

Generative Agents at Work: Redesigning Administrative Processes at the German Federal Employment Agency

Alina Chircu
University of New Mexico, USA
achircu@unm.edu

Christian Czarnecki
FH Aachen, Germany
czarnecki@fh-aachen.de

Tim Neumüller
Capgemini, Germany
tim.neumueller@capgemini.com

Sebastian Sebrak
Capgemini, Germany
sebastian.sebrak@capgemini.com

Eldar Sultanow
Capgemini, Germany
eldar.sultanow@capgemini.com

Florian Winzer
Federal Employment Agency, Germany
florian.winzer@arbeitsagentur.de

Abstract

The integration of Generative Artificial Intelligence (GenAI) in public administration offers new ways to handle service and technology complexity while meeting high standards. This paper presents a case study from the German Federal Employment Agency, where a multi-agent system using local large language models (LLMs) automates the conversion of information technology (IT) change requests into structured IT development tasks (Jira tickets). Specialized agents interpret requirements, break them into tasks, and generate consistent entries. This improves organizational efficiency, reduces routine work, and outperforms traditional automation by enabling context-aware reasoning and dialogue, all within a secure, on-premise environment. While motivated by Germany's demographic challenges, the findings have global relevance for public sector modernization and automation.

Keywords: Generative Artificial Intelligence (GenAI), Information Technology (IT) Change Requests, Large Language Models (LLMs), Multi-Agent Systems, Organizational Transformation, Public Administration

1. Introduction

Public sector organizations around the world are facing mounting challenges: rising complexity in service delivery, fragmented and aging IT infrastructures, and growing expectations from citizens for responsive, transparent, and human-centric services. In this context, Generative Artificial Intelligence (GenAI) is increasingly recognized as a transformative force in administrative modernization [1], [2].

Notably, the *Organisation for Economic Co-operation and Development (OECD)* has emphasized the strategic relevance of GenAI in the public sector, calling for responsible, trustworthy, and context-sensitive implementations. In its *G7 Toolkit for Artificial Intelligence in the Public Sector* [3], OECD outlines principles and use cases for deploying GenAI to enhance administrative capacity, policy responsiveness, and service quality—while safeguarding transparency, accountability, and data protection.

GenAI includes a variety of deep learning architectures and models that are trained on vast datasets to create original content in various formats, including text, images, audio, and video. Large Language Models (LLMs) are one type of GenAI developed initially for natural language processing [4]. Generative, in this context, describes an architecture designed to capture the underlying distribution of data, as opposed to discovering distinct data classes, as discriminative architectures do [4].

This paper investigates the application of a GenAI LLM-based multi-agent system at the German Federal Employment Agency (Bundesagentur für Arbeit, BA), one of Europe's largest public institutions. While grounded in a specific national context, the case offers generalizable insights into how generative agents can augment complex administrative workflows. The system comprises specialized agents that automate the transformation of unstructured input (e.g., user stories and change requests) into structured development tasks, and operates entirely within the agency's secure, on-premise infrastructure.

The study examines both the orchestration of generative agents within administrative processes and the organizational and technical conditions that shape their effective deployment. Moreover, it reflects

on how these systems compare to conventional automation approaches in terms of adaptability, long-term maintainability, and societal acceptance. By addressing these dimensions, the paper contributes to the growing discourse on GenAI in digital government—highlighting the potential of generative agents as cognitive collaborators in the evolving landscape of public service delivery.

With this contribution, we respond to recent calls in the literature to move from conceptual potential to real-world insights into the operational use of GenAI in the public sector [1], [2], [5]. Our case study illustrates how generative multi-agent systems can be implemented within a high-security, mission-critical environment while adhering to normative principles such as transparency, explainability, and institutional accountability as outlined by the OECD [3].

In doing so, we provide novel insights at the intersection of three research streams: (1) the evolution from traditional automation to LLM-based systems in public administration, (2) the design and orchestration of generative multi-agent architectures, and (3) the role of human–AI collaboration in shaping organizational acceptance and societal impact. These themes are reflected in our research questions and guide the structure of the paper.

The remainder of this paper is organized as follows. Section 2 defines the research objectives and guiding questions. Section 3 reviews related work across the aforementioned research areas. Section 4 presents our case study of the German Federal Employment Agency - including the motivation, context, research methodology, system design, system evaluation, and empirical results. Section 5 discusses the results. In Section 6, we reflect on the broader implications and limitations of our findings. Finally, Section 7 concludes the paper.

2. Research Objective and Questions

The objective of this study is to investigate how GenAI multi-agent systems based on LLMs can be effectively applied in large-scale public IT development environments. We aim to evaluate both technical feasibility and organizational impact in a high-security, real-world setting. Accordingly, we address the following research questions:

- **RQ1:** How can generative agents be orchestrated to automate complex administrative processes in public institutions?
- **RQ2:** What organizational and technical challenges arise when deploying GenAI

multi-agent systems within high-security, on-premise infrastructures?

- **RQ3:** How do generative agents compare to traditional automation approaches in terms of flexibility, maintainability, and societal acceptance?

By addressing these questions, this paper provides practical insights into the implementation of GenAI in the public sector and contributes to the academic discourse on human-AI collaboration in administrative contexts.

3. Related Work

The rise of GenAI has led to a growing body of literature exploring its potential to transform business processes and public administration. This section situates our study within three key research streams: automation in the public sector, generative multi-agent architectures, and human-AI collaboration in high-stakes environments.

3.1. From RPA to GenAI in Public Sector Automation

Traditional automation techniques in public administration have long relied on rule-based systems and Robotic Process Automation (RPA) [6]. While effective for repetitive and highly structured tasks, RPA systems often struggle in dynamic or semi-structured domains where interpretation of natural language and contextual reasoning are required [7]. As a result, public institutions are increasingly exploring GenAI to augment or replace static automation methods [8], [9].

Manual summarization of natural language text into structured documents is usually a very time-consuming task. Prior to the advent of GenAI technologies such as LLMs, automated text summarization was accomplished through statistical, deep learning, and pre-trained language methods. LLM summaries are now emerging as a promising, yet still developing tool [10]. Recent studies have demonstrated the applicability of LLMs for automating document classification, summarization, and decision support in a variety of domains, such as healthcare, finance, education, news, law [4], [10]–[12], as well as government workflows [13]. Ongoing testing indicates LLMs can outperform traditional manual methods and automate government reporting tasks, for example, significantly reducing time and effort while improving quality and accuracy [5].

However, few implementations in secure, large-scale public infrastructures have been documented. There

is also a need to build domain-specific systems in order to overcome some of the inherent limitations of generally-trained LLMs [4], [10].

3.2. Generative Multi-Agent Architectures

Multi-agent systems (MAS) have been a staple of distributed AI for decades [14], but their combination with generative capabilities - especially through LLMs - represents a novel research frontier. Recent frameworks such as CrewAI, AutoGen, and LangGraph enable coordination between specialized GenAI agents to simulate complex reasoning workflows [15], [16]. These systems extend beyond task execution and support planning, feedback integration, and collective goal achievement.

MAS are being used to enhance automated summary quality through self-evaluation and feedback, iterative optimization and automated prompts, or aggregation of local summaries of text sections [10]. For complex documents with many different parts, a MAS architecture can identify key narrative sections and their relative weights, distribute the summarization process across sections, and combine the individual summaries into the final result [11].

However, applications of generative MAS remain mostly experimental and are seldom evaluated in operational, policy-sensitive environments. Our work contributes to this emerging field by demonstrating how generative agents can be used to restructure administrative workflows in a real-world government setting.

3.3. Human-AI Collaboration and Societal Impact

Trust, transparency, and human oversight are central themes in GenAI adoption, particularly in the public sector [17]. Concepts such as “human-in-the-loop” automation and “responsible AI” aim to ensure that AI systems augment rather than replace human judgment - which is especially relevant when deploying LLMs in contexts with legal, ethical, or societal implications [4], [10], [18]. Emerging research indicates that human feedback is crucial for improving summarization model performance, and hybrid approaches that merge human expertise with LLM output can create more reliable outputs [10].

There is a pressing need for empirical case studies that demonstrate how GenAI systems can be embedded into public organizations while meeting regulatory and cultural expectations [19]. Our implementation shows how such systems can support, rather than replace, human decision-makers and act as force multipliers in

resource-constrained institutions.

4. Case Study: Generative AI Agents at the German Federal Employment Agency

4.1. Motivation and Context

The German Federal Employment Agency (Bundesagentur für Arbeit, BA) is responsible for managing employment and welfare processes for millions of citizens. With more than 40,000 internal users and a transactional volume in the billions, it represents a complex, large-scale public sector environment. In the face of demographic change - where around 40,000 employees are expected to retire or leave by 2032 - there is an urgent need for automation that enhances efficiency while maintaining high standards of data protection and compliance.

Our case study focuses on change requests for ALLEGRO, a large-scale IT system that supports social benefit processes for millions of citizens. ALLEGRO is maintained by over 40,000 internal users and processes a transaction volume exceeding several billion euros annually. Traditionally, the IT change request process for this system has been very time-consuming and complex. Internal users can submit requests for ALLEGRO changes (fixing bugs, enhancing functionality, developing new features, etc.) as unstructured user stories, and/or more (but not fully) structured requests for changes (RfCs).

Unstructured user stories are usually free-text descriptions of what internal users want or need. They vary a lot in length, from just a few sentences up to 20 pages. A typical example might be: *“The workflow for approving benefits currently lacks an automatic reminder function when deadlines are missed. We need a solution to show overdue tasks to caseworkers.”* They usually include a problem statement, references to the affected IT systems, and sometimes hints for possible solutions. RfCs are more structured and typically include a title or subject line, detailed description of the issue, affected systems or processes, priority or urgency, proposed solution or expected outcome, and (quite often) a release date or deadline for the change. A typical example may be: *“Automatic email notifications for escalation cases in System XY. Currently missing, causing delays. We propose implementing escalation logic. Target release date: Q3 2025.”* Usually they are between half a page and several pages long, often with bullet points or tables.

Both user stories and RfCs are human-generated text narratives, and require analysis and processing in

order to be translated into structured development tasks, which are handled through the IT development ticketing system, Jira (<https://www.atlassian.com/software/jira>), which provides a centralized platform for planning, tracking, and releasing IT projects and is used by more than 55,000 companies globally.

4.2. Research Method

This research follows a pragmatic, design science approach [20], which governs the creation of innovative artifacts (such as constructs, models, methods, or instantiations) that solve real-world problems. We follow the recommended iterative cycle of building and evaluating such an artifact, paying attention to both the practical relevance of the artifact and the rigor of its development and evaluation. Our focus is the development and evaluation of a GenAI MSA in a real-world administrative setting - the ALLEGRO system at BA. This environment provides both a high-impact application domain and strict operational constraints.

Within this context, we co-designed a GenAI-based agent system that automates the transformation of internal user stories and RfCs into structured Jira tickets for completing the changes. Our goal was to reduce repetitive manual tasks and improve process quality by introducing four specialized agents — *Reader*, *Planner*, *Creator*, and *Reviewer* — based on local LLMs. These agents were implemented and orchestrated using the CrewAI framework [15] and embedded into the agency's on-premise IT landscape.

Following [20], the research process stressed relevance (identifying inefficiencies in RfC processing), rigor (drawing from MAS and GenAI literature), and iterative design/evaluation/refinement loops. The process also involved close collaboration between enterprise architects, domain experts, and IT operations teams. The iterative design science cycle included the following stages: problem identification, artifact design, evaluation with authentic RfCs, and refinement. To evaluate and refine the system, we conducted multiple test runs using real RfC documents and evaluated system output based on plausibility, completeness, and alignment with existing Jira ticket standards. Qualitative feedback from users and technical staff supplemented these observations.

This practice-oriented methodology aligns with the design science paradigm, where the artifact (in our case: the agent-based system) is both the outcome and the vehicle for inquiry. Rather than isolating variables in a lab setting, we evaluated the artifact performance under real-world constraints —organizational, legal, and

technical. The resulting insights can inform both the academic discussion on GenAI systems in the public sector and the practical roadmap for their deployment in large-scale administrative ecosystems.

4.3. System Design

To address these challenges, the BA has implemented a pilot system of generative agents based on local LLMs. The full architecture consists of four specialized agents:

- **Reader Agent** extracts relevant content from change requests - RfCs and user stories - covering the core of the request: What's the problem? (e.g., *"No deadline monitoring implemented"*); Which system or process? (e.g., *"System XY, Process Z"*); What's needed? (e.g., *"Create automatic dashboard reminders"*).
- **Planner Agent** breaks down the extracted information into clear actionable work items that make it easier for humans or agents to handle the work step by step, such as: *"Analyze the current deadline process"*; *"Design the reminder logic"*; *"Set up the technical implementation (API or database)"*; *"Test and integrate into the system."*
- **Creator Agent** generates Jira tickets containing title, description, priority, and ticket type by following the Jira ticketing best practice guidelines: short and clear title, detailed but concise description, and incorporation of all key metadata. For example, such a ticket includes: Title: *"Implement reminder function for missed deadlines in System XY"*; Description: *"Currently, deadlines in Process Z are not tracked. No reminders are sent when deadlines are missed. Please implement a dashboard or email reminders to help staff reduce manual work."*; Priority: *"High"*; Category: *"Feature request"*; Expected outcome: *"Automatic reminders within 48 hours of a missed deadline."*
- **Reviewer Agent** checks for consistency, duplication, and completeness, incorporating feedback into an iterative improvement loop. Consistency checks make sure the ticket really reflects what is in the change request. Duplication checks look for existing tickets with similar content to avoid double work. Completeness checks and feedback loop improvement ensure that, if the ticket is incomplete or unclear,

a request is sent to the Creator Agent for improvements.

The current prototype includes three working agents (Reader, Planner, and Creator), with the fourth agent (Reviewer) being in the development stage, to be integrated in subsequent iterations to complete the feedback loop. They work sequential and pass the result to the next agent. The first agent reads the change request documents with a custom tool and extracts the most important information. It then passes the information to the next agent via the CrewAI framework. The second agent uses the input as well as Jira formatting rules and a ticket skeleton passed as context to create the required ticket elements. The ticket is parsed as a custom BaseModel derived class to the last agent. He uses another tool to send the ticket to the Jira API where it is created.

All components are designed to operate fully within the BA's secure on-premise environment, ensuring compliance with strict data protection regulations that no data leaves the agency's infrastructure. The agents are coordinated via CrewAI (<https://github.com/joaomdmoura/crewAI>), an open-source platform for multi-agent orchestration through well-defined roles and communication protocols.

Local LLMs (which do not require external API calls) — including Aleph Alpha, LLaMA, and Mistral — power the system, and do not require external API calls. The Reader Agent used a structured prompt with three explicit fields ('Problem', 'System', 'Required Change'). Local inference ran on BA's secured GPU cluster (4xA100, 80GB each), enabling processing of documents up to 16k tokens. Pre-processing and token management address the challenge of handling large documents (some exceeding 20 pages). Optimization strategies included token trimming and contextual filtering of non-relevant boilerplate content.

The Jira integration is a core part of the prototype's functionality. Using the Jira REST API, the Creator Agent directly generates real Jira tickets within the internal project space of the ALLEGRO team. All ticket generations are logged and traceable. Final approval remains with a human-in-the-loop, who validates and submits the ticket. This feedback is also returned to the Reviewer Agent to continuously improve performance.

In test deployments, the system showed high robustness and plausibility in ticket generation, even for medium-complexity change requests. Only exceptionally long or ambiguous documents required manual post-processing. The Jira prototype is now actively being evaluated as a reusable component across other administrative processes within the agency.

4.4. Prompt Language and Persona Prompting Considerations

The development of the prototype considered the impact of the prompt language and persona prompting on the system performance. In terms of prompt language, we analyzed the tradeoffs of using native (German) language versus English language. Most contemporary LLMs are predominantly trained on English-language data, yet they possess capabilities to handle multiple languages [21], [22]. While the use of native languages is particularly suitable for understanding emotions and co-reference resolution, English prompts are better suited for solving mathematical problems (see, for example, [23]), summarization, causal reasoning, natural language inference, paraphrasing, and answering questions without context [24].

In terms of persona prompting, we considered the impact of persona prompting on MSA role division and performance. LLMs can simulate roles and personalities [25], [26] - a technique known as role-playing prompting or persona prompting. In contrast to structured prompts (e.g., "*Answer the question in Italian*") or system prompts that run in the background and define the system's role (e.g., "*You are a friendly, helpful assistant*"), the goal of persona prompting is to simulate a role [26] and focus and contextualize LLM responses by activating domain-specific knowledge (e.g., technical or medical) [27], increasing stylistic consistency (e.g., factual or empathetic) [28], and enhancing the coherence of responses. In multi-agent systems, persona prompts enable explicit role division among different agents who cooperate and communicate with each other to solve complex tasks [27].

In the prototype, prompt language and persona prompting were evaluated primarily to assess their effect on Jira ticket accuracy and consistency. In practice, English-based prompts improved task decomposition, while German-native prompts better preserved domain-specific terminology. Persona prompting stabilized Planner Agent outputs by simulating domain experts.

4.5. System Evaluation

The evaluation of the prototype uses several well-known LLMs to generate 50 Jira tickets. The inputs are authentic RfCs sampled across domains (benefits, IT infrastructure, employment services) and spanning various lengths, formats, and resulting complexity levels. Low complexity RfCs were less than 1 page in length with a single change. Medium complexity

RfCs were 2–5 pages in length with multiple but related changes. High complexity RfCs were longer than 5 pages with mixed or conflicting requirements. Evaluation of the results was performed by three domain experts (with an average 12 years of experience). Inter-rater agreement among these domain experts was very high (Cohen’s kappa =0.81).

In addition to the change requests documents, Jira formatting rules and a reference skeleton of a ticket were inputted in each model. For the evaluation, each generated ticket was divided into 3 sections: the generated title of the ticket, the generated goal definition of the input document, and the generated tasks needed in order to complete the ticket.

The decision to divide each generated ticket into three distinct sections—title, goal, and tasks—was made to enable a more granular and transparent evaluation of model performance. This structure allows us to identify whether a particularly strong or weak component disproportionately influences the overall score. By isolating these sections, the evaluation can highlight specific strengths and weaknesses of the generated output, thereby providing clearer insights into areas requiring improvement. The weighting of the ticket sections was determined to reflect their relative importance in ensuring both clarity and completeness of the generated Jira ticket. The title was divided into two components: 40% allocated to the ID section and 60% to the descriptive title text. This allocation reflects the greater importance of a clear and meaningful title for the overall usability of the ticket compared to the mere presence of a correctly formatted identifier. This aligns with guidelines in [29], [30]. For the goal and task sections, a 50/50 weighting between formatting and content was applied. This decision acknowledges that while accurate content is essential for conveying the intended purpose and required actions, proper formatting is equally critical for readability and adherence to Jira standards. A ticket lacking structural consistency can lead to ambiguity and difficulties when reading the ticket. Therefore, equal emphasis on both aspects ensures that the generated tickets are not only semantically correct but also well-structured and user-friendly.

For the title, several reference titles have been used to calculate the final metric using the BERTScore F1 score - a weighted average of precision and recall ranging from 0 (worst performance) to 1 (perfect performance) [29]. The BERTScore F1 score was calculated between each reference title and the generated title. In the end the highest score was picked. Through this approach, we tried to accommodate for all the paraphrasing. This approach accounts for variations

in wording, such as synonyms and alternative sentence structures, which are common in LLM-generated text. For example, the statement “The apple is green” can appear in multiple forms, such as “Green is the apple,” “The color of the apple is green,” or “The fruit is an apple and it is green.” Considering these paraphrasing variations is essential to ensure that semantically equivalent outputs are not penalized with low similarity scores solely due to differences in phrasing. The same methodology has been used for the goal and task sections. For the metric SBERT we used a Sentence-BERT model further explained in [30] in combination with Cosinus Similarity to calculate the semantic similarity of the sections.

Field Accuracy is another metric used to evaluate the models ability to extract certain fields inside the change request documents. The Error metric describes the percentage of missing or wrong extracted information inside the final Jira ticket. One example for this being a wrong date labeled as the start date instead of the correct starting date.

Triple Similarity (TriSim) is a simple yet effective ensemble metric that combines three complementary semantic scores - two SBERT-based cosine similarities and one BERTScore F1 - into a single unified score. After normalizing the two cosine values from [-1,+1] to [0,1], TriSim is computed as the unweighted average of the three values.

The selected metrics aim to provide a comprehensive understanding of LLM performance within the given context. BERTScore F1 and SBERT capture semantic similarity to the reference tickets, enabling a content-focused evaluation. Field Accuracy and Error assess the model’s ability to extract key information from the source documents, ensuring structural and factual correctness. Finally, TriSim combines all semantic similarity measures into a single aggregated score, allowing for an overall assessment of semantic alignment with the reference tickets.

4.6. Results

Table 1 summarizes the various performance metrics for each one of the models used in the evaluation. From a practical standpoint, scores above 0.70 in TriSim and SBERT consistently indicated the generated tickets were usable without further revision. This value was determined empirically, as tickets achieving a score of 0.70 or higher consistently demonstrated sufficient semantic similarity and structural adequacy to be considered usable within the evaluation context. Models below this threshold required substantial manual editing. Field Accuracy above 0.95 correlated with

seamless Jira integration, while error rates below 0.05 were considered acceptable for production use.

OpenAI GPT 4.1 shows the highest overall performance with a Mean TriSim value of 0.8473, closely followed by DeepSeek V3 0324 and GPT 4o-mini, each with strong scores of 0.8345. In contrast, models such as Vicuna 7B v1.5 (0.0336) and Llama3.2-3B (0.1831) significantly lag behind. In case of Vicuna this is the result of the small context length of 2K tokens as well as not being suited for the German language.

Llama3.1-8B achieves perfect results (1.0000), followed closely by Mistral Nemo Instruct (0.9999), GPT 4o-mini (0.9912) and DeepSeek V3 (0.9779). Models such as Falcon3 7B Instruct (0.7836) and Gemini 2.0 Flash (0.6168) score significantly lower. The perfect score of Llama3.1-8B in this case is correlated to an occurrence during title generation. It managed to generate a perfect replica of a reference title in 16% of the generated tickets resulting in a unnatural push of the TriSim metric. Gemini 2.0 Flash, on the other hand, did not manage to incorporate the ID in the ID-part of the title mentioned in the recent chapter. In 96% of the generated tickets it mismatched the required ID part format resulting in these lower TriSim scores. On the contrary Gemini 2.0 surpasses Llama3.1-8B on average both in SBERT goal and task results with 0.8235 and 0.8056 to 0.6165 and 0.6748.

In regards to the goal results, we witnessed that the best scores come from DeepSeek V3 (0.9644) and Mistral-Small-24B (0.9631). In contrast, Vicuna 7B performs poorly here with just 0.0243. When reviewing the SBERT of the task, GPT 4.1 has the highest performance with 0.9249, closely followed by Mistral-Small-24B (0.9450). Falcon3 7B (0.8264) and particularly Vicuna 7B (0.4734) show significantly lower performance.

Reviewing the results, there are several instances where the minimal value of F1 for the title, the SBERT for goal, or SBERT for task are 0. The reason is, that the LLM did not manage to incorporate the relevant section in the right expected format or there was an error during execution.

DeepSeek V3 clearly dominates with a value of 0.9889 in Field Accuracy closely followed by GPT 4o-mini with 0.9756. Vicuna 7B shows an almost negligible value of 0.0068. The lowest error rates are provided by DeepSeek V3 (0.0025), GPT 4.1, and GPT 4o-mini (both 0.0000), while Vicuna 7B (0.9923) show an extremely high error rate, significantly detracting from its usability. Models exceeding the 0.70 TriSim threshold include GPT-4.1, GPT-4o-mini, DeepSeek V3, Gemini 2.5, and Llama4 Maverick, indicating

acceptable overall quality. Models such as Vicuna 7B and Llama3.2-3B fall far below this threshold, making them unsuitable for production use.

Analyzing the models from an on-premise versus cloud-hosted perspective reveals distinct advantages and trade-offs. On-premise deployable models, such as Mistral-Small-24B, Llama4 Maverick, and the QWEN2.5 family, offer strong performance and increased flexibility in sensitive or restricted environments. Specifically, Mistral-Small-24B demonstrated excellent metrics, particularly in SBERT *Goal* (0.9631) and *Task* (0.9450), making it suitable for organizational scenarios demanding confidentiality and control.

Cloud-based proprietary models like OpenAI GPT 4.1, GPT 4o-mini, and Gemini provide superior stability and minimal error rates, ideal for applications where accuracy and reliability are paramount. GPT 4.1 and GPT 4o-mini notably offer zero error rates, strongly positioning them as optimal choices for critical business processes requiring robust and precise outcomes. In summary, while proprietary cloud solutions demonstrate optimal overall performance, on-premise and open-source alternatives remain highly competitive, particularly where data privacy, cost management, and organizational compliance are prioritized.

5. Discussion

The deployment of the generative multi-agent system yielded valuable insights on both technical feasibility and organizational impact. Below, we summarize the key findings.

In terms of **technical performance**, the agents performed reliably across a wide range of inputs. In over 80% of test cases, the system generated Jira tickets that were deemed valid, complete, and actionable by domain experts. Specifically: a) the quality of the title and descriptions was high, as the Creator Agent produced clear, context-appropriate ticket titles and summaries that reflected the core of the RfC; b) structural consistency was achieved as the Planner Agent maintained logical task decomposition, resulting in well-formed ticket sequences; c) tokens were appropriately managed as token limits (e.g., 8192 tokens for local LLMs) were effectively handled through pre-processing techniques such as removal of boilerplate sections and whitespace; and d) on-premise compatibility was accomplished as all components operated within the agency's closed infrastructure without external dependencies or data leaks.

In terms of **organizational impact**, the prototype

Table 1. Performance Metrics of Different Generative Models (Compact Header)

Model	Mean TriSim	Max F1 Title	Min F1 Title	Max SBERT Goal	Min SBERT Goal	Max SBERT Task	Min SBERT Task	Mean Field Accuracy	Mean Error
Falcon3 7B Instruct	0.2808	0.7836	0.0000	0.9366	0.0000	0.8264	0.0000	0.4978	0.4600
Mistral-Small-24B Instruct	0.7270	0.9592	0.0000	0.9631	0.0000	0.9450	0.0000	0.8565	0.1302
Mistral Nemo Instruct 2407	0.5646	0.9999	0.0000	0.9657	0.0000	0.9284	0.0000	0.6259	0.3444
QWEN2.5 7B	0.6600	0.8256	0.1931	0.9214	0.3005	0.9016	0.3905	0.8222	0.1050
QWEN2.5 32B	0.7424	0.9137	0.4191	0.9585	0.0243	0.8944	0.5376	0.7644	0.2175
DeepSeek V3 0324	0.8345	0.9779	0.5342	0.9644	0.8102	0.9086	0.7312	0.9889	0.0025
Llama3.1-8B	0.6544	1.0000	0.2000	0.9480	0.0000	0.8937	0.0000	0.7511	0.1900
Llama3.2-3B	0.1831	0.5509	0.0000	0.8883	0.0000	0.8088	0.0000	0.4286	0.5476
Llama3.3-70B	0.5636	0.9295	0.8483	0.9426	0.0243	0.8418	0.0233	0.8956	0.0600
Llama4 Maverick 17B-128e Instruct	0.8148	0.9779	0.4986	0.9015	0.7665	0.8539	0.7596	0.9311	0.0000
Vicuna 7B v1.5	0.0336	0.6289	0.0000	0.0243	0.0000	0.4734	0.0000	0.0068	0.9923
Gemini 2.0 Flash	0.6404	0.6168	0.1107	0.9112	0.6120	0.9071	0.0000	0.9378	0.0500
Gemini 2.5 Flash Preview 05-20	0.8237	0.9267	0.5283	0.9616	0.7860	0.9441	0.7010	0.8178	0.1800
OPENAI GPT 4.1 2025/04/14	0.8473	0.9779	0.5349	0.9467	0.8086	0.9249	0.8161	0.9022	0.0000
OPENAI GPT 4o-mini 2024/07/18	0.8345	0.9912	0.5515	0.9346	0.7256	0.9015	0.7250	0.9756	0.0000

significantly reduced the manual effort required to process change requests. Manual ticket creation (reading, summarizing, typing) was largely eliminated, resulting in significant time savings. Preliminary measurements showed an average reduction of 6–8 minutes per ticket compared to manual creation. Across 1,000 monthly RfCs, this translates into 100–130 work hours saved. In addition, ticket structure and language became more consistent across different teams, resulting in increase standardization. Early testers described the generated tickets as “impressively accurate” and “immediately usable,” resulting in more user satisfaction. Last, but not least, the architecture was designed to be reusable for additional workflows beyond ALLEGRO, such as support requests or policy documentation, resulting in enhanced scalability.

In terms of **limitations**, several challenges emerged during implementation. First, long source documents sometimes resulted in incomplete outputs. RfCs exceeding the LLM token limit required truncation or summarization, which occasionally led to loss of relevant detail (this happened in 8% of cases). Second, ambiguity in source documents sometimes occurred (this happened in 5% of cases). Inconsistent or contradictory RfCs reduced the quality of generated outputs, underscoring the importance of clear input data. For example, an RfC simultaneously requesting a new interface and the deprecation of the same API resulted in conflicting Jira tasks. For these highly ambiguous RfCs, human revision outweighed automation gains. In these cases, the system acted as a drafting assistant rather than a time-saver. Third, effort estimation was challenging. The feature for predicting estimated effort is not yet fully operational and remains an area for future improvement. Fourth, all performance thresholds defined in this study are project-specific and should be further validated in future work. Last, but not least, human oversight was necessary. Despite high-quality outputs, human validation remains essential for compliance and accountability in public

administration.

Overall, the results demonstrate that generative agents can automate and standardize complex administrative tasks in secure, high-stakes environments provided they are supported by clear data structures, governance mechanisms, and human oversight. They reduce routine workload, freeing up human resources for more value-added work. The case study illustrates that GenAI deployment in public administration is not about replacing jobs but about redesigning workflows for efficiency and citizen-centric service delivery.

6. Implications and Future Directions

The deployment of generative agents in a high-stakes, process-driven environment such as the German Federal Employment Agency allows us to reflect on broader implications for AI-based automation in public administration.

Beyond RPA: Contextual Intelligence through GenAI. Compared to Robotic Process Automation (RPA), which operates through static rule sets and UI scripting, our generative agent system demonstrates significantly higher flexibility and cognitive depth. While RPA is well-suited for deterministic, repetitive tasks, it falls short when faced with ambiguous or semi-structured data—such as complex change requests. GenAI agents, on the other hand, offer the ability to interpret context, apply soft rules, and engage in reasoning-like behavior. This makes them particularly valuable for tasks where understanding, not just execution, is required.

Organizational Transformation and Human Roles. The adoption of generative agents redefines the boundary between human and machine work. Rather than replacing human labor, the system offloads tedious and repetitive tasks, allowing domain experts to focus on decision-making and process refinement. This shift has implications for workforce strategy, skill development, and digital mindset within the organization. It also

supports a transition toward more agile, adaptive public service delivery.

Trust, Transparency, and Responsibility.

Deploying GenAI in public administration raises questions of trust, explainability, and accountability. Our system addresses these concerns through a human-in-the-loop design, logging mechanisms, and on-premise hosting. The Reviewer Agent enables feedback loops that ensure output quality and traceability. Nonetheless, ongoing stakeholder engagement and training are essential to foster trust among users and maintain alignment with legal and ethical standards in the public sector.

Scalability and Transferability.

Although developed within the ALLEGRO system, the architecture is modular and can be adapted to other administrative domains. Candidate areas include document management, support ticketing, and citizen-facing services. However, successful transfer requires careful adaptation to local data structures, policy contexts, and organizational readiness. This case study offers an actionable blueprint for public institutions seeking to leverage GenAI while upholding values of transparency, fairness, and efficiency. As a pioneering step, it demonstrates how GenAI can complement human decision-making in large-scale public IT systems while adhering to strict data protection standards and fostering organizational trust.

7. Conclusion

This paper has presented a pragmatic implementation of a generative multi-agent system within the German Federal Employment Agency, targeting the automation of administrative ticket creation in a highly complex IT development landscape. By orchestrating local LLMs into a purposeful agent workflow (Reader, Planner, Creator, Reviewer), we demonstrated that GenAI can be effectively embedded into secure, process-critical government infrastructures.

The results indicate that such systems not only improve efficiency and consistency but also reshape human work by relieving experts from repetitive tasks and supporting more adaptive service delivery. Unlike traditional RPA, our approach leverages contextual reasoning and collaborative agent behavior to handle ambiguous input and maintain flexibility.

Looking forward, the architecture is well-positioned for expansion into adjacent public sector processes such as document review, citizen communication, and regulatory compliance. By combining technological capability with organizational responsibility, generative agents offer a pathway toward more intelligent,

responsive, and humane public services.

References

- [1] I. Lindgren, C. Ø. Madsen, A. Østergaard Munk, and A. F. van Veenstra, *Artificial Intelligence and the Public Sector: Transforming Public Services*. Springer, 2024.
- [2] M. Zanini, J. R. Gil-Garcia, and L. F. Luna-Reyes, “The generative ai revolution in the public sector: Opportunities, challenges, and a research agenda,” *Communications of the ACM*, vol. 67, no. 5, pp. 30–32, 2024.
- [3] OECD, “G7 toolkit for artificial intelligence in the public sector,” OECD Publishing, Tech. Rep., 2024.
- [4] A. A. Linkon, M. Shaima, M. S. U. Sarker, *et al.*, “Advancements and applications of generative artificial intelligence and large language models on business management: A comprehensive review,” *Journal of Computer Science and Technology Studies*, vol. 6, no. 1, pp. 225–232, 2024.
- [5] R. Gupta, G. Pandey, and S. K. Pal, “Automating government report generation: A generative ai approach for efficient data extraction, analysis, and visualization,” *Digital Government: Research and Practice*, vol. 6, no. 1, pp. 1–10, 2025.
- [6] S. Agostinelli, P. Laranjeira, and J. Neves, “Robotic process automation: The next transformation lever for shared services,” *Journal of Service Management*, vol. 31, no. 3, pp. 321–339, 2020.
- [7] R. Syed, W. Bandara, and E. French, “Automation in the public sector: Rpa and beyond,” *Government Information Quarterly*, vol. 37, no. 4, p. 101496, 2020.
- [8] M. Tambe, E. Rice, and M. Morton, “Ai for social good: Unlocking the opportunity for public sector transformation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14979–14985.
- [9] D. Eisen and J. Linder, “Large language models in government: Opportunities and challenges,” *AI & Society*, 2022.
- [10] H. Zhang, P. S. Yu, and J. Zhang, “A systematic survey of text summarization: From statistical methods to large language models,” *ACM Computing Surveys*, 2024.
- [11] N. K. Shukla, R. Katikeri, M. Raja, *et al.*, “Generative ai approach to distributed

- summarization of financial narratives,” in *2023 IEEE International Conference on Big Data (BigData)*, IEEE, 2023, pp. 2872–2876.
- [12] A. Dubey, R. Sharma, M. Diwakar, P. Singh, A. K. Mishra, and S. Lamba, “Genai-based news article analysis chatbot,” in *2024 4th International Conference on Advancement in Electronics & Communication Engineering (AECE)*, IEEE, 2024, pp. 476–480.
- [13] D. Lindebaum and G. Vial, “Ai in bureaucracy: Understanding resistance and adaptation,” *Organization Studies*, 2022.
- [14] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2009.
- [15] O. Collective, *Crewai: A framework for multi-agent coordination with llms*, <https://github.com/joaomdmoura/crewAI>, Accessed: 2025-05-28, 2023.
- [16] Y. Liang, X. Lin, Q. Chen, J. Liu, *et al.*, “Taskmatrix.ai: Enabling multimodal agents via chatgpt plugins,” *arXiv preprint arXiv:2303.16434*, 2023.
- [17] B. D. Mittelstadt, “Principles alone cannot guarantee ethical ai,” *Nature Machine Intelligence*, vol. 1, no. 11, pp. 501–507, 2019.
- [18] A. Verma, S. Pathak, and X. Wang, “Responsible ai: From principles to practice,” *Communications of the ACM*, vol. 65, no. 7, pp. 62–69, 2022.
- [19] B. Zhang, P. Richter, and M. Jarke, “Building trustworthy ai for government services: Challenges and design principles,” in *Proceedings of the International Conference on e-Government*, 2023.
- [20] A. Hevner, S. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [21] X. Zhang, S. Li, B. Hauer, N. Shi, and G. Kondrak, “Don’t trust chatgpt when your question is not in english: A study of multilingual abilities and types of llms,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore: Association for Computational Linguistics, 2023, pp. 7915–7927.
- [22] I. Mondshine, T. Paz-Argaman, and R. Tsarfaty, “Beyond english: The impact of prompt translation strategies across languages and tasks in multilingual llms,” *arXiv preprint arXiv:2502.09331*, 2025.
- [23] F. Shi, M. Suzgun, M. Freitag, *et al.*, “Language models are multilingual chain-of-thought reasoners,” *arXiv preprint arXiv:2210.03057*, 2022.
- [24] S. Vatsal and H. Dubey, “A survey of prompt engineering methods in large language models for different nlp tasks,” *arXiv preprint arXiv:2407.12994*, 2024.
- [25] A. Kong, S. Zhao, H. Chen, *et al.*, “Better zero-shot reasoning with role-play prompting,” *arXiv preprint arXiv:2308.07702*, 2023.
- [26] M. Zheng, J. Pei, L. Logeswaran, M. Lee, and D. Jurgens, “When ”a helpful assistant” is not really helpful: Personas in system prompts do not improve performances of large language models,” *arXiv preprint arXiv:2311.10054*, 2023.
- [27] Y.-M. Tseng, Y.-C. Huang, T.-Y. Hsiao, *et al.*, “Two tales of persona in llms: A survey of role-playing and personalization,” *arXiv preprint arXiv:2406.01171*, 2024.
- [28] J. Chen, X. Wang, R. Xu, *et al.*, “From persona to personalization: A survey on role-playing language agents,” *arXiv preprint arXiv:2404.18231*, 2024.
- [29] X. Wang, W. Wang, and C. Liu, “A design science research methodology for information systems research,” *arXiv preprint arXiv:1904.09675*, 2019.
- [30] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” *arXiv preprint arXiv:1908.10084*, 2019.