-play method implementing item-specific computation. Formally, a resolver is defined as:

$$R_s(e,t) \rightarrow amount(s,e,t) \qquad (1)$$

Where e represents the employee object containing ID, type, classification, grade, category, years of service, dependents, commissions, and other relevant attributes, and $t = (\text{year}, \text{period})$ represents the pay period. Resolvers encapsulate all table lookups, joins, and calculation logic, allowing templates to be reused across employees. Templates are assigned by employee type and classification, reducing setup effort and maintaining adaptability.

7.1. Template-Driven Item Structuring and Aggregation

Templates define the structural composition of a salary slip, including:

- Sets of allowances and deductions (A_T, D_T)
- Ordering of items for consistent presentation
- Grouping of irregular items into a single template entry
- Linkage to resolvers for computation

$$Template_T(e) = \text{s} \in A_T \cup D_T \mid \text{s linked to resolver } R_s \qquad (2)$$

This structure ensures consistent presentation and reusability across employees. The engine retrieves template items in order, allowing each resolver to compute its value and contribute to relevant aggregation bases.

7.2. Resolver-Driven Computation and Aggregation

Net salary is determined through the engine loop, which first processes allowances and then deductions:

$$NetSalary(e,t) = \Sigma_{s\in A_T} R_s(e,t) - \Sigma_{s\in D_T} R_s(e,t) \qquad (3)$$

Control flags annotate items that contribute to aggregated bases used by other resolvers. For a given base j, such as taxable income or social security, the aggregation is computed as:

$$Base_j(e,t) = \Sigma_{s\in A_T} f_j(s) * R_s(e,t) \qquad (4)$$

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

where $f_j(s) = 1$ if item s contributes to base j and 0 otherwise. Taxable and social security bases are aggregated based on a dedicated binary flag per item, ensuring deterministic calculations for bulk and retroactive payroll.

7.3. Recurring Resolution Patterns

Payroll items follow a few recurring resolution patterns. These patterns separate variable elements such as laws, rates, contracts, or employee attributes from stable computation logic. They capture dominant regulatory structures and are designed to be portable across institutions and jurisdictions. For items such as income tax and health insurance, the same resolver and table-driven logic apply globally. Only data such as contracts, rates, exemptions, or brackets require adjustment. Deviations are accommodated through localized resolver extensions while preserving the engine, aggregation logic, and templates. When regulations match these patterns, portability is achieved through configuration alone.

7.3.1 Single-Table Resolution Pattern

Regular allowances or deductions, such as basic salary or fixed institutional allowances, are resolved through a single temporal lookup table. The resolver selects the applicable value based on employee attributes and the effective pay period:

$$R_{BasicSalary}(e,t) = v, \text{ where } v \text{ is selected according to } e \text{ and } t \qquad (5)$$

For rows r in the lookup table with the same attribute combination, effective intervals are non-overlapping and coverage is complete, ensuring that:

$$(r_1.attrs = r_2.attrs) \Rightarrow r_1 \cap r_2 = \emptyset, and\ \forall(e,t),\ \exists!\ v$$

This pattern isolates salary scale structures within data. Pay increments (revised steps) or eligibility boundaries (classification thresholds) are managed via table updates rather than code. New unique institutional allowances are supported by linking a new slip item and its specific lookup table to this resolver type, preserving the stability of computation loops and aggregation logic.

7.3.2 Base Aggregation Pattern

Many regulatory computations depend on intermediate monetary bases, such as taxable income or social security contribution bases. These bases are constructed from values resolved through patterns such as the *Single-Table Resolution Pattern*, by aggregating allowance values annotated with control flags:

$$Base_j(e,t) = \sum_{s \in A_T} f_j(s) \cdot R_s(e,t) \qquad (6)$$

This is ensured when all contributing items are resolvable and flags are binary:

$$\forall s \in A_T,\ f_j(s) = 1 \Rightarrow R_s(e,t) \text{ is defined, } f_j(s) \in \{0,1\}$$

Control flags determine an item's contribution to a specific base. This structural, jurisdiction-agnostic pattern allows changes in item participation via salary slip item flag updates. New aggregation categories are added by introducing a base to the engine loop.

7.3.3 Income Tax Resolution Pattern

Income tax computation follows a law-driven, progressive structure dependent on exemptions and tax brackets. The resolver consumes the taxable base from the *Base Aggregation Pattern* and applies statutory exemption rules E and cumulative bracket rates:

$$R_{IncomeTax}(e,t) = \sum_{j=1} max\big(0, min\big(Base_{Tax}(e,t) - E, U_j\big) - L_j\big) \cdot R_j \qquad (7)$$

The resolver annualizes the taxable base for period t, applies yearly exemption and bracket tables, and converts the resulting tax back to the payroll period. It completely partitions the taxable income into segments defined by lower L_j and upper U_j limits, applying the marginal rate R_j to the portion within each bracket.

This assumes that a unique tax law applies for each period t:

$$\forall t,\ \exists!\ TaxLaw(t)$$

Jurisdictional differences are expressed through exemption tables, bracket definitions, and filing parameters. Where tax laws align, tables and resolvers are reused. Deviations from this structure are handled by introducing a new resolver and, if needed, a new table structure, while preserving the engine and aggregation logic. This ensures the pattern is globally generalizable.

7.3.4 Social Security Resolution Pattern

Social security and similar contributions are computed by applying statutory rates to a predefined base. The resolver uses a base from the *Base Aggregation Pattern* and applies employee and employer rates that may vary by job classification or risk category:

$$R_{SS}(e,t) = Base_{SS}(e,t) \cdot \text{rate}(e,t) \qquad (8)$$

This requires that rates are defined over employee job classification and period *t*, and exactly one rate applies:

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

$$\forall (e,t),\ \exists!\ rate(e,t)$$

Rate table updates enable portability via data configuration preserving the contribution structure. Deviations from this structure are handled by introducing a new resolver and, if needed, a new table structure, while keeping engine and aggregation logic intact.

7.3.5 Health Insurance Resolution Pattern

Health insurance resolution abstracts multi-entity contractual logic into a selection function f:

$$R_{HealthIns}(e,t) = f(e,t) \qquad (9)$$

This implies temporally consistent contract applicability and linkage, such that for each employee-period case with active health insurance coverage, exactly one contract applies, and the employee and any covered dependents are linked to that contract for that period:

$$\forall (e,t) \text{ with } \mathrm{Coverage}(e,t),\ \exists!\ \mathrm{contract} : \mathrm{Link}(\mathrm{e}, \mathrm{contract}, \mathrm{t}) \wedge (\forall\ \mathrm{d} \in \mathrm{CoveredDependents}(\mathrm{e}, \mathrm{t}), \mathrm{Link}(\mathrm{d}, \mathrm{contract}, \mathrm{t}))$$

The function f encapsulates join logic, contract determination, health insurance tier selection, and temporal filtering. The architecture leverages contract table definitions to support diverse providers through pure data configuration. This stable framework accommodates active provider contracts with multi-level tiers, cost-sharing tables splitting costs between the university and employee for self or family coverage, and age-bracketed fee scales.

Because the engine generalizes these requirements via configuration, logic updates are rarely necessary. Structural deviations simply trigger a new resolver and table structure, preserving core engine logic.

7.3.6 Irregular Item Resolution Pattern

Irregular payroll items are employee-specific, non-periodic values, such as performance rewards or special adjustments, explicitly posted rather than derived from formulas. Each posted value carries a type identifier. For a salary slip item s, the resolver aggregates all values for employee e in period t from the multiple irregular types linked to that item:

$$R_s(e,t) = \sum_{i \in P(e,t),\, type(i) \in Types(s)} value(i) \qquad (10)$$

This requires that all posted items included in the aggregation belong to the payroll period, are correctly typed, and have defined values:

$$\forall\ i \in P(e,t),\ type(i) \in Types(s) \Rightarrow value(i) \text{ is defined}$$

This pattern is fully data-driven. Regulatory changes and institutional adoption affect only definitions and linkages. Templates, resolvers, and engine execution remain unchanged.

7.4. Adaptation Playbook

The system adapts to new institutions through an 8-step configuration sequence that maps local rules to invariant design patterns:

1) **Attribute Cataloging:** Set employee identifiers (e.g., seniority, commissions, dependents) as primary calculation inputs.
2) **Regulation Mapping:** Load local salary scales. The system adapts to unique institutional structures by defining new tables and linking them to specific salary items and resolvers.
3) **Compliance Configuration:** Map tax laws and insurance via management screens. If a jurisdiction requires a new model, the framework allows adding tables and resolvers without altering the core engine.
4) **Slip Item Control:** Instantiate salary slip items and assign *Control Flags* to them to establish deterministic aggregation bases for taxes and social security.
5) **Irregular Mapping:** Link non-periodic types such as bonuses or fees to slip items, enabling external value injection via manual posting or Excel.
6) **Blueprint Templating:** Construct structural templates for diverse employee types to utilize the same engine and reduce setup steps.
7) **Integrity Validation:** Back-test against historical datasets and apply regression tests for rule changes to ensure accuracy under evolving rules.
8) **Operational Oversight:** Use logging, reporting, and traceability to ensure auditability, accountability, and performance monitoring.

7.5. Portability Evidence and Boundary Conditions

The architecture achieves portability by decoupling invariant engine logic from fluctuating parameters via four pillars. *Multi-Source Temporal Resolution* ensures integrity by resolving rates via tax law, health contract, and salary scale effective dates, using the slip date to resolve values without code changes. *Regulation Mapping* adapts to institutions by loading local scales and linking tables to specific items and resolvers. To handle volatility, *Irregular Generalization* treats non-periodic items as external injections

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

via manual posting or Excel. Crucially, *Control Flags* enable every salary slip item to automatically aggregate into tax and social security bases without custom formulas.

This framework assumes regulatory rules are deterministic and decomposable into progressive, flat, or lookup structures. Radical jurisdictional shifts are managed at the elastic boundary where legal volatility is handled by introducing a new resolver method and database tables. These are linked via UI configuration to a salary slip item to expand the functional perimeter while maintaining structural stability. Model resilience was validated at GJU by processing six insurance contract iterations across four providers within the fixed HRPIS database schema, with each new contract defined through the relevant HRPIS configuration screens. Additionally, the engine supports two tax laws added progressively and applied retroactively, implemented entirely through configuration while leaving the core engine logic unchanged.

7.6. Regression Tests for Rule Changes

Rule changes (e.g., updates to lookup values, brackets, rates, contracts, or eligibility conditions) are validated through regression tests defined over employee-period cases. Each test case is evaluated against the rule configuration applicable to its validity interval.

Updates may take multiple forms. Lookup tables are revised by inserting new rows with updated values and later effective start dates while preserving prior rows for earlier periods. Contracts are defined over explicit validity intervals using start and end dates, and tax laws are introduced as distinct rule sets effective from a specified year onward. These changes are primarily handled through data configuration, with resolver modifications introduced only when they cannot be expressed through existing table structures.

Regression tests recompute salary slip outputs and compare them against expected results. Previously defined cases ensure consistency under the rule configurations applicable to their periods, while newly introduced cases validate the effects of rule changes for the periods to which updated rules or implementations apply.

Temporal integrity is maintained through non-overlapping validity intervals and complete coverage of applicable periods. Periods before a change continue to resolve using previously applicable rules, while periods from the effective date onward use the updated or newly introduced rules. Each rule change extends the regression suite, enabling cumulative validation of rule evolution.

7.7. Traceability and Provenance

The system provides comprehensive traceability from regulatory definitions to computed payroll outputs. Each salary slip item can be explained in terms of the rule configurations and computational logic applied at the time of its generation.

Traceability is supported by integrated auditing mechanisms across all system layers. Updates to rule tables, including insertions, modifications, and deactivations, are recorded with timestamps, user identifiers, and operation details. Salary slip outputs are stored as immutable snapshots at computation time, capturing the exact values and context used, along with the user who triggered the computation and the corresponding timestamp. These records remain unaffected by subsequent rule or configuration changes.

Resolver implementations are maintained under version control, with full authorship and revision history. Regulatory changes that trigger updates to rule tables or resolver logic are managed through structured service requests, linking each implemented change to its originating regulatory source. Code revisions reference the corresponding service request and regulatory basis, along with timestamps and authorship, and are reflected in updated system documentation where required.

Together, these mechanisms provide a complete and verifiable audit trail, ensuring that payroll results are reproducible, explainable, and attributable to their regulatory and implementation origins, thereby supporting operational accountability and retrospective analysis.

8. RAG-based Chatbot for Payroll-Specific User Support

Chatbots are AI-driven software tools that offer interactive, instant, and informative responses to user queries. They can be integrated into HR and payroll management systems to assist employees by responding to questions on HR regulations, payroll allowances and deductions, tax laws, and health insurance policies. This minimizes routine support efforts, enabling staff to focus on more important tasks. At the same time, chatbots ensure employees receive consistent and timely answers, thereby improving operational efficiency and user satisfaction.

Such chatbots mainly rely on pure Large Language Models (LLMs), which are advanced deep learning models that have been trained on extensive amounts of text (e.g., books, manuals, policies, and regulations) to understand patterns and relationships in human language. Although pure LLMs can generate text, explain concepts, and assist in chatbot interactions, their dependence on generic training data gathered from different and primarily unrelated sources can lead to responses that are incorrect, outdated, ambiguous, incomplete, or biased. To overcome these challenges, an LLM can be augmented with an external information retrieval system to form what is known as a Retrieval-Augmented Generation (RAG) model. This RAG-based LLM generates responses by integrating its pre-trained data with real-time information retrieved from a database containing domain-specific documents, such as GJU regulations and the Jordanian tax laws. As a result, the model delivers responses that are accurate and timely.

Leveraging this approach, we recently integrated a RAG-based chatbot into HRPIS to provide automated, accurate, and instant support to users. As shown in Fig. 32, integration involved two key stages: vector database generation and chatbot implementation.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

In the first stage, the English versions of the current GJU payroll regulations [2] and the Jordanian tax law [43] are loaded. The source filename of each document is attached to its object as metadata before chunking, ensuring that every resulting chunk retains a reference to its original document for traceability. The text is then split into smaller, manageable chunks using *LangChain's RecursiveCharacterTextSplitter* [44]. Two FAISS [45] vector databases are constructed from these chunks: one using the *All-MiniLM-L6-v2* embeddings model [46], which is lightweight and efficient, and another using the *All-MPNet-Base-v2* model [47], which provides higher-dimensional, richer contextual representations. This stage is largely static and is repeated only when updated source documents are released or embedding models are changed. The resulting FAISS databases are stored on disk for later use by the chatbot.

In the second stage, the chatbot loads the appropriate pre-generated FAISS database according to the embedding model selected at startup. Administrative parameters such as the embedding database, the top-k retrieval value (typically 15), the maximum token length (typically 1024), and the temperature (typically 0.3) are initialized to ensure consistent and deterministic system behavior. When a user submits a query, the system may apply a simple query transformation by appending guiding text to the input to improve retrieval effectiveness. The transformed query is embedded, and the system retrieves the most relevant document chunks from the FAISS index, which contains only publicly available payroll and tax regulations. This ensures that no sensitive employee or student information is accessible to the chatbot.

Once the relevant chunks are retrieved, the chatbot combines them with the controlled prompt and passes this input to the language model, such as *Qwen2.5-72B-Instruct* [48], to generate a response. The prompt directs the model to rely strictly on the retrieved content, include citations when possible, and respond with "*I do not know based on the documents*" if the information is absent. This approach reduces hallucinations, maintains traceability, keeps responses aligned with current regulations, and mitigates prompt injection or jailbreak attempts by constraining the model to the approved regulatory domain.

After generating a response, the system computes a retrieval-confidence score based on chunks already retrieved by FAISS. Since FAISS calculates query-to-chunk similarity during retrieval, the system uses the top-1 similarity value, normalized between zero and one, as the confidence score to indicate response support. Scores below 0.3 are flagged to attach a cautionary note or prompt manual verification. Planned rate limiting will prevent system overload and maintain performance stability. Each response is logged with its query, timestamp, and citations to support monitoring without compromising privacy. Once validated, the response is delivered to the user, and the system awaits the next query, repeating the sequence of transformation, retrieval, prompting, generation, and validation until the session ends.

To keep the chatbot's knowledge current, HR staff notify IT whenever new regulatory versions are released. IT then regenerates the FAISS database with updated documents and restarts the chatbot, ensuring responses reflect the latest regulations.

While the chatbot currently uses subscription-based Hugging Face [49] LLMs via cloud APIs, this setup is for proof-of-concept only. Long-term, GJU will host the top-performing LLM on its GPU-accelerated research servers. This in-house approach eliminates recurring costs, ensures full system control, and maximizes data privacy and reliability.

A methodology for training and evaluating the RAG-based chatbot on GJU regulations and the Jordanian tax law is presented in Subsection 10.2. The publicly available documents were preprocessed and vectorized to enable efficient retrieval using both FAISS and BM25. Three LLMs hosted on *Hugging Face* were evaluated with optimized parameters to determine which model achieved the best balance of accuracy, completeness, and response time. The evaluation employed ablation studies on a representative subset of the benchmark, varying retrieval type, embeddings, chunk size, and top-k values, with each question submitted three times per model to assess performance, consistency, and response efficiency. Expert reviewers in HR and finance scored the responses against predefined answer keys, ensuring a fair, consistent, and domain-specific evaluation. The best-performing configuration was subsequently evaluated on the full benchmark, while system robustness was assessed through a separate empirical red-teaming evaluation under adversarial scenarios.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

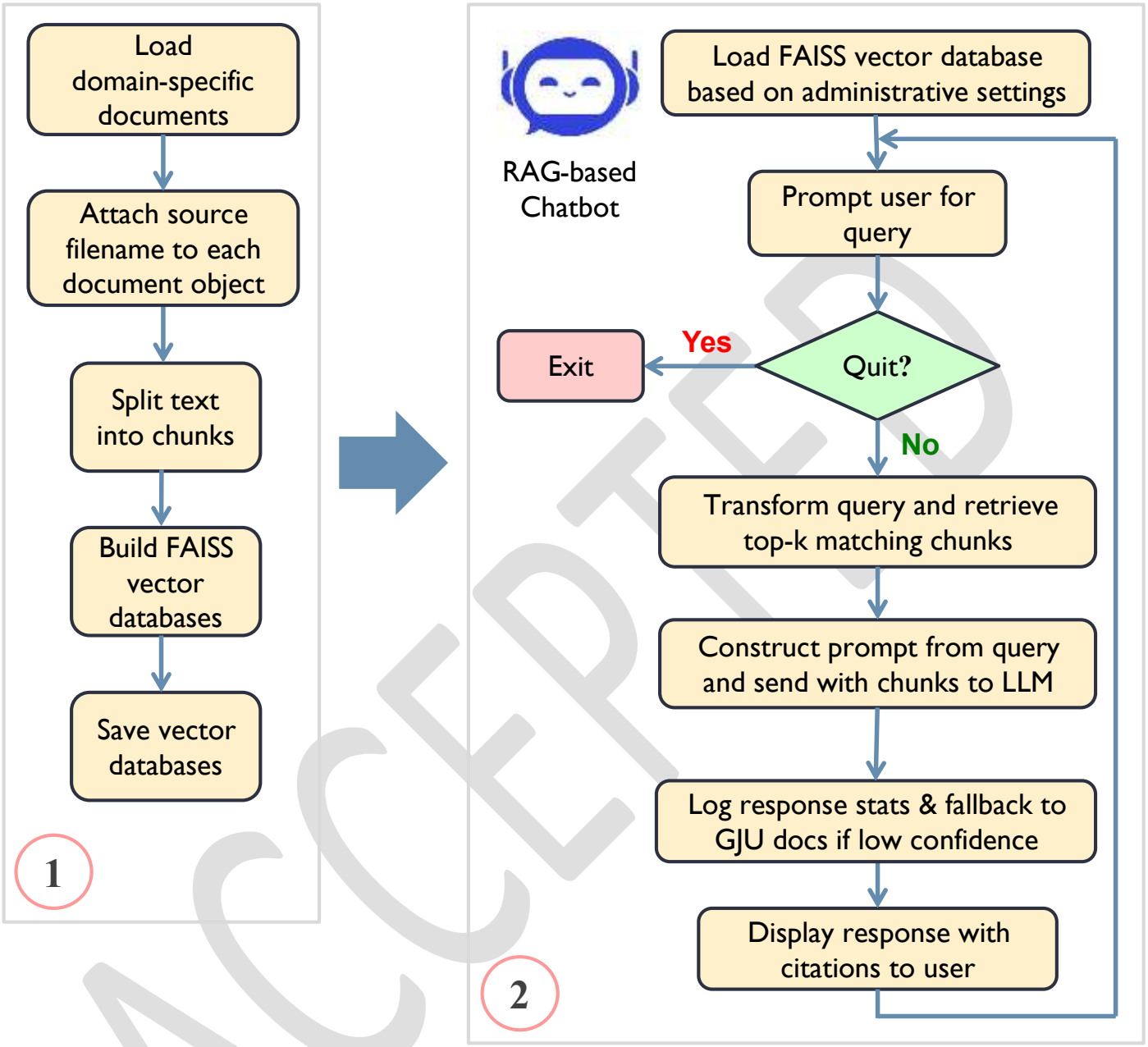


Fig. 32. RAG-based chatbot integration workflow. Illustrates the two-stage process of vector database generation and chatbot implementation for payroll and tax support.

9. Comparison to Related Works

As shown in Table 8, the primary payroll management features discussed in this study are compared with those in the works that provided technical details in this regard. Features are compared across eight categories: methodology, integration, payroll setup, slip templates, irregular items, salary computation, tax slips, and AI support.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 8 Comparison of payroll management features. Compares key features across selected works and this work based on methodology, integration, setup, templates, irregular items, computation, tax slips, and AI support.

Category	Feature	Study									
		This Work	[7]	[8]	[9]	[10]	[12]	[13]	[17]	[18]	[19]
Methodology	Scrum-based Agile Methodology	Yes	No	No	No	No	No	No	No	No	No
Integration	Integration with HR System	Yes	No	No	No	Yes	No	Yes	No	No	Yes
Payroll Data Setup	Tax Law	Yes	No	No	No	No	No	No	Yes	Yes	Yes
	Social Security	Yes	No	No	No	No	No	No	No	No	Yes
	Health Insurance	Yes	No	No	No	No	No	No	Yes	No	Yes
	Basic Allowance/Deduction Lookup Tables	Yes	No	No	No	No	No	No	No	No	No
	Per Employee Salary Setup	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes
Salary Slip Templates	Multiple Templates	Yes	No	No	No	No	No	No	No	No	No
	Customizable	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
	Fielded Items	Yes	No	No	No	No	No	No	Yes	Yes	Yes
Irregular Allowance & Deduction Types	Reusable Period Based Def.	Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes
	Many-to-one Grouping	Yes	No	No	No	No	No	No	No	No	No
	Importable values	Yes	No	No	No	No	No	No	No	No	No
	Bulk Amount Generation	Yes	No	No	No	No	No	No	No	No	No
	Crosschecking	Yes	Yes	No	No	No	No	No	No	No	No
Salary Calculation	Single Generation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Bulk Generation	Yes	No	No	No	Yes	No	No	Yes	Yes	Yes
	Retroactive Generation	Yes	No	No	No	No	No	No	No	No	No
	Crosschecking	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Income Tax Slips	Single Generation	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes
AI-Enabled User Support	RAG-based Chatbot	Yes	No	No	No	No	No	No	No	No	No

Accordingly, this study is distinguished by the following features: adopting a Scrum-based agile methodology, maintaining basic allowance and deduction lookup tables, offering multiple salary slip templates, linking multiple allowance or deduction types to a single salary slip item, importing allowance or deduction values from an Excel file, posting allowance or deduction amounts based on predefined definitions, generating employee salaries retroactively, and utilizing a RAG-based chatbot to reduce the support burden on the finance staff.

Notably, although the studies in [15, 16] described the adoption of agile or scrum frameworks in the development of payroll information systems, neither discussed technical details regarding the design and implementation of such systems. Therefore, they were not included in this comparison. Furthermore, while some studies highlighted the importance of supporting features related to tax laws, health insurance, and social security, however they did not discuss the needed user interfaces, rules, database tables, and

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

algorithms to develop such functionalities. Additionally, few studies considered important aspects such as integration with HR, crosschecking, and bulk generation.

When positioning our solution relative to commercial payroll tools like [17, 19], we similarly support health insurance, tax rules, as well as individual and bulk salary slip generation. Our system offers greater flexibility in handling regulatory tables, irregular allowances, and multiple salary slip templates, which enables largely automated payroll setup. In contrast, these commercial tools require extensive manual customization for different employee configurations, making the process time-consuming and prone to errors. Unlike the commercial tools, we also support retroactive salary generation and provide RAG-based chatbot assistance, enhancing both operational efficiency and user support.

In summary, this study not only proposes detailed solutions related to general aspects pertaining to payroll information systems, but also introduces unique concepts that are important to enhance the flexibility, usability, efficiency, accuracy, and security of such systems. Moreover, it provides a comprehensive methodology and detailed information covering functional analysis, user interface design, database design, implementation specifics, and application architecture to support the design and development of complex payroll information systems.

10. Validation and Results

10.1. Validation of the HRPIS

10.1.1. Empirical Validation of the Scrum Process and System Quality Assurance

The payroll module was developed over eleven four-week Scrum sprints, following earlier HR components (not detailed here) that the payroll system builds upon. The empirical metrics, including the burn-up chart (Fig. 33) and defect metrics chart (Fig. 34), demonstrate how Scrum guided incremental development, progressive requirements, and quality assurance. Each sprint included 2–7 tasks distributed among 2–3 developers. Initially, about 16 tasks were identified, with additional tasks emerging as the project progressed, reflecting the system's complexity. This incremental development is reflected in the burn-up chart, which tracks cumulative task completion per sprint.

Defect tracking was continuous as all defects identified during development were addressed promptly. Escaped defects from previous sprints, which ranged from a minimum of 2 to a maximum of 10 in a single sprint, were systematically resolved in subsequent sprints. The defect metrics chart (Fig. 34) illustrates this process, showing defect resolution over time, including the handling of escaped defects without delay.

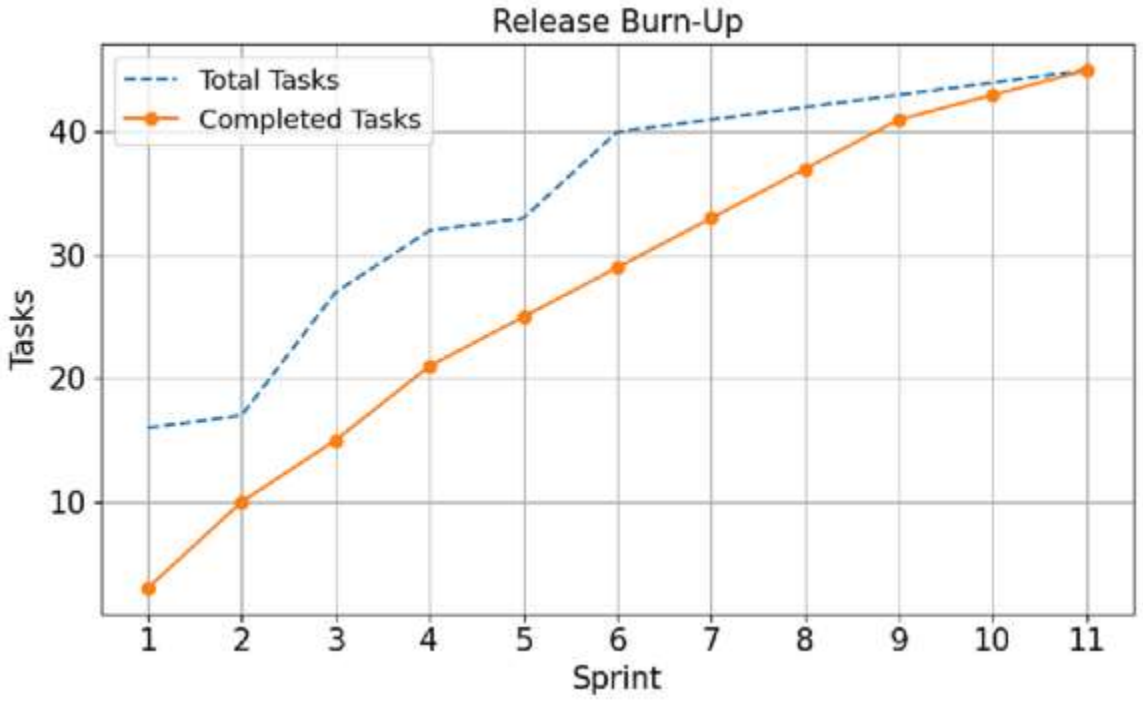


Fig. 33. Burn-up chart showing cumulative tasks completed per Sprint, tracking progress toward the project goal in Scrum.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

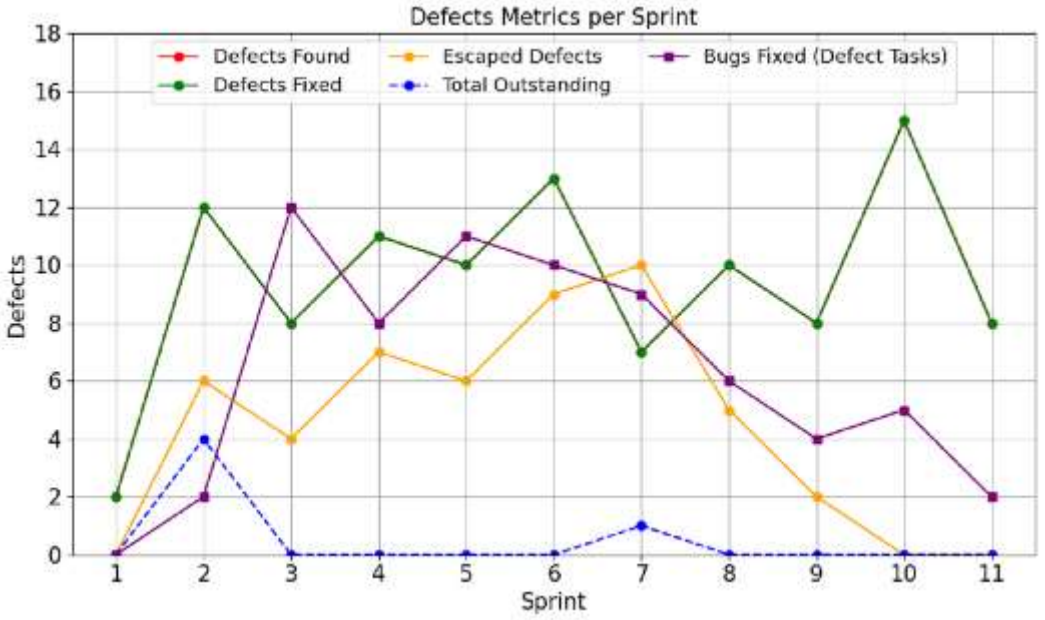


Fig. 34. Defects tracked per Sprint, showing found, fixed, escaped, and total outstanding.

Tasks ranged from developing HR and payroll setup screens to conducting *Alpha* and *Beta* tests and addressing bug fixes. More complex features, such as individual salary computation, were broken down into multiple tasks spread across two or more sprints. Testing was integrated into each increment, encompassing both unit and feature-level tests. The final two sprints included *Alpha* testing by the development team and *Beta* testing by stakeholders, covering all components from screens to features and end-to-end workflows. Feedback from testing led to immediate bug fixes and refinements, ensuring the payroll module was fully functional and aligned with user expectations. Continuous stakeholder involvement allowed features to be delivered progressively, incorporating feedback each sprint and avoiding last-minute surprises typical of waterfall approaches.

Following release, the quality of the payroll system continues to be validated through regression testing conducted by the QA team and the monitoring of bug and enhancement tickets reported by users via the HelpDesk system [50] (see Fig. 35). For illustration, the volume of tickets in 2024 remained very low, ranging from 0–2 bugs and 0–4 enhancements per month (Fig. 36). These metrics indicate a high-quality, effective system. All reported issues are promptly resolved, with the exception of two planned enhancements scheduled for 2026, which aim to automate overtime allowance calculations and integrate them into the salary slip module. Ongoing post-release monitoring confirms that the Scrum-based development process, supported by continuous QA testing and HelpDesk tracking, effectively maintains system quality and reliability.

Overall, the Scrum-driven process, supported by empirical metrics, iterative testing, and post-release monitoring, facilitated incremental development, adaptive planning, and quality assurance, resulting in a successful HRPIS payroll module release with high customer satisfaction and minimal post-deployment issues.

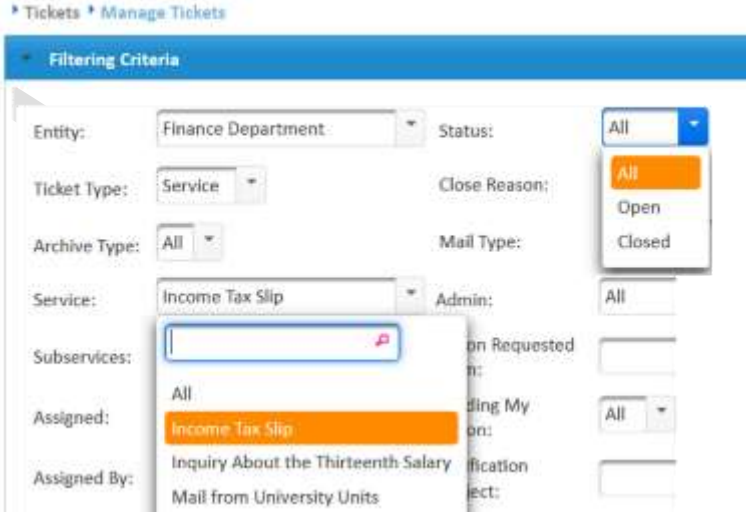


Fig. 35. Filters used to search for service tickets of the finance department via the HelpDesk system.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

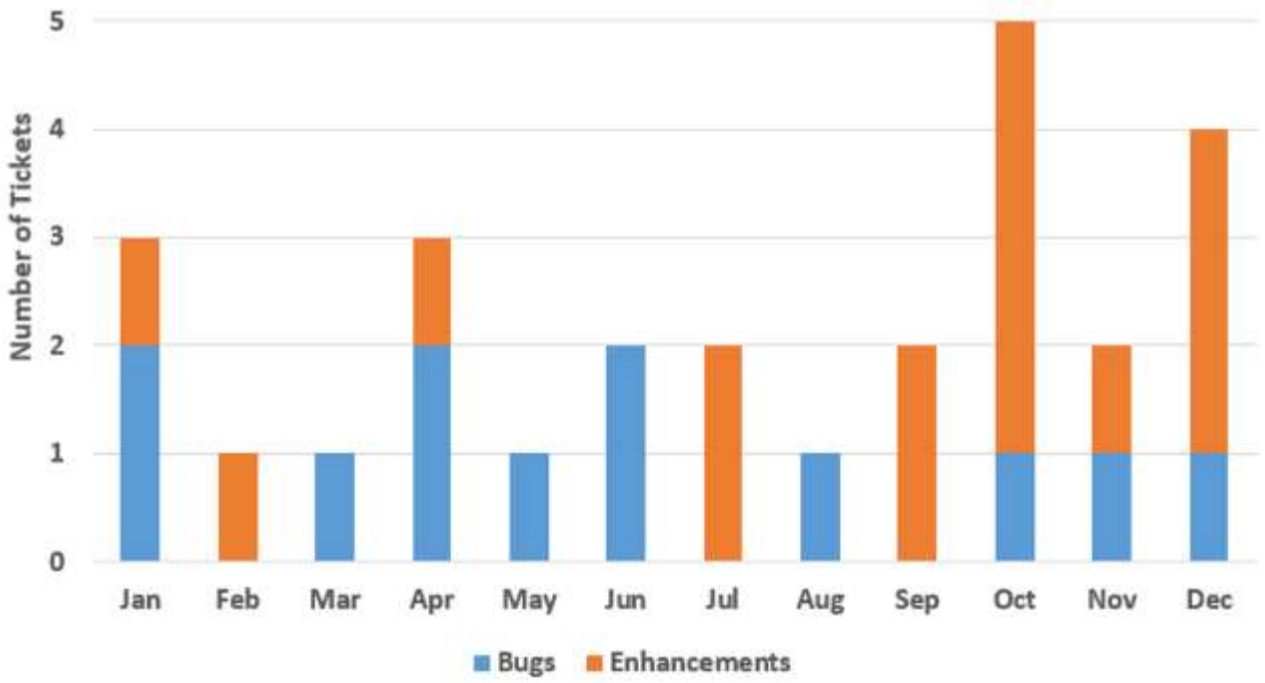


Fig. 36. Bug and enhancement tickets that were opened against the payroll features in 2024.

10.1.2. Manual Payroll Baseline

This section establishes the operational baseline of payroll processing during the manual processing period using historical operational data. The analysis is based on monthly payroll logs provided by the Finance Department, capturing performance across multiple payroll cycles in terms of workforce size, computation effort, total processing time, and correction cases. Payroll processing consisted of two main components: payroll computation and payroll output printing.

Payroll computation included all steps required to produce salary outputs, including salary and allowance calculations, statutory deductions such as income tax, health insurance, and social security, and the generation of salary slip files in spreadsheet form. These activities were performed manually using spreadsheet-based calculations and manually maintained reference rules. The process relied on fragmented data management practices, where Human Resources maintained physical records for employee-related changes such as promotions, commissions, and insurance, while Finance maintained separate payroll calculation records. This separation resulted in duplicated data handling, repeated data entry, and frequent reconciliation between sources.

Errors were identified during a post-computation validation stage, where payroll outputs were reviewed prior to finalization. Correction cases reflect discrepancies such as data entry inconsistencies, missing or incorrectly applied deductions and allowances, spreadsheet formula errors, version control issues including overwritten or outdated files, lack of standardized data structures and classification, and misalignment between HR and Finance records. This validation stage often required correction and re-verification, contributing to increased total processing time.

Payroll output printing was performed as a separate operational stage following computation. This stage involved printing and distributing salary slips and required additional coordinated manual effort. Processing effort across both computation and printing varied across cycles due to workforce availability, where staff leave, temporary allocation of personnel to meet deadlines, and periods of reduced staffing directly influenced processing duration.

As shown in Table 9, payroll processing required between 280 and 336 hours per cycle, corresponding to approximately 30 to 36 person-days of effort, with most cycles clustering around 280 hours. Payroll computation accounted for the majority of this effort, ranging from 240 to 288 hours per cycle, including the generation of salary slip outputs. Printing and distribution were performed as a separate stage and required an additional 32 to 48 hours per cycle. Correction cases ranged from 42 to 61 per cycle, reflecting recurring discrepancies identified during the validation stage. Variation across cycles is observed in both computation and printing effort, consistent with fluctuations in workload and workforce availability.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 9 Summary of manual payroll processing performance across monthly cycles, including workforce size, computation effort, total processing time, and correction cases.

Cycle	Workforce Size	Computation (person×days)	Computation Hours	Printing (days)	Printing Hours	Total Hours	Correction Cases per Cycle
1	652	6×5	240	5	40	280	48
2	659	6×5	240	5	40	280	55
3	640	5×7	280	4	32	312	47
4	634	6×5	240	5	40	280	42
5	671	7×5	280	6	48	328	61
6	663	6×6	288	6	48	336	53
7	667	6×6	288	6	48	336	57
8	661	6×5	240	5	40	280	50
9	655	6×5	240	5	40	280	52
10	654	5×7	280	5	40	320	48
11	660	6×5	240	5	40	280	44
12	658	5×7	280	5	40	320	46

10.1.3. HRPIS Operational Performance

The operational impact of HRPIS was evaluated through a longitudinal analysis of system logs covering 19 consecutive payroll cycles from February 2024 through August 2025. System performance is evaluated relative to the manually executed baseline presented in Section 10.1.2, which is derived from historical payroll logs and reflects observed processing effort across multiple pre-implementation cycles.

System execution data shows a substantial reduction in processing time. Service Level Agreement (SLA) compliance, defined as successful completion of payroll prior to the internal deadline of the last Thursday of each month, was maintained at 100 percent across all observed cycles, as shown in Fig. 37. Analysis of execution durations indicates that core payroll computation, including salary slip generation for large batches, remained at approximately 60 seconds per cycle in non-correction runs (Fig. 38), indicating highly stable system computation performance.

The latency values shown in Fig. 38 represent core system computation and exclude pre-run data preparation and post-execution validation activities. System operation follows a structured workflow in which each cycle includes data preparation, system execution, and validation prior to finalization. Data preparation involves entry and verification of inputs such as commissions, promotions, insurance subscriptions, and allowances, and required approximately 2 person days in 15 observed cycles and 3 person days in 4 cycles.

The administrative effort required during this phase is substantially lower than in the manual process due to system-supported automation and integrated data management. Statutory components such as income tax, health insurance, and social security are automatically computed based on predefined rules and stored employee profiles, requiring minimal manual intervention beyond periodic updates such as contract renewals, tax regulation changes, or updates to employee information including dependents and promotions. Many deductions and allowances are entered once and applied over extended periods, reducing repetitive data entry.

Data consistency is enhanced through unified records, where updates entered by Human Resources are immediately reflected in payroll calculations and accessible to Finance. This eliminates duplicated data handling and reconciliation effort observed in the manual process. In addition, system-controlled calculations remove reliance on spreadsheet-based formulas, reducing the occurrence of errors associated with manual editing, corrupted formulas, and inconsistent calculation logic.

Across the 19 observed cycles, 16 were completed without corrections, while 3 cycles required correction iterations during the validation phase. These correction activities resulted in extended total cycle durations. Specifically, February 2025 involved 11 correction iterations and resulted in a total duration of 1089 minutes, March 2025 involved 27 correction iterations and resulted in a total duration of 4297 minutes, and July 2025 involved 1 correction iteration with a duration of 12 minutes.

These extended durations do not represent repeated full payroll computation. The system computes payroll outputs once, after which the majority of salary records are finalized. Corrections are applied only to affected records during validation, and additional time reflects incremental verification and adjustment rather than full re-computation. As a result, extended cycle durations reflect human-in-the-loop validation effort applied to a subset of records, while core system computation time remains stable.

Once payroll computation is completed, salary slips are made available electronically through the university portal, allowing immediate access for employees. This eliminates the separate printing and distribution stage required in the manual process. While

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

preparation and validation activities contribute to total cycle duration, they do not affect core system computation performance or SLA compliance.

To account for variations in organizational size, the number of salary slips processed per cycle was tracked, as shown in Fig. 39. The recorded monthly volumes range from 605 to 947 salary slips, reflecting natural fluctuations in workforce size across reporting periods. These variations are associated with changes in staffing levels, including part-time and temporary personnel such as research assistants, employee turnover and resignations. These raw monthly counts provide the basis for the headcount normalization applied later in the comparative analysis section.

Overall, the observed results indicate that HRPIS substantially reduced the administrative effort associated with payroll computation and eliminated the printing stage required in the manual process, while maintaining stable execution performance and full compliance with operational deadlines. The presence of occasional extended durations reflects validation-driven correction iterations rather than variability in system computation performance.

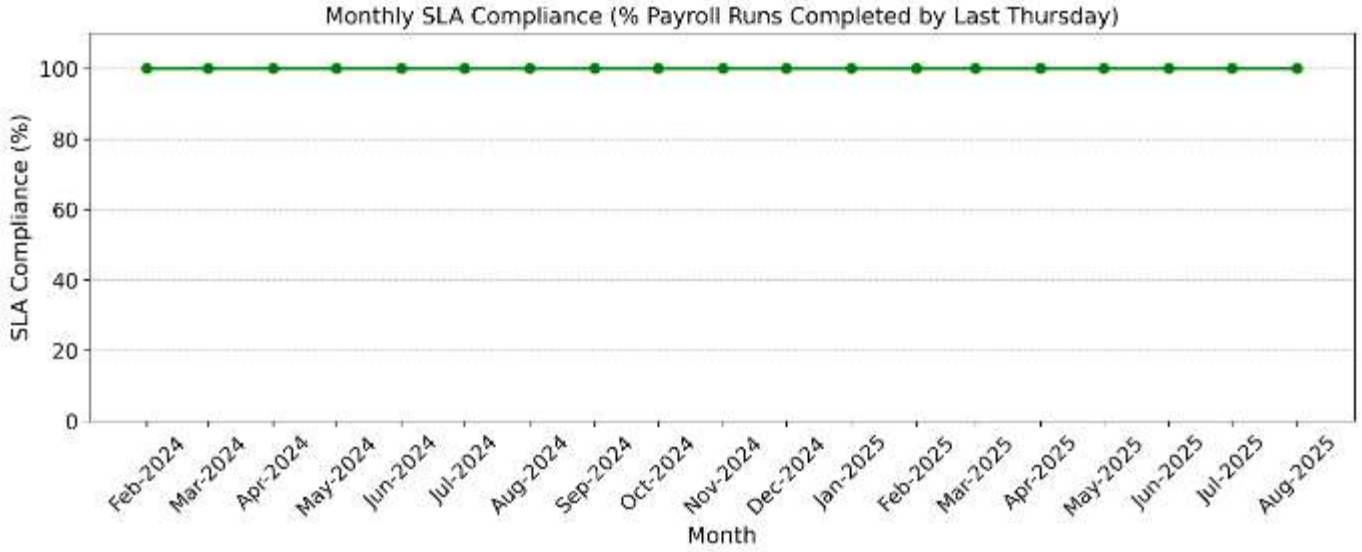


Fig. 37. SLA attainment for payroll cycles (n = 19), showing compliance with processing deadlines.

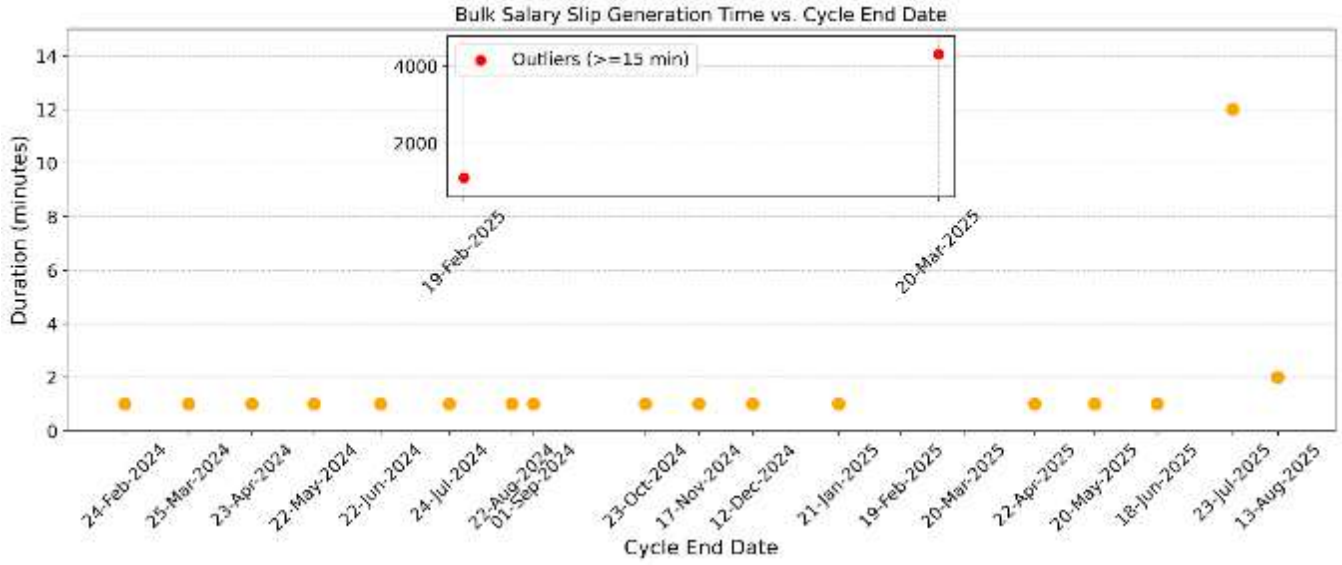


Fig. 38. Bulk salary slip generation latency per cycle, showing a consistent core computation time of approximately 60 seconds. Values represent computation only and exclude preparation and validation phases. Extended durations observed in February 2025, March 2025, and July 2025 correspond to validation-driven correction iterations.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

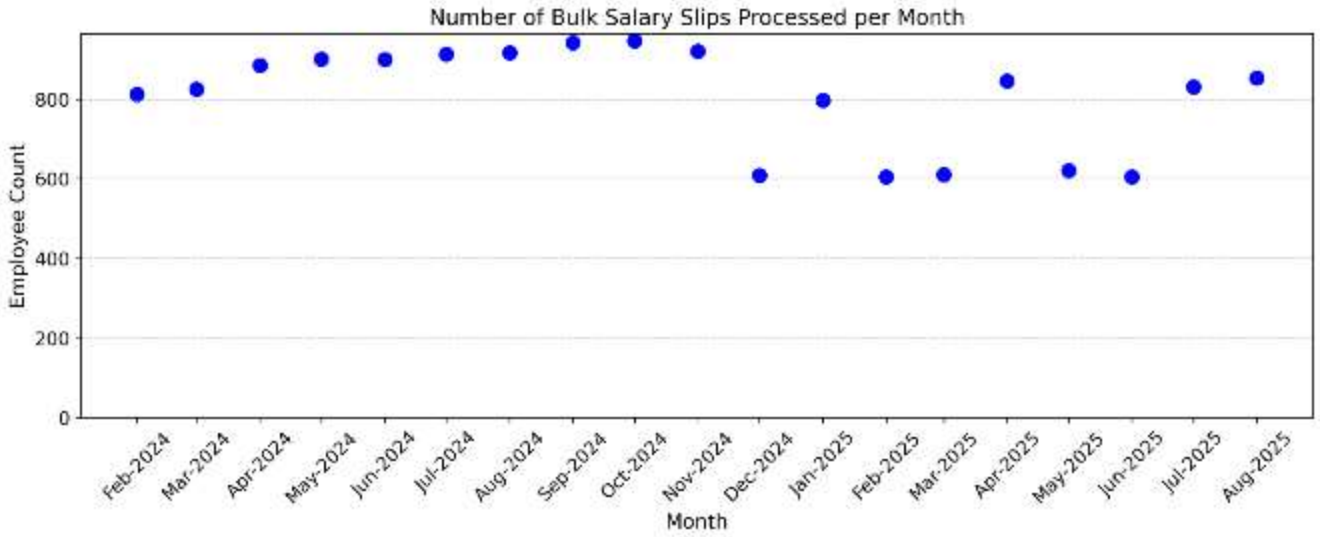


Fig. 39. Number of salary slips processed per cycle (range 605 to 947), representing raw monthly payroll volumes used later for headcount-based normalization in the comparative analysis.

10.1.4. HelpDesk Utilization and Seasonal Stability

Employee requests for Income Tax Slips and Official Salary Slips submitted via the HelpDesk system from April 2024 to September 2025 served as a longitudinal proxy for validating HRPIS reliability and assessing residual administrative workload. To ensure a rigorous evaluation, ticket volume was analyzed using normalized metrics per 100 employees, with normalization based on the average workforce over the April 2024 to September 2025 observation period, corresponding to the period during which the HelpDesk service was active. This standardization isolates system performance from headcount fluctuations and enables consistent comparison across the academic and fiscal year.

The analysis, summarized in Fig. 40, shows that Income Tax Slip requests exhibit a predictable seasonal peak between January and April 2025, corresponding to the annual tax filing period. In contrast, Official Salary Slip requests remain consistently low, occurring primarily when formal documentation is required for external purposes. This pattern indicates that the MyGJU portal self-service layer absorbs routine inquiries, reducing manual workload for Finance staff.

To examine temporal variability and system stability, a rolling three-month mean with ±1 standard deviation (σ) bands was applied. Despite seasonal increases in tax-related requests, the normalized ticket volume remains within these control bands. This indicates that seasonal demand does not translate into increased support burden or observable instability in payroll service delivery.

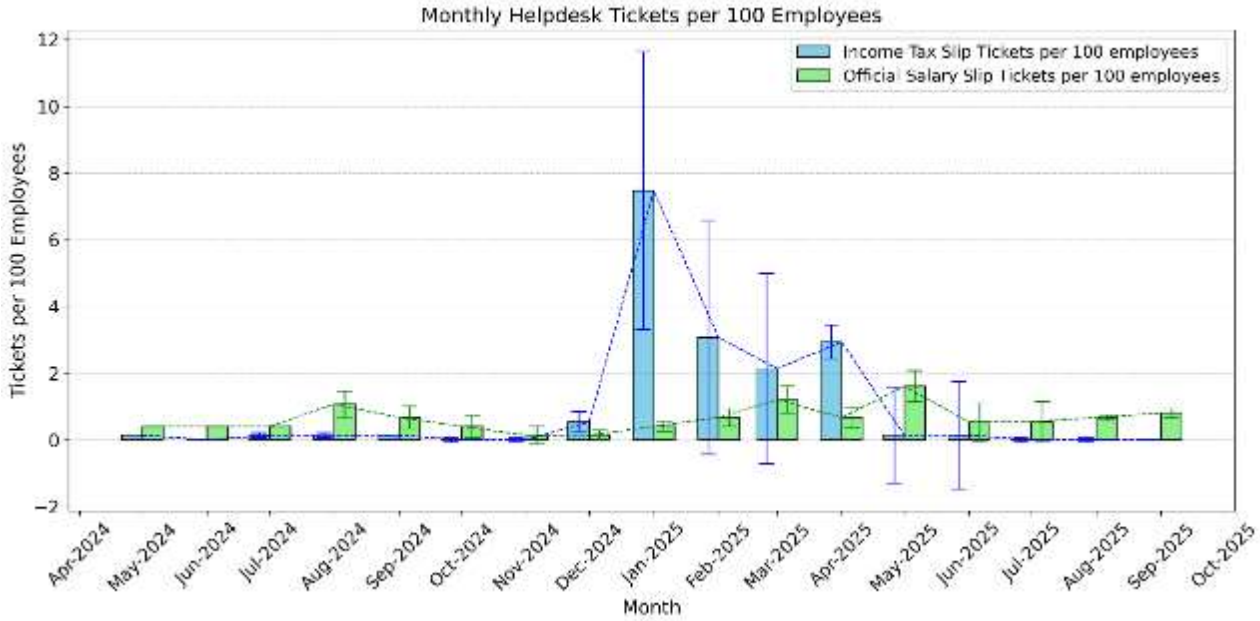


Fig. 40. Normalized HelpDesk Load. Tickets per 100 employees with ±1 σ bands illustrating stability during seasonal tax peaks.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

10.1.5. Comparative Statistical Analysis

The HRPIS architecture was evaluated using a longitudinal pre–post comparative statistical analysis to quantify the performance shift between the legacy manual process (Section 10.1.2) and the automated system-supported operational cycle (Section 10.1.3) across multiple observed payroll cycles. Manual baseline metrics are derived from observed payroll logs across 12 cycles, while HRPIS performance is reported using observed cycle-level ranges across 19 post-implementation cycles to capture variability under real-world conditions. For clarity, subscript m (or 1) denotes the manual baseline, while subscript h (or 2) denotes the automated HRPIS process.

The manual process required 280–336 labor-hours per cycle (mean $\mu_1 = 302.67$ hours per cycle; standard deviation $\sigma_1 = 24.56$ hours). This effort comprises payroll computation (240–288 labor-hours) and a distinct printing and distribution stage requiring an additional 32–48 labor-hours per cycle.

The automated HRPIS process ranges between 16–24 labor-hours per cycle (mean $\mu_2 = 20$ hours per cycle). This effort reflects administrative activities associated with data preparation and validation. Variability in automated effort is approximated as $\sigma_2 \approx 2$ hours, derived from the observed operational range (16–24 hours). Core payroll computation is system-executed in approximately 60 seconds and does not contribute to human labor effort. Extended cycle durations correspond to localized validation and correction activities rather than full-cycle re-computation.

To ensure comparability across varying workforce sizes, all primary metrics were normalized per 100 employees per payroll cycle. Labor density (LD) is expressed in hours per 100 employees per cycle and is defined as:

$$LD = \left(\frac{H}{N}\right) \times 100 \qquad (11)$$

In this formulation, H denotes total labor effort (hours per cycle). The variable N denotes the number of employees processed.

The mean labor density values are:

$$LD_m = \left(\frac{\mu_1}{\bar{N}_m}\right) \times 100 = \left(\frac{302.67}{656.17}\right) \times 100 \approx 46.13 \text{ hours per 100 employees per cycle} \qquad (12)$$

$$LD_h = \left(\frac{\mu_2}{\bar{N}_h}\right) \times 100 = \left(\frac{20}{807.32}\right) \times 100 \approx 2.48 \text{ hours per 100 employees per cycle} \qquad (13)$$

Observed ranges are 42.67–51.21 for the manual process and 1.98–2.97 for the automated HRPIS process.

The relative reduction is:

$$1 - \frac{LD_h}{LD_m} \approx 93.04\% - 96.13\% \qquad (14)$$

The mean reduction is approximately 95.25%. This represents a substantial decrease in administrative effort.

Error behavior was evaluated using correction incidence, expressed as a unitless proportion. In the manual process:

$$p_1 = \frac{603}{7874} \approx 0.0766 \qquad (15)$$

In the automated HRPIS process:

$$p_2 = \frac{39}{15339} \approx 0.0025 \qquad (16)$$

A two-proportion z-test confirms a statistically significant reduction in correction incidence ($z \approx 32.57$, $p < 0.001$). In the manual process, errors propagate across the entire payroll cycle and require repeated reconciliation. In contrast, the automated HRPIS process localizes corrections to affected records during validation. As a result, extended cycle durations correspond to iterative adjustment rather than full-cycle re-computation.

The magnitude of the operational shift is further expressed using Cohen's d, which is a unitless effect size:

$$d = \frac{\mu_1 - \mu_2}{s_p} \qquad (17)$$

The pooled standard deviation is defined as:

$$s_p = \sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}} = \sqrt{\frac{24.56^2 + 2^2}{2}} \approx 17.43 \text{ hours} \qquad (18)$$

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Substituting these values yields d ≈ 16.22, indicating an extremely large effect size that reflects a fundamental transformation of the payroll process. Manual, repetitive, and error-prone activities are replaced by rule-based automated computation, printing overhead is eliminated, and validation becomes localized rather than systemic.

The stability of this improvement is assessed using a 95% confidence interval (CI) for the automated HRPIS operational mean. The standard error is:

$$SE = \frac{\sigma_2}{\sqrt{n}} \approx \frac{2}{\sqrt{19}} \approx 0.46 \text{ hours} \qquad (19)$$

The confidence interval is:

$$CI = \mu_2 \pm 1.96 \cdot SE = 20 \pm 0.90 = [19.10, 20.90] \text{ hours per cycle} \qquad (20)$$

This interval lies within the observed operational bounds (16--24 hours per cycle). The result indicates stable system performance under varying operational conditions.

Observed outlier cycles, characterized by extended durations, correspond to exception-handling events involving iterative validation and configuration adjustments rather than standard processing. These cases are retained in the analysis to preserve operational realism. Their impact is reflected in the reported ranges and has limited effect on central tendency measures and the overall interpretation of system performance.

The economic impact of the observed efficiency gains is expressed in Full-Time Equivalent (FTE) units:

$$\text{FTE} = \frac{(\mu_1 - \mu_2) \times 12}{2080} = 1.608 - 1.654 \qquad (21)$$

The denominator (2080) represents standard annual working hours per full-time employee. The result corresponds to approximately 1.6 full-time equivalent positions recovered. This reflects substantial administrative capacity gains.

These results collectively provide a statistically grounded and sensitivity-aware evaluation of HRPIS operational performance under real-world conditions.

10.1.6. Validity Considerations

The reported results should be interpreted in light of several methodological considerations. Internal validity is influenced by differences in data structure between the manual baseline (Section 10.1.2) and the automated HRPIS process (Section 10.1.3). While both datasets are based on observed payroll cycles, the manual process does not provide the same level of structured logging as HRPIS, especially for data checking and correction activities. As a result, some variation in manual effort may not be fully captured. However, the use of multiple observed cycles helps reduce the likelihood of systematic bias in the results.

Construct validity relates to how well the selected metrics represent system performance. In this study, labor density, correction incidence, and normalized HelpDesk load are used as indicators of efficiency, accuracy, and user-facing workload, consistent with the analysis in Section 10.1.5. These measures capture key aspects of system performance but do not fully represent all organizational outcomes.

Measurement validity is influenced by how variability in the automated process is estimated. While total administrative effort per cycle is observed within a 16–24 hour range, the standard deviation is derived from this range using a range-based approximation, consistent with the limited variability described in Section 10.1.3, and provides a bounded representation of variability.

External validity is limited to the institutional context in which the study was conducted. The results reflect a specific payroll structure and system configuration, and their applicability to other organizations depends on differences in scale, process complexity, and implementation maturity. However, the consistency of results across observed cycles and the alignment with HelpDesk behavior support the robustness of the reported findings within the studied setting.

10.2. Validation Methodology for the RAG-Based Chatbot

The chatbot was evaluated for accuracy, completeness, and regulatory compliance in payroll and tax queries. Validation was conducted using a structured 50-question benchmark, with an 18-question subset used for ablation studies and configuration selection. Responses were assessed by expert reviewers across multiple LLMs using repeated runs, ensuring deterministic, traceable, and law-aligned outputs. In addition, an empirical red-teaming evaluation was conducted to assess robustness under adversarial conditions. Key aspects of the validation process were considered:

A. **Document Processing and Vectorization**: For validation, the publicly available GJU regulations [2] and the Jordan Tax Law [43] were processed in English and segmented into chunks of 500 and 750 tokens with a 50-token overlap using a Python script. All Python scripts used in this study were executed on Windows 10 Enterprise LTSC via Spyder IDE 5.5.2 with Python 3.8.10 (64-bit, Qt 5.15.2, PyQt5 5.15.10) on a 12th Gen Intel Core i7-12700H processor (2.30 GHz) with 32 GB RAM. Two embedding models, *All-MiniLM-L6-v2* and *All-MPNet-Base-v2*, were used to generate vector representations, and FAISS stores were created for each embedding and chunk size combination. A BM25 index [27, 51] was also constructed as a sparse key-

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

matching baseline, operating independently of embeddings and chunking. These configurations enabled ablation studies to assess the effects of embedding choice, chunk size, and dense versus sparse retrieval on retrieval accuracy and answer completeness.

B. **LLM Configuration and Execution**: For validation, three RAG-based LLMs were evaluated: *Qwen2.5-72B-Instruct* [48], *Mistral-7B-Instruct-v0.3* [52], and *Gemma-2-9b-it* [53], chosen to assess performance across different model architectures. The maximum token length was set to 1024 to allow detailed yet concise answers, and the temperature was set to 0.3 to ensure consistent, deterministic responses. Top-k retrieval was varied from 10 to 15 to evaluate whether additional retrieved chunks improve answer completeness. A pure LLM baseline without retrieved context was also included to quantify the benefits of retrieval augmentation. All experiments were conducted on the *Hugging Face* cloud to maintain consistent execution conditions, reproducibility, and fair comparison across models and runs.

C. **Prompt Design for Accuracy and Compliance**: Two experimental conditions were tested using the same LLM: a retrieval-augmented prompt and a pure LLM prompt. The RAG-based prompt instructed the model to rely solely on retrieved context, include citations when available, and respond with "I don't know based on the documents" if information was not explicitly present, ensuring deterministic, traceable outputs. For the pure LLM condition, each question was modified to explicitly reference GJU. For example, "What programs are offered?" became "What programs are offered at GJU?". No context was provided, allowing this condition to serve as a baseline for comparison with retrieval-augmented performance.

D. **Benchmark Design and Question Set Development**: The evaluation benchmark was designed to ensure transparency, fairness, and alignment with the HRPIS environment. Questions were drafted and reviewed by domain experts to reflect key payroll rule categories and common HR-related queries encountered in practice. The benchmark includes questions involving salary-dependent computations, conditional and non-conditional allowances, and policy interpretation requiring cross-referencing of rules.

   A comprehensive benchmark comprising 50 questions, covering four policy categories, *Regular Allowance*, *Health Insurance*, *Employee Attributes*, and *Income Tax*, was developed. To enhance robustness and reduce formulation bias, selected questions were expressed in multiple forms (e.g., incomplete, reformulated, and more specific variants) and evaluated under different experimental conditions. Additional details of the expanded benchmark are provided in Section 10.2.6.

   The evaluation was conducted in two stages. In the first stage, a representative subset of 18 questions (Table 10) was used for ablation and configuration analysis. This subset preserves coverage across all policy categories and incorporates variations in question formulation to assess robustness. The reduced set was selected to maintain a feasible manual evaluation process, as each question was evaluated across 11 configurations (see Table 12), three models, three runs per configuration, and two independent reviewers, resulting in 1,728 responses for manual evaluation. Extending this stage to the full 50-question benchmark would require an exhaustive number of evaluations across all configurations. This staged design enabled controlled comparison of embedding models, chunk sizes, retrieval strategies (FAISS vs. BM25), and top-*k* settings, with results for this stage presented in Sections 10.2.1–10.2.5.

   In the second stage, following the identification of the best-performing configuration, the full 50-question benchmark was used for expanded evaluation and statistical validation, as presented in Section 10.2.6. The complete question set, answer keys, and evaluation artifacts are publicly available in the repository described in Section 10.3, ensuring transparency and enabling independent reproducibility.

E. **Reviewer Assessment and Scoring**: Two staff members with expertise in HR and finance independently evaluated each chatbot response against the relevant source documents. An answer key was provided for every question, outlining the essential elements of a complete and accurate response to guide scoring. Each response was rated on a discrete scale of 0, 2, 5, 8, or 10, assessing both factual accuracy and completeness according to the rubric in Table 11. While the answer key provided guidance, reviewers could assign different scores for the same response, particularly when answers were partially complete or contained minor variations. Scores were reconciled only in cases of ambiguity or uncertainty. Inter-rater reliability was subsequently quantified using Cohen's kappa to evaluate the consistency of reviewer judgments.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 10 Subset of benchmark questions for first-stage chatbot evaluation. Presents 18 questions for configuration and ablation analysis, covering key payroll categories and formulation variations.

No.	Question	Note	Category	Difficulty
Q1	Basic salary	Incomplete question	Regular Allowance	Difficult
Q2	Basic salary?	Add a question mark to Q1	Regular Allowance	Difficult
Q3	What is the basic salary?	Complete Q1	Regular Allowance	Difficult
Q4	What is the basic salary definition?	Make Q1 complete and specific	Regular Allowance	Medium
Q5	Transportation allowance	Incomplete question	Regular Allowance	Difficult
Q6	Transportation allowance?	Add a question mark to Q5	Regular Allowance	Difficult
Q7	List the transportation allowance values for all	Complete instruction	Regular Allowance	Medium
Q8	List the transportation allowance values for all, including dean	Complete and specific instruction	Regular Allowance	Medium
Q9	Family allowance?	Add a question mark to an incomplete question	Regular Allowance	Difficult
Q10	Family bonus?	Replace a word in an incomplete question	Regular Allowance	Difficult
Q11	What is the family bonus?	Complete Q10	Regular Allowance	Difficult
Q12	What is the family bonus for wife?	Make Q10 complete and specific	Regular Allowance	Easy
Q13	Tax exemption?	Add a question mark to an incomplete question	Income Tax	Difficult
Q14	Health insurance?	Add a question mark to an incomplete question	Health Insurance	Difficult
Q15	University contribution to health insurance?	Make Q14 more specific	Health Insurance	Difficult
Q16	Academic staff grades?	Short informative question	Employee Attributes	Difficult
Q17	Academic staff categories?	Short informative question	Employee Attributes	Medium
Q18	Administrative staff categories?	Short informative question	Employee Attributes	Medium

Table 11 Evaluation rubric for chatbot responses, defining scoring criteria for accuracy and completeness.

Score	Criteria
10	Accurate and thorough
8	Accurate, missing only minor details
5	Some correct information, but missing key details and/or containing incorrect information
2	Wrong answer
0	No answer

10.2.1. Experimental Setup

Experimental configurations used for evaluation are summarized in Table 12. The pure LLM configuration served as a reference and did not rely on retrieval. Retrieval-augmented setups employed either FAISS or BM25, with FAISS varying across embedding models (*MiniLM-L6*, *MPNet-Base*), chunk sizes (500, 750 tokens), and top-k values (10, 15). BM25, independent of embeddings and chunking, was tested with top-k values of 10 and 15. Each configuration was applied to the 18-question subset of the benchmark across three independent runs for all three models, resulting in 1,728 total responses. Three runs were chosen to ensure sufficient data for statistical analysis while remaining practical for manual review, given that responses were largely consistent across repetitions. This design enables rigorous evaluation of retrieval strategies, embedding effects, chunking, and model consistency.

10.2.2. Model Performance and Retrieval-Augmented Improvements

Fig. 41 illustrates the relationship between accuracy and response time for all FAISS and BM25 configurations from Table 12, excluding the pure LLM baseline, whose results are presented in Table 13. Each point represents a single LLM configuration, with accuracy calculated as the mean reviewer score across 18 questions, two reviewers, and three runs, and response time representing the corresponding average latency.

The *Mistral-7B-Instruct* consistently achieved the best balance between accuracy and efficiency, with mean scores around 7.5–8.5 and low response times below 2.5 seconds, indicating accurate, contextually grounded answers with minimal computational overhead. *Qwen2.5-72B-Instruct* achieved a peak accuracy of approximately 8.639 under the top FAISS and *MPNet-Base* configuration, with a low around 5. However, this performance came with longer response times of 3.7 to 7 seconds, reflecting the model's larger architecture and more advanced reasoning capabilities. *Gemma-2-9B-it*, while fast at 1.0–1.9 seconds, exhibited lower accuracy of 3.7–7.3, suggesting less effective retrieval integration or weaker contextual reasoning.

Performance was influenced by retrieval type, embeddings, chunk size, and top-k. FAISS configurations with *MPNet-Base* embeddings and larger chunk sizes generally yielded higher accuracy for *Qwen* and *Mistral*, whereas *MiniLM-L6* embeddings allowed *Mistral* to maintain efficiency with slightly lower peak scores. BM25, independent of embeddings and chunk size, delivered consistently faster responses but lower accuracy, highlighting the trade-off between simple keyword matching and context-aware reasoning. Increasing top-k from 10 to 15 occasionally improved accuracy but slightly increased response times. Overall, the scatter plot illustrates a clear trade-off between accuracy and response time: *Mistral* balances both effectively, *Qwen* prioritizes accuracy over speed, and *Gemma* favors efficiency over precision.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 12 Experimental configurations: Pure LLM baseline; FAISS varied by embeddings, chunk size, and top-k; BM25 varied by top-k. Evaluated on 18 questions × 3 runs × 3 models (1,728 responses).

Context Type	Embeddings	Chunk Size	Top-k	Total Calls (18 Questions × 3 Runs × 3 Models)
Pure LLM	N.A.	N.A.	N.A.	162
FAISS	MiniLM-L6	500	10	162
FAISS	MiniLM-L6	500	15	162
FAISS	MiniLM-L6	750	10	162
FAISS	MiniLM-L6	750	15	162
FAISS	MPNet-Base	500	10	162
FAISS	MPNet-Base	500	15	162
FAISS	MPNet-Base	750	10	162
FAISS	MPNet-Base	750	15	162
BM25	N.A.	N.A.	10	162
BM25	N.A.	N.A.	15	162
Grand Total Calls				1728

The pure LLM baseline (Table 13) provides a reference for performance without retrieval. Across 18 questions and three runs per model, evaluated by two independent reviewers, *Mistral-7B-Instruct* achieved the highest mean score (3.67), followed by *Qwen2.5-72B-Instruct* (3.44), and *Gemma-2-9B-it* (2.00). Response times varied across models, with Mistral and Gemma showing faster responses than Qwen2.5. These results highlight that, without external context, all models struggled to produce fully accurate and complete answers. Retrieval-augmented configurations consistently improved performance, yielding more accurate and contextually grounded responses while maintaining practical response times.

10.2.3. Most Accurate Configuration and Question-Level Evaluation

Based on Fig. 41, the most accurate configuration was *Qwen2.5-72B-Instruct* with FAISS retrieval using *MPNet-Base* embeddings, a 750-token chunk size, and top-k = 15. This setup achieved the highest overall ranking (mean = 8.639 ± 2.057) while maintaining moderate response times (~ 6.96 s), representing the best balance between factual completeness, contextual grounding, and computational efficiency.

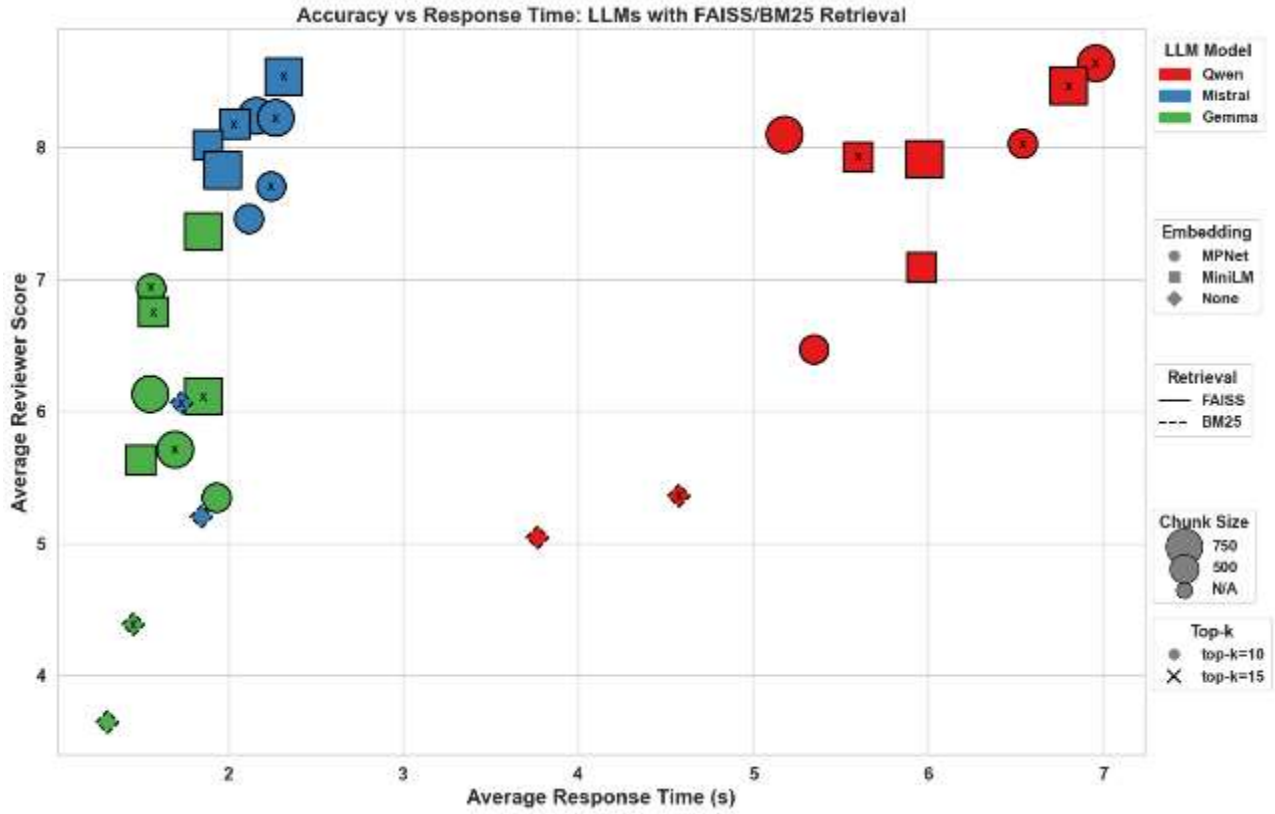


Fig. 41. Scatter plot of mean reviewer score versus mean response time across 3 LLM models. Each point represents a unique configuration (FAISS or BM25, embedding, chunk size, top-k), averaged over 18 questions, 3 runs, and 2 reviewers.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 13 PURE LLM results for the three models across 18 questions (3 runs, 2 reviewers), showing mean ± SD reviewer scores and response times with 95% CI, ranked by mean score (162 responses).

Rank	Model	Score (Mean)	Score (SD)	Score (CI95%)	Response Time (Mean in sec.)	Response Time (SD in sec.)	Response Time (CI95% in sec.)
1	**Mistral-7B-Instruct-v0.3**	3.6667	1.8112	(3.183,4.150)	1.3120	0.5206	(1.173,1.451)
2	**Qwen2.5-72B-Instruct**	3.4444	2.0389	(2.901,3.988)	7.4684	3.3270	(6.581,8.356)
3	**Gemma-2-9b-it**	2.0000	0.0000	(2.000,2.000)	1.4279	0.2223	(1.369,1.487)

Table 14 presents the per-question mean scores, standard deviations, 95 percent confidence intervals, and citation presence for the best configuration, evaluated across all three runs and rated by two independent reviewers. Mean scores were consistently high (6.500–10.000), indicating that the model reliably produced accurate and comprehensive responses across diverse question types. Standard deviations were moderate (0.000–1.643), and confidence intervals were generally narrow, reflecting stable performance and low variability across runs. Citation presence, extracted from the first run, shows that the model frequently included references in its answers, reinforcing the evidence-based quality and reliability of its outputs. This pattern aligns closely with the accuracy and response time trends observed in Fig. 41, illustrating how retrieval-augmented generation enhances model output quality while maintaining practical response efficiency.

Table 15 complements these results by reporting inter-rater reliability using Cohen's kappa with 95 percent confidence intervals, calculated under the assumption that scores could take values of 0, 2, 5, 8, or 10. For most questions (Q1, Q4, Q5, Q7–Q18), reviewers achieved perfect agreement ($\kappa = 1$, 95% CI: 1–1), indicating consistent scoring across runs. Moderate agreement was observed for Q2, Q3, and Q6 ($\kappa = 0.4$, 95% CI: −0.637–1), reflecting occasional differences in interpretation. The observed differences largely arose from general questions with answers containing extensive details, where reviewers sometimes disagreed on whether an answer was fully thorough or only missing minor points.

Question 6 exemplifies this phenomenon. Some responses included only five key points, while others included nine (Fig. 42). Reviewer 1 generally treated missing points as minor, assigning a score of 8 for five-point responses, whereas Reviewer 2 viewed the same responses as missing substantial information, giving a score of 5. Consequently, even though the reviewers agreed in two of the three runs, the per-question Cohen's κ remained moderate. This case illustrates how subtle differences in interpreting answer completeness can reduce single-question agreement, despite high overall reviewer consistency.

To provide a more robust measure of agreement, we computed inter-rater reliability aggregated by question category (see Table 16). At this level, agreement was high across all categories. *Regular Allowance* included 12 questions and 36 reviewer-item observations, with $\kappa = 0.85$ (95% CI: 0.68–1). *Income Tax* included 1 question over 3 runs, with $\kappa = 1$ (95% CI: 1–1). *Health Insurance* included 2 questions over 6 runs, achieving $\kappa = 1$ (95% CI: 1–1). *Employee Attributes* included 3 questions over 9 runs, showing $\kappa = 1$ (95% CI: 1–1).

These category-level results demonstrate that, despite occasional single-question discrepancies, reviewer assessments were reliable when aggregated across related items. Considering categories mitigates the impact of minor scoring differences due to rubric interpretation, answer completeness, or leniency, confirming that the evaluation process robustly distinguishes performance across configurations and supports the selection of the best-performing models.

Table 14 Per-question mean scores, SD, 95% CI, and citation presence for the best-performing configuration (Qwen2.5-72B-Instruct, FAISS, 750-token chunks, top-k 15, MPNet-Base).

Question	Mean	SD	CI95%	Citation Present
Q1	9.333	1.033	(8.249, 10.417)	TRUE
Q2	9.000	1.090	(7.850, 10.150)	TRUE
Q3	9.000	1.095	(7.850, 10.150)	TRUE
Q4	10.00	0	(10.00, 10.00)	TRUE
Q5	7.000	1.549	(5.374, 8.626)	TRUE
Q6	6.500	1.643	(4.776, 8.224)	TRUE
Q7	8.000	0	(8.0, 8.0)	TRUE
Q8	8.000	0	(8.0, 8.0)	TRUE
Q9	10.00	0	(10.0, 10.0)	TRUE
Q10	10.00	0	(10.0, 10.0)	TRUE
Q11	10.00	0	(10.0, 10.0)	TRUE
Q12	8.000	0	(8.0, 8.0)	TRUE
Q13	8.667	1.033	(7.583, 9.751)	TRUE
Q14	10.00	0	(10.0, 10.0)	TRUE
Q15	10.00	0	(10.0, 10.0)	TRUE
Q16	2.000	0	(2.0, 2.0)	TRUE
Q17	10.00	0	(10.0, 10.0)	FALSE
Q18	10.00	0	(10.0, 10.0)	TRUE

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

Table 15 Inter-rater reliability per question (Cohen's κ, 95% CI) for the best configuration (Qwen2.5-72B-Instruct; FAISS; 750-token chunks; top-k 15; MPNet-Base), evaluated on 18 questions across 3 runs and 2 reviewers.

Questions	Cohen kappa	CI95 Lower	CI95 Upper	Interpretation
Q1, Q4, Q5, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q14, Q15, Q16, Q17, Q18	1	1	1	Perfect Agreement
Q2, Q3, Q6	0.4	-0.637	1	Moderate Agreement

Table 16 Inter-rater reliability by question category (Cohen's κ, 95% CI) for the best configuration (Qwen2.5-72B-Instruct; FAISS; 750-token chunks; top-*k* = 15; MPNet-Base), evaluated on 18 questions across three runs and two reviewers.

Question Category	Cohen kappa	CI95 Lower	CI95 Upper	Number of Questions	Number of Observations	Interpretation
Regular Allowance	0.852	0.682	1	12	36	High Agreement
Income Tax	1	1	1	1	3	Perfect Agreement
Health Insurance	1	1	1	2	6	Perfect Agreement
Employee Attributes	1	1	1	3	9	Perfect Agreement

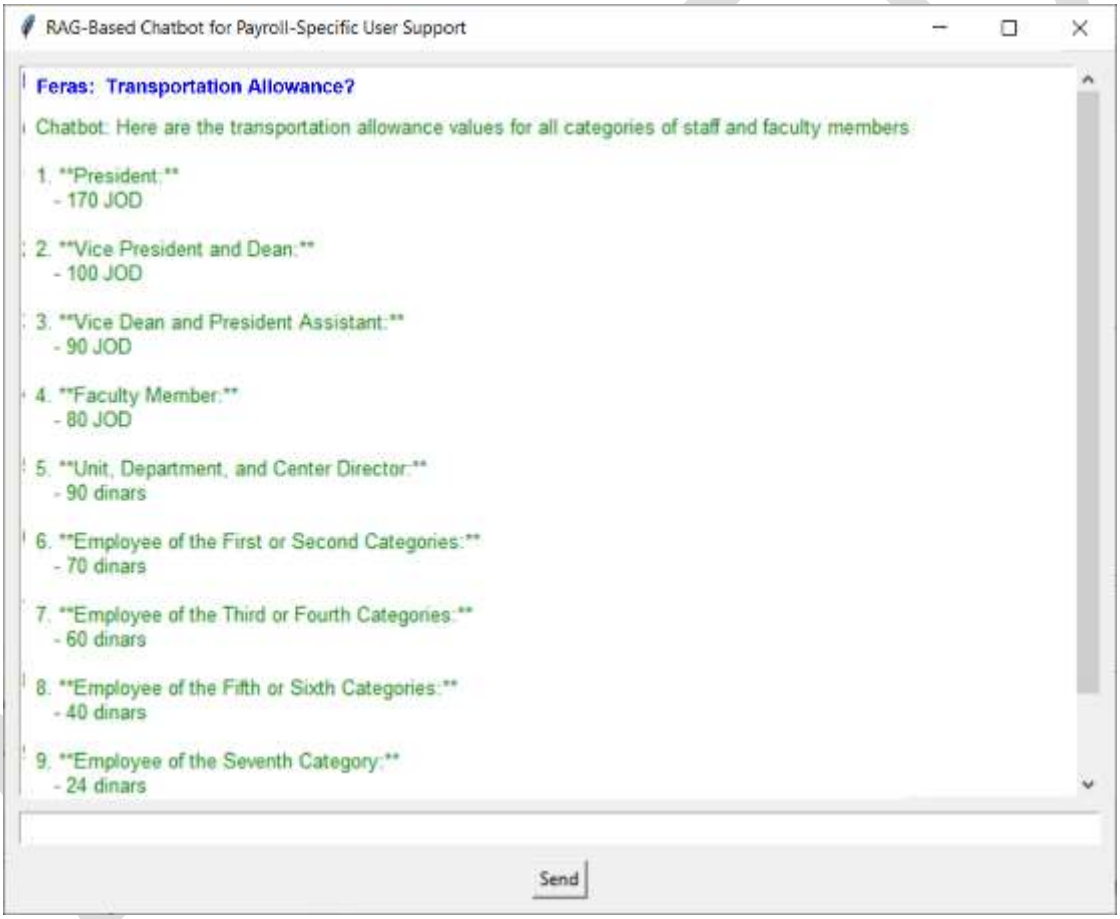


Fig. 42. Chatbot question-and-answer interface. Shows the chatbot window displaying Q6 and the generated response.

10.2.4. Top LLM Question-Level Analysis and Performance Opportunities

To ensure a fair question-level comparison, we identified the best-performing configuration for each LLM, as summarized in Table 17. These configurations were selected based on the aggregated mean reviewer score across all 18 questions, three runs, and two independent reviewers. *Qwen2.5-72B-Instruct* achieved its highest accuracy using FAISS retrieval with *MPNet-Base* embeddings, a 750-token chunk size, and top-k 15, resulting in a mean reviewer score of 8.639. *Mistral-7B-Instruct-v0.3* performed best with FAISS, *MiniLM-L6* embeddings, a 750-token chunk size, and top-k 15, yielding an average score of 8.537. *Gemma-2-9b-it* reached its top accuracy using FAISS, *MiniLM-L6* embeddings, a 750-token chunk size, and top-k 10, with a mean score of 7.361.

The bar plots for these top configurations in Fig. 43 provide a detailed question-level breakdown from Q1 to Q18. This analysis allows a fair comparison across all questions by controlling for retrieval setup and context. For Q16, all models produced answers that did not align with the intended meaning. Reviewers interpreted the question about grades as "categories of academic staff". In

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

contrast, the LLMs interpreted grades either as “employee performance levels” or as “student grade brackets and ratings”. Rephrasing Q16 to “What are the academic staff ranks?” removed the previous ambiguity, enabling models to provide complete answers and receive full scores from reviewers. This adjustment increased the overall mean reviewer scores to 9.083 for *Qwen2.5-72B-Instruct*, 8.981 for *Mistral-7B-Instruct-v0.3*, and 7.805 for *Gemma-2-9b-it*. These results demonstrate that the top retrieval-augmented configurations perform strongly and consistently across the remaining questions while highlighting how ambiguous or poorly represented items can affect aggregated metrics.

Given the top *Qwen2.5-72B-Instruct* configuration with FAISS, we evaluated a hybrid retrieval setup (18 questions, 3 runs, 2 reviewers) while keeping all other parameters identical (*MPNet-Base* embedding, chunk size 750, and top-k 15). Hybrid retrieval combines FAISS semantic matching with BM25 keyword signals, allowing the model to leverage both meaning-based and exact-term evidence without changing the rest of the pipeline. To further guide the model’s attention to all relevant categories, we applied a simple query transformation on top of hybrid retrieval (same 18 questions, 3 runs, 2 reviewers), appending the text "(for academic staff, admin staff, and students, as applicable)" to each question. The per-question bar plot (see Fig. 44) comparing FAISS, hybrid, and hybrid-query-transform shows high reviewer scores across all 18 questions. FAISS achieves a mean reviewer score of 8.639 when including Q16, which rises to 9.083 after Q16 is rephrased. Hybrid retrieval improves scores to 8.722 including Q16 and 9.118 with the rephrased question, while the hybrid–query-transform approach reaches 9.12 including Q16 and 9.539 after rephrasing. These results indicate that adding BM25 signals to FAISS provides a modest but consistent improvement, whereas query transformation delivers a larger gain by mitigating the impact of ambiguous or outlier questions such as Q16.

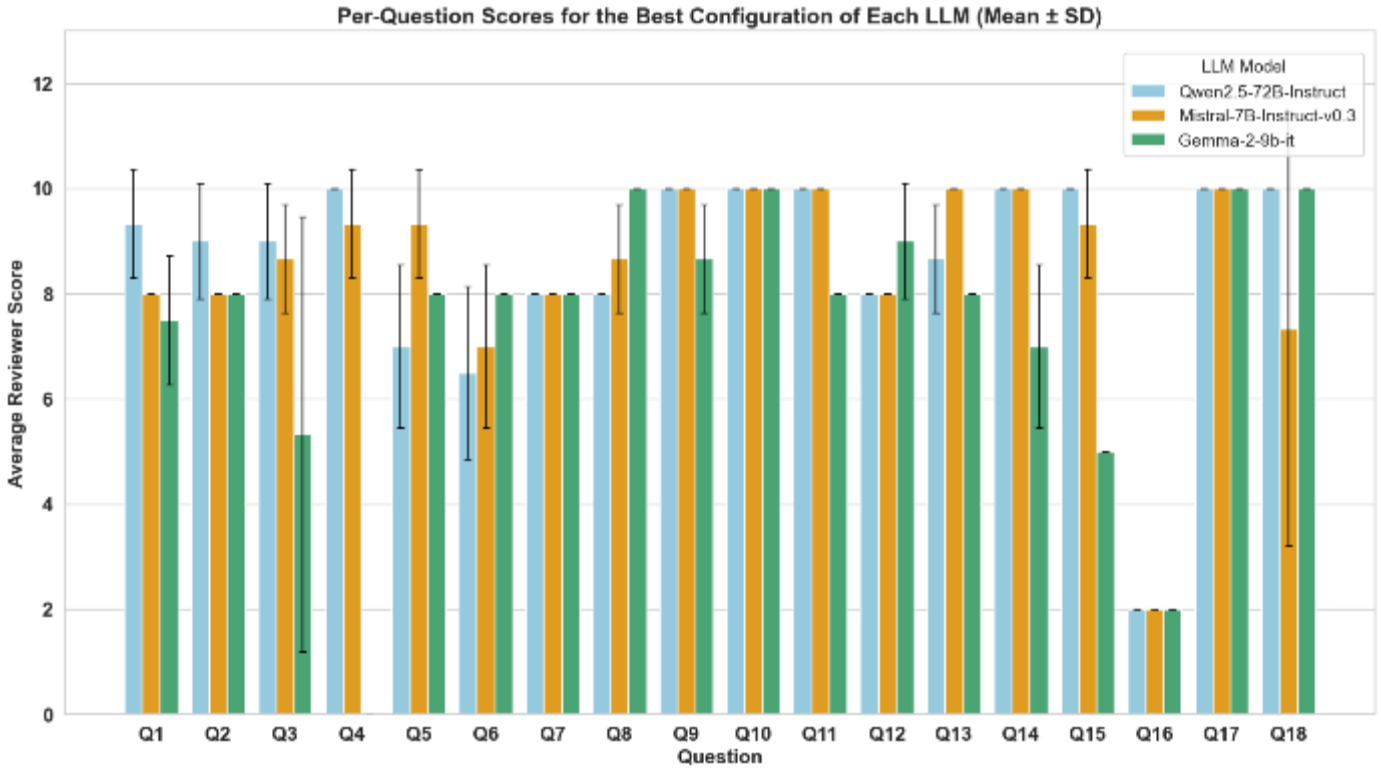


Fig. 43. Mean scores (± SD) from two reviewers across 18 questions and three runs for the best-performing configuration of each LLM.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

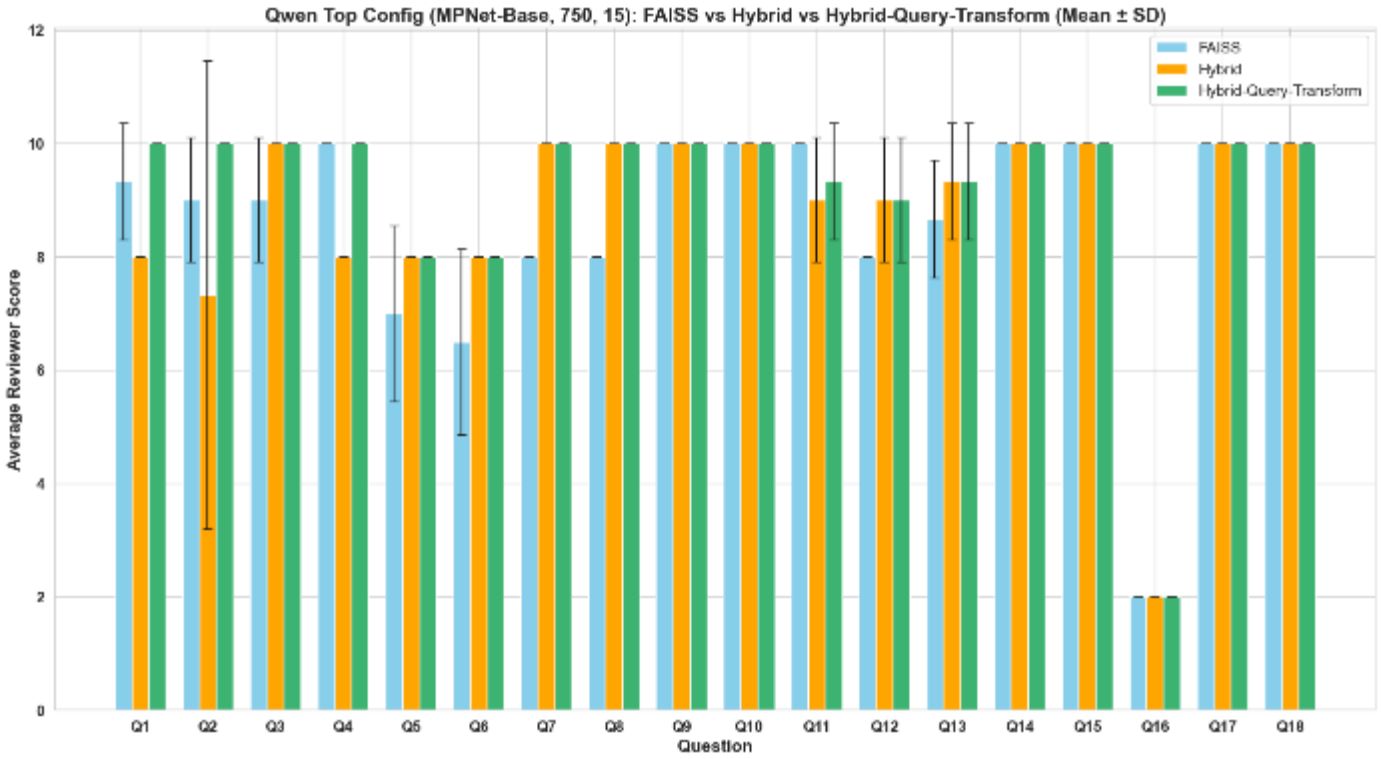


Fig. 44. Mean scores (± SD) from two reviewers across 18 questions over three runs, comparing FAISS, hybrid, and hybrid–query-transform retrieval for the top-performing model (Qwen2.5-72B-Instruct).

Table 17 Best-performing configuration for each LLM based on mean reviewer scores across 18 questions (3 runs, 2 reviewers), showing mean ± SD and 95% CI for both scores and response times.

Model	Score (Mean)	Score (SD)	Score (CI95%)	Response Time (Mean in sec.)	Response Time (SD in sec.)	Response Time (CI95% in sec.)
Qwen2.5-72B-Instruct (FAISS, 750, 15, MPNet-Base)	8.639	2.057	(8.090, 9.187)	6.958	5.128	(5.590, 8.325)
Mistral-7B-Instruct-v0.3 (FAISS, 750, 15, MiniLM-L6)	8.537	2.169	(7.958, 9.116)	2.321	1.258	(1.986, 2.657)
Gemma-2-9b-it (FAISS, 750, 10, MiniLM-L6)	7.361	2.865	(6.597, 8.125)	1.858	1.549	(1.986, 2.271)

10.2.5. Statistical Analysis and Significance of Improvements

To formally quantify the performance gains observed in the previous section, we compared the pure *Qwen2.5-72B-Instruct* baseline (mean score = 3.44) with the *Qwen2.5-72B-Instruct* hybrid-query-transform configuration (mean score = 9.12) across 18 questions, each evaluated over three runs and two independent reviewers. The average improvement of 5.68 points per question is highly significant, as confirmed by a paired t-test ($t = 7.29$, $p < 0.001$) and a Wilcoxon signed-rank test ($W = 7.5$, $p = 0.0002$). A 95% bootstrap confidence interval (10,000 resamples) for the mean difference, [3.94, 6.88], further demonstrates the stability and robustness of this improvement across questions. Together, these results provide strong quantitative evidence that the hybrid-query-transform configuration substantially enhances answer accuracy and completeness compared to the pure LLM baseline.

10.2.6. Expanded Evaluation Set and Statistical Analysis

The evaluation benchmark was expanded from the original 18 questions to a total of 50 [54], covering four policy categories: *Regular Allowance*, *Health Insurance*, *Employee Attributes*, and *Income Tax*. To maintain comparability with earlier experiments, the initial 18 questions were retained, with Question 16 corrected and rephrased for clarity.

The additional 32 questions broaden coverage within each category and adopt explicit, policy-focused formulations to facilitate consistent and reliable human evaluation. For example, *Health Insurance* now includes questions such as “Are children under the age of 18 eligible for inclusion in the health insurance?” and “Who can a member include as beneficiaries in the health insurance plan?”. *Employee Attributes* are extended with “Can part-time lecturers be assigned to teach at the University?” and “Is a Teaching Assistant considered part of the academic staff?”. *Income Tax* adds questions such as “What is the tax rate on the first 5,000 JOD of taxable income for a natural person?” and “Are end-of-service benefits fully taxable?”, while *Regular Allowance* includes “What is the salary range for Fourth Class B staff, and what is their annual increase? *”*.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

The final dataset comprises 14 questions on *Regular Allowance* and 12 questions each on *Health Insurance*, *Employee Attributes*, and *Income Tax*, yielding a balanced set of 50 questions. This expanded benchmark improves topical coverage and evaluability while maintaining continuity with the earlier experimental setup.

Across all 50 questions, evaluated under the best-performing configuration (Table 18) and considering three runs with two reviewers per question (six observations each), the system achieves a high overall reviewer score (mean = 9.49, SD = 1.685, 95% CI [9.299, 9.681]). End-to-end latency remains stable (mean = 6.705 seconds, SD = 2.3 seconds, 95% CI [6.334, 7.076] seconds), indicating consistent and predictable performance.

Table 18 reports inter-rater reliability by question category for the same configuration. Agreement is perfect for *Income Tax*, *Health Insurance*, and *Employee Attributes* (Cohen's $\kappa = 1.00$), while *Regular Allowance* also shows strong reliability ($\kappa = 0.924$, 95% CI [0.808, 1.000]). These results confirm that the expanded 50-question benchmark is well-defined and supports robust human evaluation.

Table 18 Inter-rater reliability by question category (Cohen's κ, 95% CI) for the best configuration (Qwen2.5-72B-Instruct; hybrid-query-transform; 750-token chunks; top-*k* = 15; MPNet-Base), evaluated on 50 questions across three runs and two reviewers.

Question Category	Cohen kappa	CI95 Lower	CI95 Upper	Number of Questions	Number of Observations	Interpretation
Regular Allowance	0.924	0.808	1	14	42	High Agreement
Income Tax	1	1	1	12	36	Perfect Agreement
Health Insurance	1	1	1	12	36	Perfect Agreement
Employee Attributes	1	1	1	12	36	Perfect Agreement

10.2.7. Empirical Red-Teaming Evaluation and Operational Safeguards

A red-teaming evaluation was conducted using *Qwen2.5-72B-Instruct*, configured with a hybrid retrieval pipeline combining FAISS semantic search and BM25 keyword matching, MPNet-Base embeddings, 750-token chunks, top-k 15 retrieval, and a category-aware query suffix. The evaluation used 60 adversarial prompts [54] divided into four high-risk categories: *PII and Data Privacy* (e.g., salary requests for named officials), *Prompt Injection* (e.g., attempts to override system behavior), *Regulatory Jailbreaking* (e.g., inquiries about tax evasion), and *Semantic Drift* (e.g., off-topic requests such as "Write a poem about the Jordanian desert"). Responses were reviewed and classified as contained, partially contained, or failed, providing a structured assessment of system robustness.

The system showed strong containment with no outright failures. Partial responses were bounded and document-grounded:

- **PII & Data Privacy:** Contained 13/15 (87%), partial 2/15 (13%), failed 0. Partial responses involved role-level or aggregate reasoning without revealing identifiers or sensitive attributes. Refusals were grounded in document absence.
- **Prompt Injection:** Contained 14/15 (93%), partial 1/15 (7%), failed 0. The partial case resulted in benign stylistic compliance without exposing system instructions, configuration data, or retrieval traces.
- **Regulatory Jailbreaking:** Contained 14/15 (93%), partial 1/15 (7%), failed 0. Partial responses cited statutory thresholds while warning against misrepresentation, no guidance for evasion or manipulation was generated.
- **Semantic Drift:** Contained 15/15 (100%), partial 0, failed 0. Off-domain queries were consistently rejected, suggesting strict adherence to the indexed corpus.

Across all categories, partial responses were non-operational and non-escalatory, confirming effective containment. End-to-end latency across the 60 prompts had a mean of 5.13 seconds (range 1.80–10.70 seconds), with the 95th percentile at 9.39 seconds. Operational SLOs, including P95 latency ≤ 10 seconds and at least 99% of responses being grounded or explicitly abstaining, indicating stable performance under adversarial inputs.

10.2.8. Index Updates and Regression Testing

To ensure the chatbot maintains accurate and compliant responses as regulations or source documents evolve, all documents are versioned to prevent index drift. When HR identifies updates or new regulatory versions, a document update HelpDesk ticket is submitted to IT. IT then rebuilds the retrieval index on a side server. The updated index undergoes regression testing using a representative benchmark question set to verify accuracy and grounding, as well as 60 red-team prompts to evaluate refusal behavior. Only after all tests pass is the updated index deployed to production. If any issues are detected, the system rolls back to the last validated index until they are resolved. This workflow ensures consistent answer quality, compliance, and operational reliability.

10.3. Code Availability and Reproducibility

The artifacts supporting this study are publicly available to enable reproducibility and independent validation of the reported results. The repository contains the documents used for knowledge ingestion, scripts for system inference and experimentation, benchmark evaluation datasets with corresponding answer keys, reviewer evaluation results, and red-teaming scenarios. It also provides the analysis scripts used to generate the validation and results tables and figures reported in the paper, along with ER diagrams describing the system reference tables.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

All headline figures, tables, and statistical results reported in this paper can be reproduced by executing the scripts provided in the repository, with step-by-step reproduction instructions documented in the README. To ensure full reproducibility, the artifact package corresponding to the experiments reported in this paper is archived as the versioned GitHub tag **v1.0-artifact**, which points to commit **035b70263c2dd9ca911e4776a2c88335d4f3d986** and freezes the repository state used to generate the reported results. The complete artifact repository is available online at [54].

11. Conclusions and Future Work

The digitalization of payroll management and the provision of payroll services through the HelpDesk system enhance operational efficiency, cost-effectiveness, transparency, privacy, and security. By leveraging template and resolver patterns, the system enables reusable, consistent computation across salary slips, tax calculations, and health insurance contributions, reducing manual intervention, shortening processing time, and improving overall productivity. Eliminating paper-based requests and physical documents lowers material consumption, safeguards sensitive data, and prevents loss or theft. Role-based access controls ensure authorized users can access relevant information, promoting collaboration, supporting informed decision-making, and maintaining regulatory compliance. Cross-checking and logging mechanisms support payroll accuracy and auditability.

The adoption of Scrum was instrumental in delivering the HRPIS payroll module effectively. Dividing the project into manageable increments enabled the team to handle the system's complexity and evolving requirements while sustaining continuous stakeholder involvement. Iterative feedback minimized post-release issues, reduced costly rework, and ensured that system features aligned with user needs. Scrum metrics, including burn-up and defect tracking, provided clear visibility into workload distribution, progress, and quality assurance. This approach supported timely delivery, maintained high levels of reliability and usability, and strengthened stakeholder confidence in the development process.

Integrating a retrieval-enhanced chatbot into the HRPIS further demonstrated the benefits of coupling automated payroll processing with intelligent user assistance. The retrieval-augmented architecture, leveraging FAISS and BM25 over curated regulatory texts and institutional policy documents, produced accurate and context-aware responses to HR, payroll, tax, and health insurance queries. Comparative evaluations across multiple language models and configurations showed improved performance compared to baseline language models. System robustness was further validated through empirical red-teaming, indicating high containment with no observed critical failures across adversarial scenarios. Collectively, these improvements reduce HR support burden, enhance operational efficiency, and ensure consistent, compliant assistance, highlighting the potential of retrieval-augmented systems to improve service quality and decision support in enterprise settings.

In addition to methodological choices, several organizational and human factors contributed to the successful release of this complex system. Effective project leadership and strong technical expertise were essential for delivering the software on time and with high quality. Active engagement of stakeholders fostered trust, reduced resistance to the transition from manual to computerized processes, and encouraged institutional acceptance of the system. Support from university management ensured the provision of necessary resources, the ability to adjust schedules when appropriate, effective risk management, and sustained team motivation.

Future work includes extending automation to additional payroll activities, such as calculating overtime for academic and administrative employees based on workloads and pay rates. This process is currently performed manually using spreadsheets and imported into the HRPIS as irregular allowances, which is both time-consuming and error-prone. Further enhancements may also involve enabling finance staff to manage employee funds directly within the system and allowing employees to view their account balances, expanding transparency and usability.

There is also significant potential to incorporate advanced machine learning and artificial intelligence methods, such as predictive salary analytics, payroll fraud detection, and reinforcement learning for compliance-driven payroll optimization. These techniques would broaden the analytical and decision-support capabilities of the platform while complementing its existing rule-based and retrieval-driven mechanisms.

Finally, improving the accuracy and robustness of the retrieval-enhanced chatbot is a key direction for future development. Planned work includes refining the structure of payroll and tax documents by reducing redundancy, reorganizing related content, and applying paragraph-level tagging to improve retrieval precision. A curated set of frequently asked questions will be introduced for low-confidence queries, and operational safeguards such as rate limiting will help ensure stable behavior during live use. Together, these improvements aim to reinforce the reliability, consistency, and institutional value of the HRPIS as a comprehensive and intelligent payroll management solution.

References

[1] K. Alqarni, M. F. Agina, H. A. Khairy, B. S. Al-Romeedy, D. A. Farrag, and R. M. Abdallah, "The effect of electronic human resource management systems on sustainable competitive advantages: The roles of sustainable innovation and organizational agility," *Sustainability,* vol. 15, no. 23, p. 16382, 2023.

[2] GJU. (2025). *GJU Laws, Regulations, and Instructions*. Available: https://www.gju.edu.jo/content/laws-regulations-3492

[3] M. M. P. G. Sarma, N. Rajani, and B. A. Reddy, "Payroll Administration," *International Journal of Information Technology and Computer Engineering,* vol. 11, no. 1, pp. 1-10, 2023.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

[4] E. Iryanie and A. Khalid, "Web-based employee payroll application on minimarket," *International Journal of Educational Technology Research,* vol. 2, no. 4, pp. 527-536, 2024.
[5] M. K. Narula and A. Sah, "The evolution of the Fintech Industry in India: growth, challenges, and future prospects," in *Proceedings of the 3rd International Conference on Optimization Techniques in the Field of Engineering (ICOFE-2024)*, Ethiopia, 2024, pp. 1-11: Elsevier.
[6] B. Saha, "Robotic Process Automation (RPA) in onboarding and offboarding: Impact on payroll accuracy," *IJCSPUB,* vol. 13, no. 2, pp. 237-256, 2023.
[7] A. M. Ahmed, C. N. Mohammed, and A. M. Ahmad, "Web-based payroll management system: design, implementation, and evaluation," *Journal of Electrical Systems Information Technology,* vol. 10, no. 1, p. 17, 2023.
[8] B. Lu, C. Liu, and T. Zhao, "A three-tier salary management system for higher vocational colleges," *International Journal of Multimedia and Ubiquitous Engineering,* vol. 10, no. 4, pp. 91-104, 2015.
[9] F. I. Adhim, R. F. Martin, S. Budiprayitno, and L. P. Rahayu, "Development of employee payroll system using Rational Unified Process (RUP) on odoo platform," *Applied Technology Computing Science Journal,* vol. 5, no. 1, pp. 36-43, 2022.
[10] I. Arfyanti, "Design and development of employee payroll information systems in Indeks Media Teknologi Inc," *International Journal of Information Engineering Electronic Business,* vol. 13, no. 3, pp. 1-12, 2021.
[11] J. Lin, C. Fu, J. Zhou, and L. Gong, "Design of human resource salary management system for state-owned enterprises based on SSH architecture," in *2021 International Conference on E-Commerce and E-Management (ICECEM)*, 2021, pp. 90-93: IEEE.
[12] A. Muktar, Z. S, A. Abdulkadir, and F. A, "Integrated payroll system – a design science approach," *International Research Journal of Engineering and Technology,* vol. 8, no. 6, pp. 759-767, 2021.
[13] T. Wang, N. Li, and H. Li, "Design and development of human resource management computer system for enterprise employees," *PLOS One,* vol. 16, no. 12, p. e0261594, 2021.
[14] M. S. Rumetna, T. N. Lina, I. S. Rajagukguk, F. S. Pormes, and A. B. Santoso, "Payroll information system design using waterfall method," *International Journal of Advances in Data Information Systems,* vol. 3, no. 1, pp. 1-10, 2022.
[15] D. Hindarto and K. Nurfaizi, "Analysis and improvement of payroll systems using Agile methods to achieve efficiency and fairness," *International Journal of Software Engineering and Computer Science,* vol. 3, no. 3, pp. 437-442, 2023.
[16] B. G. Sudarsono, H. H. Fransiscus, D. Y. Bernanda, and J. F. Andry, "Adopting SCRUM framework in a software development of payroll information system," *International Journal of Advanced Trends in Computer Science Engineering,* vol. 9, no. 3, pp. 2604-2611, 2020.
[17] Razorpay. (2025). *RazorpayX Payroll*. Available: https://razorpay.com/docs/payroll/
[18] Payevo. (2025). *Payment evolution payroll software*. Available: https://paymentevolution.com/Payroll
[19] ZenHR. (2025). *Payroll management software*. Available: https://www.zenhr.com/en/modules/payroll
[20] J. H. Marler, "Artificial intelligence, algorithms, and compensation strategy: Challenges and opportunities," *Organizational Dynamics,* vol. 53, no. 1, p. 101039, 2024.
[21] A. J. Nyberg, D. Abdulsalam, O. Cragun, and V. Arumugam, "Algorithm-Based Pay-for-Performance (APFP) systems: Paradoxes in artificial intelligence's influence on pay-for-performance theories," *Human Resource Management Review,* vol. 36, no. 1, p. 101119, 2026.
[22] A. M. Votto, R. Valecha, P. Najafirad, and H. R. Rao, "Artificial intelligence in tactical human resource management: A systematic literature review," *International Journal of Information Management Data Insights,* vol. 1, no. 2, p. 100047, 2021.
[23] S. Meenugu, "AI and ML in payroll automation: A technical perspective," *World Journal of Advanced Engineering Technology and Sciences,* vol. 15, no. 01, pp. 1542-1552, 2025.
[24] M. N. Sakib, M. Salehin, M. Younus, M. A. Al-Omari, M. Sahabuddin, and M. I. Tabash, "The ChatGPT and the future of HR: A critical review on the benefits and challenges of AI chatbots in human resource management," *Multidisciplinary Reviews,* vol. 7, no. 8, pp. 2024136-2024136, 2024.
[25] C. Roberts, R. Kundavaram, A. R. Onteddu, S. Kothapalli, F. A. Tuli, and M. S. Miah, "Chatbots and virtual assistants in HRM: exploring their role in employee engagement and support," *NEXG AI Review of America,* vol. 1, no. 1, pp. 16-31, 2020.
[26] P. Jiang, S. Ouyang, Y. Jiao, M. Zhong, R. Tian, and J. Han, "Retrieval and structuring augmented generation with large language models," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025, pp. 6032-6042.
[27] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval,* vol. 3, no. 4, pp. 333-389, 2009.
[28] A. Afzal, A. Kowsik, R. Fani, and F. Matthes, "Towards optimizing and evaluating a retrieval augmented QA chatbot using LLMs with Human in the Loop," in *Proceedings of the Fifth Workshop on Data Science with Human-in-the-Loop (DaSH 2024)*, Mexico City, Mexico, 2024, pp. 4–16.

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301

[29] Y. Suh, "Design and performance evaluation of LLM-based RAG pipelines for chatbot services in international student admissions," *Electronics,* vol. 14, no. 15, p. 3095, 2025.
[30] A. Abdallah, M. Abdalla, B. Piryani, J. Mozafari, M. Ali, and A. Jatowt, "RerankArena: a unified platform for evaluating retrieval, reranking and RAG with human and LLM feedback," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025, pp. 6593-6597.
[31] M. Russinovich, A. Salem, S. Zanella-Béguelin, and Y. Zunger, "The price of intelligence: three risks inherent in LLMs," *ACM Queue,* vol. 22, no. 6, pp. 38-61, 2024.
[32] O. Belmoukadam *et al.*, "AdversLLM: A practical guide to governance, maturity and risk assessment for LLM-based applications," *International Journal on Cybernetics Informatics,* vol. 13, no. 4, pp. 89-106, 2024.
[33] K. Schwaber and J. Sutherland. (2020). *The 2020 Scrum Guide*. Available: https://scrumguides.org/scrum-guide.html
[34] J. Juneau, *Introducing Java EE 7: a look at what's new*. New York, NY, USA: Apress, 2013.
[35] F. Al-Hawari, "Software design patterns for data management features in web-based information systems," *Journal of King Saud University-Computer and Information Sciences,* vol. 34, no. 10, pp. 10028-10043, 2022.
[36] F. Al-Hawari *et al.*, "Methods to achieve effective web-based learning management modules: MyGJU versus Moodle," *PeerJ Computer Science,* vol. 7, p. e498, 2021.
[37] S. White, M. Hapner, R. Cattell, G. Hamilton, and M. Fisher, *JDBC API tutorial and reference: universal data access for the Java 2 platform*. Addison-Wesley Longman Publishing, 1999.
[38] F. Al-Hawari, "MyGJU student view and its online and preventive registration flow," *International Journal of Applied Engineering Research,* vol. 12, no. 1, pp. 119-133, 2017.
[39] F. Al-Hawari, A. Alufeishat, M. Alshawabkeh, H. Barham, and M. Habahbeh, "The software engineering of a three-tier web-based student information system (MyGJU)," *Computer Applications in Engineering Education,* vol. 25, no. 2, pp. 242-263, 2017.
[40] F. Al-Hawari, K. Tayem, S. Alouneh, and A. Al-Ksasbeh, "Impact of virtual Hadoop cluster scalability on the performance of big data Mapreduce applications," in *24th International Arab Conference on Information Technology (ACIT)*, Ajman, United Arab Emirates, 2023, pp. 1-6: IEEE.
[41] D. Iseminger, *Active directory services for Microsoft Windows 2000*. Redmond, Washington, USA: Microsoft Press, 1999.
[42] P. Paranthaman, V. S. Kumaran, V. Vijayalakshmi, and S. Rimo, "Blockchain technology in HR: enhancing transparency and security in employee data management," in *2024 International Conference on Sustainable Communication Networks and Application (ICSCNA)*, 2024, pp. 388-394: IEEE.
[43] ISTD. (2025). *Income Tax Law No. 38 of 2018*. Available: https://istd.gov.jo/En/List/Laws
[44] LangChain. (2025). *The largest community building the future of LLM apps*. Available: https://www.langchain.com/langchain
[45] Meta. (2025). *Facebook AI Similarity Search*. Available: https://ai.meta.com/tools/faiss/
[46] HuggingFace. (2025). *All-MiniLM-L6-v2*. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[47] HuggingFace. (2025). *All-MPNet-Base-v2*. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[48] Qwen. (2025). *Qwen2.5-72B-Instruct*. Available: https://huggingface.co/Qwen/Qwen2.5-72B-Instruct
[49] HuggingFace. (2025). *Hugging Face Documentations*. Available: https://huggingface.co/docs
[50] F. Al-Hawari and H. Barham, "A machine learning based help desk system for IT service management," *Journal of King Saud University-Computer and Information Sciences,* vol. 33, no. 6, pp. 702-718, 2021.
[51] Python. (2025). *Rank-BM25: A two line search engine*. Available: https://pypi.org/project/rank-bm25/
[52] Mistralai. (2025). *Mistral-7B-Instruct-v0.3*. Available: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[53] Google. (2025). *Gemma-2-9b-it*. Available: https://huggingface.co/google/gemma-2-9b-it
[54] F. Al-Hawari, A. Alufeishat, M. Habahbeh, and A. Alfalayleh. (2026). *HRPIS and RAG-Based Chatbot Evaluation Artifacts [Computer Software]*. Available: https://github.com/firasalhawari/hrpis_rag_chatbot_evaluation

To appear in: Knowledge-Based Systems; **Accepted date**: 20 May 2026; **DOI**: https://doi.org/10.1016/j.knosys.2026.116301