## Chapter 1: Introduction

### 1.1 Background

Modern software development environments heavily depend on issue tracking systems like Jira to manage bugs, feature requests, enhancements, and other support tickets. These tickets often contain rich metadata, including reporter details, assignees, project components, issue types, timestamps, and resolution history. Efficient retrieval of relevant historical tickets is critical to accelerating issue triaging, facilitating knowledge reuse, and reducing operational bottlenecks.

Conventional Retrieval-Augmented Generation (RAG) systems, which integrate dense vector similarity search with generative language models, have significantly improved the retrieval process by incorporating semantic matching capabilities. However, such vector-based models lack explicit mechanisms to capture structured relationships between ticket attributes, resulting in a surface-level understanding of ticket data. They also fail to provide transparent reasoning paths that explain why certain tickets are retrieved in response to a query.

Knowledge Graphs (KGs) offer a promising alternative, as they provide a structured, semantically rich representation of entities such as reporter, assignee, component, issue type, and their interconnections. Integrating KGs with RAG frameworks introduces relationship-aware reasoning capabilities, enabling better handling of complex multi-entity queries and improved explainability.

### 1.2 Problem Statement

The existing Jira ticket retrieval systems are primarily based on vector similarity search, which approximates semantic similarity using dense embeddings of ticket summaries and user queries. While this enables retrieval of contextually similar tickets, it lacks the ability to capture explicit interrelationships between structured ticket metadata. This limitation prevents the system from effectively addressing multi-entity, relationship-driven queries, leading to suboptimal ticket matching and slower issue resolution.

Additionally, the opaque nature of vector similarity results erodes user trust, as the rationale behind retrieving specific tickets remains unexplained. These shortcomings necessitate the development of a Knowledge Graph-enhanced retrieval system capable of reasoning over entity relationships and offering explainable reasoning paths for each retrieval result.

### 1.3 Objectives

* To construct a domain-specific Knowledge Graph from historical Jira ticket data, capturing key entities and their interrelationships.
* To integrate the Knowledge Graph with an existing vector-based RAG system for hybrid retrieval.
* To enable the system to process complex, multi-entity, relationship-based queries with improved contextual relevance.
* To enhance the explainability and transparency of retrieval outcomes using traceable reasoning paths.
* To evaluate the system’s performance using quantitative metrics such as Mean Reciprocal Rank (MRR), Precision@k, and qualitative user feedback.

### 1.4 Scope of the Research

This research encompasses the end-to-end design, development, and offline evaluation of a Knowledge Graph-enhanced Jira ticket retrieval system. It includes the construction of a domain-specific Knowledge Graph, integration with an existing RAG framework, development of a query interface for multi-entity queries, and rigorous performance evaluation. While this report documents mid-semester progress on methodology, literature review, and system design, subsequent phases will focus on expanding graph coverage, integrating explainability modules, and deploying the system on enterprise infrastructure.

### 1.5 Significance of the Study

The proposed system addresses critical gaps in current enterprise ticket retrieval pipelines by offering an explainable, relationship-aware retrieval mechanism. By modeling explicit relationships between ticket entities, the system improves retrieval precision, reduces triaging time, and promotes efficient organizational knowledge reuse. The transparency provided by the Knowledge Graph’s reasoning paths builds user trust and confidence in the retrieval process.

Moreover, the modular, scalable, and on-premise-compatible design positions the system for broader enterprise adoption, enabling seamless integration with existing customer relationship management (CRM) and multilingual ticketing platforms.

## Chapter 2: Literature Review

### 2.1 Theoretical Framework

The proposed work draws upon foundational concepts from Information Retrieval (IR), Knowledge Graph construction, and Retrieval-Augmented Generation (RAG) systems. Information Retrieval involves techniques for identifying relevant documents or records in response to user queries, historically relying on keyword-based or vector-space models. Recent advancements in deep learning have produced RAG frameworks capable of combining dense vector similarity search with generative language models for response generation or document retrieval.

However, dense vector approaches have limitations in representing structured relationships inherent in enterprise systems like Jira. Knowledge Graphs address this by providing a semantically rich, graph-based representation of entities (e.g., reporters, assignees, components) and their interconnections. Graph-based models allow multi-hop reasoning and precise relationship-based filtering, essential for complex query handling and explainable retrieval.

The integration of Knowledge Graphs into RAG pipelines has shown promising improvements in both retrieval performance and result transparency. This dissertation situates itself within this emerging paradigm, applying it to enterprise issue management systems.

### 2.2 Review of Related Technologies

Several studies have examined the fusion of Knowledge Graphs with retrieval systems:

* **Linders & Tomczak (2025)** proposed a KG-extended RAG framework for question answering tasks. Their system improved multi-hop reasoning and explainability without retraining language models, demonstrating superior performance on the MetaQA benchmark.
* **Sanmartin (2024)** introduced KG-RAG, a pipeline constructing Knowledge Graphs from unstructured text and performing retrieval over graph structures. Their results indicated a significant reduction in hallucinations and improved fact consistency in generative outputs.
* **Xu et al. (2024)** applied a similar methodology to customer service question answering systems by building a KG from historical customer issues. They achieved a 77.6% improvement in Mean Reciprocal Rank (MRR) and a 28.6% reduction in median issue resolution time.

These findings confirm the viability of integrating Knowledge Graphs with RAG frameworks for information-heavy enterprise use cases, providing a strong justification for applying the approach to Jira ticket retrieval.

### 2.3 Research Gap Identification

While KG-augmented retrieval systems have been explored in domains like question answering and healthcare, there is a lack of work specifically targeting issue tracking systems like Jira. Enterprise ticket data features structured fields (e.g., assignee, reporter, component) whose interrelationships are critical for contextual relevance in retrieval—a dimension not effectively handled by existing vector-based RAG implementations.

Additionally, most existing solutions prioritize cloud-based deployments, raising data security and compliance concerns for enterprises. An on-premise, Knowledge Graph-enhanced retrieval system for Jira with explainable reasoning paths thus represents a novel and valuable contribution.

## Chapter 3: Research Methodology

### 3.1 Research Design

The project follows an incremental, phased methodology over 16 weeks:

* **Phase 1:** Conduct a literature review and landscape analysis of existing RAG, KG, and enterprise IR systems.
* **Phase 2:** Design a Knowledge Graph schema tailored to Jira data, capturing key entities like reporter, assignee, component, issue type, and their relationships.
* **Phase 3:** Extract and preprocess historical Jira ticket data to populate a prototype Knowledge Graph using Neo4j.
* **Phase 4:** Integrate the KG with an existing vector-based RAG pipeline.
* **Phase 5:** Develop a query interface supporting complex, relationship-based queries.
* **Phase 6:** Evaluate system performance using metrics such as MRR, precision, and qualitative user feedback.

### 3.2 Data Collection Methods

The primary dataset comprises historical Jira tickets containing metadata like ticket ID, summary, description, issue type, reporter, assignee, components, and timestamps. The data is sourced from internal enterprise repositories while ensuring data privacy and security compliance.

The preprocessing phase includes cleaning and normalizing ticket metadata, extracting entity instances, and establishing relationship mappings suitable for graph representation.

### 3.3 Tools and Frameworks

* **Neo4j:** For Knowledge Graph modeling and Cypher-based querying.
* **Python Libraries:** Pandas for data preprocessing, NetworkX for graph operations, and REST APIs for Jira data extraction.
* **Existing RAG Framework:** For baseline vector similarity retrieval.

### 3.4 Evaluation Metrics

* **Mean Reciprocal Rank (MRR):** To measure the effectiveness of ranked retrieval results.
* **Precision @ k:** The proportion of relevant tickets in the top-k retrieved results.
* **User Feedback:** Qualitative evaluation of result relevance and transparency.

## Chapter 4: System Design and Architecture

### 4.1 System Overview

The proposed system integrates a Knowledge Graph with an existing Retrieval-Augmented Generation framework for enhanced Jira ticket retrieval. It introduces relationship-aware reasoning capabilities into an otherwise dense vector-based retrieval pipeline.

**Workflow Summary:**

* User queries are processed by both a vector similarity search engine and a Knowledge Graph filtering mechanism.
* The KG performs relationship-based filtering based on entity interconnections.
* Results from both components are merged using a ranking fusion strategy.
* Final results are presented along with reasoning paths traced through the KG.

### 4.2 System Architecture Diagram

+--------------------+  
| User Query |  
+--------------------+  
 |  
+--------+--------+  
| Vector Similarity |  
| Search |  
+--------+--------+  
 |  
+--------+--------+  
| KG-based Filtering|  
+--------+--------+  
 |  
 Result Fusion  
 |  
 Explainable Output

### 4.3 Component Design

* **Knowledge Graph:** Models entities (ticket, reporter, assignee, components) and relationships (reported\_by, assigned\_to, affects\_component) using Neo4j.
* **RAG Pipeline:** Computes vector embeddings for ticket summaries and queries, retrieves semantically similar tickets.
* **Hybrid Retrieval Module:** Merges KG-filtered and vector similarity results using score normalization and ranking fusion techniques.
* **Query Interface:** Allows users to construct multi-entity, relationship-based queries through a web interface.

## Directions for Future Work

* Expand KG coverage with multilingual and cross-project Jira tickets.
* Integrate explainability modules providing reasoning traces for each retrieved result.
* Implement continuous KG population pipelines for dynamic environments.
* Benchmark against pure vector-based and commercial cloud-based retrieval systems.
* Explore integration with CRM systems for end-to-end issue management.
* Develop on-premise deployment frameworks ensuring data security and enterprise compliance.

## References

1. Linders & Tomczak, (2025). Knowledge Graph-extended Retrieval Augmented Generation for Question Answering. *arXiv:2504.08893*
2. Sanmartin, (2024). KG-RAG: Bridging the Gap Between Knowledge and Creativity. *arXiv:2405.12035*
3. Xu et al., (2024). Retrieval-Augmented Generation with Knowledge Graphs for Customer Service. *arXiv:2404.17723*

*End of Mid Semester Report.*