Faculty of Artificial Intelligence Engineering
Department of Software Engineering
and Intelligent Information Systems

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

AI Powered Ticketing System

Senior2 Project- Completed the requirements for obtaining a bachelor's degree in Informatics Engineering - Software Engineering and Information Systems

Prepared by

Bayan Alallan Nuha Tinawi Mohammad Yasser Bazallah

Supervised By

Dr. Riad Sonbol

Eng. Anas Abdulaziz

January 2026

كلية هندسة الذكاء الاصطناعي

قسم هندسة البرمجيات ونظم المعلومات الذكية

نظام ذكي للمساعدة في إدارة طلبات العملاء

مشروع تخرج 2 – قدم استكمالا لمتطلبات الحصول على درجة البكالوريوس في هندسة المعلوماتية– هندسة البرمجيات ونظم المعلومات

إعداد

نهى تيناوي بيان العلان محمد ياسر بازالله

إشـراف

الدكتور رياض سنبل

المهندس أنس عبد العزيز

كانون الأول 2026

Supervisor certification

I certify that the preparation of this project entitled AI Powered Ticketing System, prepared by Nuha Tinawi, Bayan Alallan and Mohammad Yasser Bazallah was made under my supervision at Department of Software and Information Systems Engineering in partial of Bachelors of Degree the for Requirements the of fulfillment Software and Information Systems Engineering.

Name: Dr. Riad Sonbol Signature:……..…. Date: ……

Name: Eng. Anas Abdulaziz Signature:……..…. Date: ……

ACKNOWLEDGEMENTS

الحمدُللهِ الذي هيَّأَ البدء ويسرَ الطريق وطيّب المنتهى، الحمدُلله على التمام وحسْنُ الخِتام

في البداية نتقدّم **إلى الدكتور رياض سنبل** بخالص الشكر والامتنان على إشرافه المتواصل، وتوجيهه البنّاء، وحرصه الدائم على دفعنا نحو التفكير العميق والعمل المنهجي.

خلال رحلتنا في هذا المشروع، كنّا نلمس في كل مرحلة اهتمامه بالتفاصيل، وتشجيعه لنا على تطوير أنفسنا وقدراتنا.

نحن نقدّر القيمة العلمية الكبيرة التي اكتسبناها خلال هذه التجربة بفضله، ونعتبر تجربتنا تحت إشرافه من المحطات المهمة في مسيرتنا الأكاديمية.

إلى الدكتور مهيب النقري – عميد الكلية

نتوجّه إليكم بخالص الامتنان على دعمكم الدائم، واهتمامكم المستمر، وثقتكم التي كانت حافزًا لنا منذ بداية مشوارنا وحتى هذه المرحلة.

نقدّر الجهود الكبيرة التي بذلتموها في تهيئة بيئة تعليمية محفّزة، ونثمّن كل ما قدّمتموه من وقت وعمل في سبيل نجاح الطلبة وتقدّمهم.

نتقدّم بخالص الشكر والتقدير إلى **المهندس أنس عبد العزيز** على دوره في متابعة هذا المشروع، وما قدّمه من توجيه وتنظيم أسهم بشكل مباشر في حسن سير العمل.

شكرا على هذه المساهمة القيّمة التي كان لها أثر مباشر في نجاح المشروع.

Abstract

The proposed system provides a structured solution for managing customer requests and complaints in an organized and efficient manner. Traditional customer support systems often suffer from delays and inefficiencies in meeting customer needs, which can lead to decreased satisfaction levels. To address these challenges, the system offers a centralized environment that supports the creation, tracking, and resolution of support tickets throughout their lifecycle.

The system facilitates coordination between clients and support teams through role-based access and well-defined workflows, ensuring that requests are handled in a timely and consistent manner. It also enables administrators to configure system settings and monitor performance, while providing support teams with the necessary tools to manage and resolve issues effectively.

Overall, the system aims to improve the quality of customer support services by reducing response delays, enhancing operational efficiency, and increasing customer satisfaction through a clear and structured ticket management approach.

الملخص

يتناول هذا المشروع نظامًا آليًا ذكيًا لدعم إدارة طلبات العملاء وشكاويهم في المؤسسات والشركات. تعاني الأنظمة التقليدية لإدارة التذاكر من تأخير في الاستجابة وضعف في معالجة طلبات العملاء بكفاءة، مما يؤدي إلى انخفاض مستوى رضا المستخدمين. يهدف هذا المشروع إلى معالجة هذه المشكلة من خلال تقديم نظام منظم يساعد على تحسين كفاءة الاستجابة وتسريع معالجة الطلبات.

يوفر النظام بيئة مركزية لإدارة دورة حياة التذكرة بدءًا من إنشائها وحتى إغلاقها، مع دعم إدارة المستخدمين، فرق الدعم، وتخصيص الأدوار بما يضمن توزيع المهام بشكل مناسب. كما يساهم النظام في تنظيم سير العمل وتحسين التنسيق بين الأطراف المختلفة، مما يساعد فرق الدعم على التعامل مع الطلبات بطريقة أكثر فعالية ودقة.

يساهم هذا المشروع في تحسين جودة خدمات الدعم الفني من خلال تقليل زمن المعالجة، رفع كفاءة العمل، وتعزيز تجربة العميل، ويُعد خطوة نحو تطوير أنظمة خدمة العملاء وتحويلها من أساليب تقليدية إلى حلول أكثر تنظيمًا وذكاءً تعتمد على تحليل البيانات واتخاذ القرارات الداعمة للاستجابة السريعة.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of abbreviations

Abbreviation	Definition
API	Application Program Interface
NLP	Natural Language Processing
AI	Artificial Intelligence
SQL	Structured Query Language
SOW	Statement of Work
SRS	Software requirement specification
STM	Support team manager
STMe	Support team member
UML	Unified Modeling Language
SOW	Statement of Work
RTM	Requirements traceability matrix
RAG	Retrieval-Augmented Generation

Table 1 list of abbreviations

Chapter 1 Introduction

1. Introduction

This chapter introduces the core elements of the AI-Powered Ticketing System project, laying the foundation for understanding the problem it addresses, the project goals, the proposed system, the organization of the report, and a brief summary of key points.

2. Problem Statement

The project addresses the existing limitations in customer support systems, where traditional models often struggle with delayed responses, inefficient ticket handling, and limited ability to accurately assess customer needs. These challenges can lead to poor coordination between support teams, inconsistent request handling, and ultimately reduced customer satisfaction.

Such limitations highlight the need for a more structured and intelligent solution that enhances the management of customer requests and complaints. This project proposes an automated support system that improves response efficiency, organizes ticket processing, and supports effective management of users and support teams. By enhancing the overall handling of support tickets and enabling better prioritization of requests, the system aims to provide timely responses and improve the quality of customer support services.

3. Project Objective

The main objective of this project is to develop a streamlined and efficient ticketing system that enhances customer support services through structured automation and intelligent request handling. The project aims to improve response efficiency, organize ticket processing, and support effective management of users and support teams, ensuring that customer requests are handled in a timely and consistent manner.

The scope of the project focuses on designing and implementing an integrated system that includes user management, team management, ticket management, and request prioritization mechanisms. The system is designed to support role-based access and flexible workflows that adapt to organizational needs. An analytical methodology is adopted to define system requirements and manage data effectively, ensuring clarity, reliability, and scalability of the proposed solution.

4. Proposed System

The The AI-Powered Ticketing System is designed as an automated and intelligent solution for managing customer support requests and complaints. The system aims to improve the efficiency of support operations by organizing the ticket lifecycle and facilitating structured interaction between clients and support teams. Traditional limitations in ticket handling are addressed through automation and intelligent analysis of customer requests.

The system leverages artificial intelligence techniques to enhance ticket processing, including sentiment analysis to assess the tone of customer messages and support the prioritization of urgent requests. In addition, the system supports automated ticket categorization and assignment to appropriate support teams, contributing to faster and more accurate request handling. These intelligent features are configurable, allowing flexibility in how automation is applied within different organizational contexts.

Furthermore, the system provides role-based management tools for users and support teams, ensuring clear responsibility distribution and controlled access to system functionalities. By combining intelligent ticket analysis with structured workflow and user management, the proposed system enables more effective response handling and improved coordination across the organization.

5. Report Organization

The report is structured as follows:

Chapter 1: Introduction

Chapter 2: Basic Concepts and Reference Study

Chapter 3: Project Management

Chapter 4: System Analysis

Chapter 5: System Design

Chapter 6: Artificial Intelligence Design and Techniques

Chapter 7: Practical Implementation

Chapter 8: Report Overview

6. **Summary**

This chapter has outlined the need for an AI-driven ticketing system, explained the objectives of the project, introduced the proposed solution at a high level, and provided an overview of the report's structure. This introduction sets the stage for a deeper exploration of each aspect of the project in the following chapters.

Chapter 2 Fundamental Concepts and Literature Review

1. Introduction

This chapter provides a foundation for understanding the AI-Powered Ticketing System by exploring fundamental concepts and reviewing related studies. It examines essential terminology, key theories, and the principles behind the technologies used in the system. The aim is to present an overview of AI applications in ticketing systems and support management, along with insights from existing solutions to guide and support the development of the proposed system.

2. Fundamental Concepts

This section presents the fundamental concepts, terms, and theoretical principles that form the foundation of the proposed system. These concepts provide the necessary background for understanding how intelligent ticketing systems operate and support customer service processes. including:

- **Artificial Intelligence (AI) and Machine Learning (ML):** This subsection introduces the basic principles of artificial intelligence and machine learning and their role in enabling automation and intelligent decision-making within customer support systems. It discusses key concepts such as supervised learning, natural language processing (NLP), and data-driven decision models, which support automated analysis and handling of customer requests.

- **Natural Language Processing (NLP):** This subsection explains natural language processing and its importance in ticketing systems for processing, understanding, and organizing text-based customer inquiries. Core NLP concepts such as tokenization, language models, and entity recognition are discussed to illustrate how textual data can be analyzed to support accurate ticket classification and routing.

- **Sentiment Analysis:** Discussion on sentiment analysis as a specialized area of NLP that focuses on identifying the emotional tone of customer messages. Understanding customer sentiment plays an important role in support systems

by helping distinguish urgent or critical requests from routine inquiries, thereby contributing to more effective prioritization and response handling.

- **Role-Based Access Control (RBAC):** An introduction to RBAC and its importance in system security, allowing for controlled access based on user roles, such as Admin, Client, and Support Team, to ensure data security and proper workflow management.

- **Ticket Management and Workflow Automation:** Explanation of the ticket lifecycle, from creation to resolution, and the role of automation in ticket assignment, prioritization, and status tracking. This concept covers how workflow automation improves efficiency by routing tickets based on predefined rules and priorities.

3. Literature Review for the system

The purpose of this Literature Review is to analyze and evaluate existing systems and methodologies related to the development of AI-powered ticketing platforms. This study aims to provide a comprehensive overview of the current landscape in technical support and ticketing solutions, focusing on how artificial intelligence, machine learning, and automation are utilized to enhance efficiency and customer satisfaction. The analysis includes comparisons between four systems and their key features:

- **Zendesk**

  Zendesk is a customer service software designed to improve customer interactions and relationships through a streamlined ticketing system, making it popular in various industries

  Advantages:

  1) **Multi-Channel Support:** Integrates easily across email, chat, social media, phone, and web to manage customer inquiries.

2) **Scalability:** Suitable for businesses of all sizes, from startups to large enterprises.
3) **Easy to Use:** Intuitive interface for both agents and customers.
4) **Strong Reporting Tools:** Advanced reporting and analytics features.
5) **AI & Automation:** Built-in AI (Answer Bot) for automating responses and ticket categorization.
6) **Robust Integration Ecosystem:** Integrates with many third-party apps like Salesforce, Shopify, etc.

Disadvantages:

1) **Costly for Advanced Features**: The more advanced functionalities (automation, analytics, etc.) are available only in higher-priced plans.
2) **Limited Customization**: Customization can be somewhat limited compared to other platforms.
3) **Slow Load Time:** Some users report slower load times when dealing with a large volume of tickets.
4) **Complex Setup**: Can require more time and technical knowledge to set up compared to simpler systems.

Main Features:

- Ticketing System: Centralized system for managing customer inquiries from multiple channels.
- AI-Powered Bots: Answer Bot to handle common queries automatically.
- Knowledge Base: Create FAQs and self-help resources for users.
- Reporting & Analytics: Detailed reports on customer interactions and agent performance.
- Multi-Language Support: Supports different languages for global teams.
- Collaboration Tools: Allows teams to collaborate efficiently on complex tickets.

- NLP-Driven Ticket Processing: Utilizes Natural Language Processing to interpret and categorize incoming tickets, enabling more accurate routing and faster resolution times.
- Sentiment Analysis: Analyzes customer communications to determine sentiment, helping agents prioritize and tailor responses based on customer emotions.

- **Freshdesk**

  Freshdesk is a cloud-based customer support platform designed for managing customer conversations across multiple channels and automating repetitive tasks.

Advantages:

1) **Affordable Pricing**: Competitive pricing for businesses of all sizes.
2) **Freddy AI:** Built-in AI helps with ticket categorization and prioritization.
3) **Multi-Channel Support:** Supports email, chat, phone, social media, and more.
4) **User-Friendly Interface:** Intuitive for both agents and customers, requiring minimal training.
5) **Customizable Workflows:** Allows businesses to set up workflows and automation for repetitive tasks.
6) **Multi-Language Support:** Allows teams to support customers across different regions.

Disadvantages:

1) **Limited Advanced Features:** Some advanced features (like deep analytics) are only available in higher tiers.
2) **Scalability Issues:** May face some performance issues when scaling to larger organizations.
3) **Customer Service:** Some users report slower response times for support tickets related to the Freshdesk platform itself.
4) **Basic SLA Management:** May not be robust enough for IT-specific needs.
5) **Customization Limitations:** Not as flexible in terms of deep customization compared to other platforms.

Main Features:

- **Ticketing System:** Omnichannel support for managing tickets from multiple sources.
- **Freddy AI:** AI-driven automation for ticket management and customer support.
- **Knowledge Base:** Offers a self-service portal for common queries.
- **Collaboration Tools:** Helps agents work together on tickets with internal notes and team huddles.
- **Reporting & Analytics:** Provides reports on customer interactions, agent performance, and more.
- **Integration:** Supports integration with popular tools like Slack, Shopify, and Google Analytics.
- **Automated Workflow Routing:** How It Works: Automates ticket assignment based on predefined rules, such as agent skills, workload, and ticket priority, ensuring efficient distribution.
- **NLP-Driven Ticket Processing:** Uses NLP to analyze ticket content, categorize issues, and route them to the appropriate teams, enhancing accuracy in ticket handling
- **Customizable to Business Needs:** Allows customization of workflows, ticket fields, automation rules, and integrations to align with specific business processes.
- **Sentiment Analysis:** Analyzes customer communications to determine sentiment, helping agents prioritize and tailor responses based on customer emotions
- **Pre-Populated Fields:** AI can automatically fill in ticket details like issue type, category, priority, and more based on the description alone.

- **Zoho Desk**

Zoho Desk is a customer support platform from Zoho, aimed at businesses of all sizes, especially those already using other Zoho products.

Advantages:

1) **Seamless Zoho Integration:** Works well with the Zoho ecosystem (CRM, Zoho Projects, etc.).
2) **Zia AI:** Built-in AI (Zia) for automating responses and ticket routing.
3) **Affordable Pricing:** Competitive pricing, especially for businesses already using Zoho tools.
4) **User-Friendly Interface**: Easy to navigate for agents and customers alike.
5) **Customizable**: Offers a good range of customization options for workflows and ticket handling.
6) **Multi-Channel Support:** Supports email, phone, chat, social media, and web forms.

Disadvantages:

1) **Limited Third-Party Integrations**: Compared to Zendesk and Freshdesk, Zoho Desk may have fewer third-party integrations.
2) **Limited Features in Basic Plans**: Advanced features like AI automation are only available in higher-tier plans.
3) **Less Popular for ITSM**: While good for customer support, it's not as focused on IT service management as ServiceNow.
4) **Reporting**: Reporting is good but not as comprehensive as some enterprise-level tools like ServiceNow.
5) **Scaling:** May face limitations when scaling to very large enterprises.

Main Features:

- **Ticketing System**: Handles customer queries across multiple channels.
- **Zia AI:** AI-driven features for automating responses, categorizing tickets, and routing to the right team.
- **Knowledge Base:** Self-service portal for common customer queries.
- **Reporting & Analytics:** Good range of reports, though not as detailed as enterprise-grade solutions.
- **Collaboration Tools**: Agents can collaborate on tickets for faster resolution.
- **Integration with Zoho Suite**: Seamless integration with Zoho's CRM, projects, and other tools**.**
- **Automated Workflow Routing:** Utilizes automation rules to assign tickets to appropriate agents or departments based on criteria like ticket type, priority, and agent skill set
- **NLP-Driven Ticket Processing:** Applies NLP to interpret and categorize tickets automatically, improving the accuracy of ticket routing and reducing manual intervention.
- **Sentiment Analysis:** Analyzes customer messages to determine sentiment, enabling agents to prioritize and customize their responses based on customer emotions.

- **HubSpot Service Hub**

  HubSpot Service Hub is a part of the HubSpot ecosystem, designed to enhance customer service operations. It integrates seamlessly with HubSpot's CRM, providing a unified platform for managing customer interactions and support tickets.

Advantages:

1) Seamless CRM Integration: Integrates effortlessly with HubSpot CRM, providing a unified view of customer interactions.
2) User-Friendly Interface: Intuitive and easy to navigate for both agents and customers.
3) Automation Tools: Offers powerful automation features for ticket routing, follow-ups, and more.
4) Knowledge Base: Robust self-service options with a customizable knowledge base.
5) Multi-Channel Support: Supports email, live chat, and conversational bots.
6) Comprehensive Reporting: Detailed analytics and reporting to track performance and customer satisfaction.
7) Scalability: Suitable for businesses of all sizes, from startups to large enterprises.

Disadvantages:

1) Cost: Can become expensive as you scale and add more features.
2) Limited Customization: Some limitations in customizing workflows compared to specialized ticketing systems.
3) Learning Curve: May require training for teams unfamiliar with HubSpot's ecosystem.
4) Integration Dependencies: Best performance when fully integrated with other HubSpot tools, which may not suit all businesses.
5) Feature Limitations in Lower Tiers: Advanced features are often locked behind higher-priced plans.

Main Features:

- **Ticketing System**: Centralized platform for managing and tracking customer support tickets.
- **Automation & Workflows:** Automate repetitive tasks, ticket routing, and follow-ups.
- **Knowledge Base**: Create and manage a self-service portal for customers.
- **Live Chat & Bots**: Real-time communication with customers through live chat and AI-powered bots.
- **Customer Feedback**: Collect and analyze customer feedback to improve service.
- **Reporting & Analytics**: Comprehensive dashboards and reports to monitor support performance.
- **Integration with HubSpot CRM**: Unified customer data across sales, marketing, and service teams.

In conclusion, the AI Powered Ticketing System (APTS) provides a cutting-edge solution for streamlining technical support services. By integrating advanced AI features such as Sentiment Analysis, Natural Language Processing (NLP), and automated ticket routing, APTS enhances the efficiency and accuracy of ticket management. Its intuitive interface and comprehensive user, team, and account management systems ensure seamless operations for both customers and support agents. As APTS continues to evolve, future enhancements like NLP-driven ticket processing and AI-powered chatbots promise to further improve response times and customer satisfaction, making it a vital tool for modern technical support teams.

Here's a summary of the standout service for each of the 4 Ticketing System mentioned

Table 2 Comparative Analysis of Similar Systems and the Target System

System Feature	Zendesk	Freshdesk	Zoho Desk	HubSpot Service Hub	Our system
Customer Feedback		✓		✓	✓
AI Powered Chatbot	✓	✓	✓		✓
workflow customization	✓	✓	✓	✓	✓
categories Ticket	✓	✓	✓		✓
Integration with communication channels	✓	✓	✓	✓	✓ *
Analytics and reporting	✓	✓	✓		✓
priorities Ticket by sentiment analysis	✓	✓	✓		✓
Support for Notification	✓	✓	✓	✓	✓
User-Friendly interface		✓	✓		✓
Pre-Populated Fields					✓ *
Knowledge Base			✓	✓	
Multi-Language		✓	✓		✓ *
(*) means features that will be achieved in the future, outside the scope of this semester.					

4. A Comparative Literature Review on Machine Learning Approaches for Intelligent Ticket Classification and Auto-Assignment in Helpdesk Systems

Academic studies comparing machine learning approaches for ticket classification and automatic team assignment found that support vector machines and BERT-based models achieved the strongest performance, with SVMs reaching 94.5% to 98% accuracy and BERT models achieving up to 95.2% accuracy on text-based ticket data.

1. Abstract

Giwa et al. (2025) state that a unified model based on BERT achieved 89.4% accuracy, 88.7% precision, 90.2% recall, and an F1 of 89.4% on customer support tickets. Shah (2020) reports that an artificial neural network attained 95% accuracy in a multilingual ticketing system, while Paramesh and Shreedhara (2018) and Powell et al. (2020) note that support vector machines performed robustly—with Powell et al. recording 94.5% to 98% accuracy—across both classification and routing tasks.

Zangari et al. (2023) indicate that a hierarchical, Multi-Label BERT model improved F1 and overall accuracy by roughly 5 percentage points over a baseline. Feng et al. (2022) describe an ensemble driven by a Robustly Optimized BERT Pretraining Approach that reached 95.2% top-3 accuracy (and 79% top-5 for resolver assignment) on a dataset of over 200,000 tickets. In several studies, support vector machines emerged as the most frequently best-performing approach, with transformer-based models also excelling when dataset size, noise filtering, and hierarchical labels were taken into account.

Taken together, papers on ticket classification and automatic team assignment agree that support vector machines and BERT-based models yield the strongest performance on text-based ticket data.

2. Review of Studies:

We analyzed 10 sources, using 8 screening criteria. Each paper was reviewed for 6 key aspects that mattered most to the research.

3. Results

Characteristics of Included Studies

Study	Study Focus (Classification/ Routing/Both)	Algorithms Evaluated	Dataset Size	Primary Evaluation Metrics
Giwa et al., 2025	Both (Unified)	Bidirectional Encoder Representations from Transformers (BERT), Graph Neural Network (GNN), Prototypical Networks, Random Forest	We didn't find mention	Accuracy, Precision, Recall, F1-score
Shah, 2020	Both (Likely Unified)	K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, Artificial Neural Network (ANN), Bidirectional Long Short-Term Memory (BLSTM)	We didn't find mention	Accuracy, Confidence Interval
Paramesh and Shreedhara, 2018	Both (Unified)	Multinomial Naive Bayes, Logistic Regression, K-Nearest Neighbors, Support Vector Machine	We didn't find mention	We didn't find mention
Zangari et al., 2023	Classification only	Bidirectional Encoder Representations from Transformers (BERT), Multi-Label BERT (ML-BERT, hierarchical)	20,000/35,000 tickets	F1-score, Accuracy
Powell et al., 2020	Both (Unified)	Support Vector Machine (ensemble), Artificial Neural Network, Term Frequency-Inverse Document Frequency (TF-IDF)	~17,000 tickets	Accuracy, Error Rate, Confidence Interval
Feng et al., 2022	Both (Unified, multi-task)	Bidirectional Encoder Representations from Transformers (BERT), Robustly Optimized BERT Pretraining Approach (RoBERTa), DistilBERT, DistilRoBERTa, Ensemble, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)	203,300 tickets	Top-K Accuracy
Wibowo and Hariyanto, 2023	Both (Unified)	Naive Bayes, Support Vector Machine	1,327 tickets (post-cleaning)	Accuracy, Precision, Recall, Confidence Interval
Marcuzzo et al., 2022	Classification (Hierarchical)	Bidirectional Encoder Representations from Transformers (BERT), Support Vector Machine, Multi-level classifier	We didn't find mention	Classification Accuracy
Zicari et al., 2021	Classification only	Heterogeneous deep learning ensemble	We didn't find mention	We didn't find mention
Han et al., 2020	Routing only (Unified)	Unified Framework for Ticket Routing (UFTR, custom ranking models)	500,000 tickets	We didn't find mention

Study Focus:

- **Unified approaches:** Four studies focused on both classification and routing in a unified approach, and one used a unified, multi-task approach.
- **Classification only:** Three studies focused solely on classification.
- **Routing only:** One study focused solely on routing.
- **Ambiguous**: One study likely used a unified approach but did not state this explicitly.

Algorithms Evaluated:

- **Support Vector Machine (SVM):** Evaluated in five studies.
- **Bidirectional Encoder Representations from Transformers (BERT):** Evaluated in four studies.
- **Random Forest, K-Nearest Neighbors, Artificial Neural Network, Naive Bayes:** Each evaluated in two studies.
- **Other algorithms:** Graph Neural Network, Prototypical Networks, Bidirectional Long Short-Term Memory, Logistic Regression, Multi-Label BERT, Term Frequency-Inverse Document Frequency, Robustly Optimized BERT Pretraining Approach, DistilBERT, DistilRoBERTa, Ensemble, Hierarchical Density-Based Spatial Clustering of Applications with Noise, Multi-level classifier, Unified Framework for Ticket Routing—each evaluated in one study.

Dataset Size:

- **Reported:** Five studies reported dataset sizes, ranging from 1,327 to 500,000 tickets.
- **Not reported:** Five studies did not mention dataset size.

Primary Evaluation Metrics:

- **Accuracy:** Most commonly used (five studies).
- **Confidence Interval:** Reported in three studies.
- **F1-score, Precision, Recall:** Each reported in two studies.
- **Error Rate, Top-K Accuracy, Classification Accuracy:** Each reported in one study.
- **No mention:** Three studies did not mention primary evaluation metrics.

4. Effects

Ticket Classification Performance

Study	Algorithm	Best Reported Performance	Evaluation Metric	Dataset Context
Giwa et al., 2025	Bidirectional Encoder Representations from Transformers (BERT)	89.4% accuracy, 88.7% precision, 90.2% recall, 89.4% F1	Accuracy, Precision, Recall, F1	Customer support, size not mentioned in the abstract
Shah, 2020	Artificial Neural Network	95% accuracy (Confidence Interval: 93.6–95.1%)	Accuracy, Confidence Interval	Multilingual ticketing, size not mentioned in the abstract
Paramesh and Shreedhara, 2018	Support Vector Machine	"Performed well" (no metric)	We didn't find mention in the abstract	IT service desk, size not mentioned in the abstract
Zangari et al., 2023	Multi-Label BERT	+5.7% F1, +5.4% accuracy over baseline	F1-score, Accuracy	20,000/35,000 tickets, hierarchical labels
Powell et al., 2020	Support Vector Machine	94.5% accuracy (fields), 98% (routing fields)	Accuracy	U.S. Navy, ~17,000 tickets
Feng et al., 2022	Robustly Optimized BERT Pretraining Approach (ensemble)	95.2% top-3 accuracy (group)	Top-3 Accuracy	203,300 tickets, customer support
Wibowo and Hariyanto, 2023	Support Vector Machine	85.3% accuracy, 84.3% precision, 89.5% recall	Accuracy, Precision, Recall	1,327 tickets, IT helpdesk
Marcuzzo et al., 2022	Multi-level classifier	We didn't find mention in the abstract	Classification Accuracy	Software bugs, size not mentioned in the abstract
Zicari et al., 2021	Deep learning ensemble	We didn't find mention in the abstract	We didn't find mention in the abstract	Public ticket-mining, size not mentioned in the abstract
Han et al., 2020	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract	500,000 tickets, customer service

Algorithm types:

- **Support Vector Machine**: Reported as best in three studies.
- **Transformer-based models (BERT, Multi-Label BERT, Robustly Optimized BERT Pretraining Approach):** Reported as best in three studies.
- **Artificial Neural Network**: Reported as best in one study.
- **Multi-level classifier, deep learning ensemble:** Each reported as best in one study.
- **No mention:** One study did not mention the best algorithm.

Performance metrics:

- **Accuracy:** Reported in five studies.
- **F1-score:** Reported in two studies.
- **Precision and recall:** Each reported in two studies.
- **Top-3 accuracy:** Reported in one study.
- **Confidence intervals:** Reported in one study.
- **No mention:** Four studies did not mention performance metrics.

Dataset context:

- **Customer support/service/helpdesk:** Four studies.
- **IT service desk/helpdesk:** Two studies.
- **Multilingual ticketing:** One study.
- **U.S. Navy ticket data:** One study.
- **Hierarchical labels:** One study.
- **Software bug data:** One study.
- **Public ticket-mining:** One study.

Dataset size:

- **Reported:** Five studies (1,327 to 500,000 tickets).
- **Not reported:** Five studies.

Automatic Team Assignment Performance

Study	Algorithm	Best Reported Performance	Evaluation Metric	Dataset Context
Giwa et al., 2025	Bidirectional Encoder Representations from Transformers (BERT)	89.4% accuracy (unified with classification)	Accuracy	Customer support.
Shah, 2020	Artificial Neural Network	95% accuracy (unified with classification)	Accuracy	Multilingual ticketing.
Paramesh and Shreedhara, 2018	Support Vector Machine	"Performed well" (no metric)	We didn't find mention in the abstract	IT service desk
Powell et al., 2020	Support Vector Machine	98% accuracy (routing fields)	Accuracy	U.S. Navy
Feng et al., 2022	Robustly Optimized BERT Pretraining Approach (ensemble)	95.2% top-3 accuracy (group), 79% top-5 (resolver)	Top-K Accuracy	203,300 tickets
Wibowo and Hariyanto, 2023	Support Vector Machine	85.3% accuracy	Accuracy	1,327 tickets
Han et al., 2020	Unified Framework for Ticket Routing	Outperformed baselines (no metric)	Routing metrics	500,000 tickets
Zangari et al., 2023	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract
Marcuzzo et al., 2022	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract
Zicari et al., 2021	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract	We didn't find mention in the abstract

Algorithm types:

Support Vector Machine: Used in three studies.

Transformer-based models (BERT, Robustly Optimized BERT Pretraining Approach): Used in two studies.

Artificial Neural Network: Used in one study.

Unified Framework for Ticket Routing: Used in one study.

No mention: Three studies did not mention an applicable algorithm.

Performance and evaluation metrics:

Accuracy: Main metric in four studies, with values from 85.3% to 98%.

Top-K accuracy: Reported in one study (95.2% top-3, 79% top-5).

Routing metrics: Reported in one study.

No mention: Three studies did not mention a performance metric.

Dataset context:

Specified: Seven studies.

No mention: Three studies.

5. Overview of Datasets Used in Ticket Classification and Routing Studies

Study	Dataset Name	Source Type	Composition	Dataset Size	Language
Giwa et al., 2025	Not mentioned	Customer support	Not detailed	Not mentioned	Not mentioned
Shah, 2020	Not mentioned	Ticketing system	Not detailed	Not mentioned	Multiple languages (translated to English)
Paramesh and Shreedhara, 2018	Not mentioned	IT infrastructure service desk	Not detailed	Not mentioned	Not mentioned
Zangari et al., 2023	Not mentioned	Customer support; IT bug reporting	Customer complaints; bug reports	20,000/35,000 tickets	Not mentioned
Powell et al., 2020	Not mentioned	U.S. Navy IT support	Support requests	~17,000 tickets	English
Feng et al., 2022	Not mentioned	Customer support/helpdesk	Ticket descriptions, group/resolver assignments	203,300 tickets	English
Wibowo and Hariyanto, 2023	Not mentioned	IT helpdesk	Ticket data with attributes	1,327 tickets (post-cleaning)	English
Marcuzzo et al., 2022	Not mentioned	IT (software bugs)	Software bug reports	Not mentioned	Not mentioned
Zicari et al., 2021	Not mentioned	Public ticket-mining	Not detailed	Not mentioned	Not mentioned
Han et al., 2020	Not mentioned	Customer service	Archived tickets	500,000 tickets	Not mentioned

Cross-Task Algorithm Comparison

Study	Classification Best	Routing Best	Notes
Giwa et al., 2025	Bidirectional Encoder Representations from Transformers (BERT)	Bidirectional Encoder Representations from Transformers (BERT)	Unified task; BERT outperforms Graph Neural Network, Prototypical Networks, Random Forest
Shah, 2020	Artificial Neural Network	Artificial Neural Network	Unified task; only Artificial Neural Network metrics reported
Paramesh and Shreedhara, 2018	Support Vector Machine	Support Vector Machine	Unified task; Support Vector Machine robust to noisy data
Powell et al., 2020	Support Vector Machine	Support Vector Machine	Support Vector Machine competitive with deep learning; real-world deployment
Feng et al., 2022	Robustly Optimized BERT Pretraining Approach/ensemble	Robustly Optimized BERT Pretraining Approach/ensemble	Ensemble improves resolver assignment; practical for real-time
Wibowo and Hariyanto, 2023	Support Vector Machine	Support Vector Machine	Support Vector Machine outperforms Naive Bayes; needs more data for optimality
Zangari et al., 2023	Multi-Label BERT	We didn't find mention in the abstract	Hierarchical info boosts classification
Marcuzzo et al., 2022	Multi-level classifier	We didn't find mention in the abstract	Hierarchical/multi-level approach effective
Zicari et al., 2021	Deep learning ensemble	We didn't find mention in the abstract	Ensemble effective for categorization
Han et al., 2020	We didn't find mention in the abstract	Unified Framework for Ticket Routing	Unified Framework for Ticket Routing outperforms baselines for routing

Best-performing models for classification tasks (as reported by the papers):

- **Support Vector Machine:** Three studies.
- **Bidirectional Encoder Representations from Transformers, Artificial Neural Network, Robustly Optimized BERT Pretraining Approach/ensemble, Multi-Label BERT, multi-level classifier, deep learning ensemble**: Each reported as best in one study.
- **No mention:** One study did not mention the best classification model.

Best-performing models for routing tasks (as reported by the papers):

- **Support Vector Machine:** Three studies.
- **Bidirectional Encoder Representations from Transformers, Artificial Neural Network, Robustly Optimized BERT Pretraining Approach/ensemble, Unified Framework for Ticket Routing:** Each reported as best in one study.
- **No mention:** Three studies did not mention routing results.

Discussion and Model Selection Justification

Across the reviewed studies, both Support Vector Machine (SVM) and BERT-based models were consistently reported as strong performers in ticket classification and routing tasks. While SVM was the most frequently identified best-performing model across multiple studies, transformer-based models also demonstrated competitive performance, particularly in scenarios involving complex textual data and contextual understanding.

To further evaluate these findings within the scope of this project, an experimental comparison was conducted by training both SVM and BERT models on a publicly available ticket dataset obtained from Kaggle. The experimental results indicated that the BERT-based model achieved superior performance in understanding ticket context and classification accuracy compared to SVM. Based on these observations, BERT was selected as the primary model for ticket categorization in the proposed system, as it better aligns with the system's requirement for accurate and context-aware ticket analysis.

Chapter 3 Project Management

1. Introduction:

In this chapter, we will dive into the management phase of the project, which is a critical aspect of ensuring the project's success. We will examine the project charter, project plan, Statement of Work (SOW) document, stakeholder analysis, and risk management strategies to effectively guide and control the project from initiation to completion.

2. Project charter

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision-making and project governance.

- **Project Title:** AI Powered Ticketing System
- **Project Start Date:** October 1, 2024
- **Projected Finish Date:** December 30, 2025
- **Project Manager:** Dr. Riad Sonbol and Eng. Anas Abdulaziz.
- **Project Objectives:**
  - Develop a flexible and scalable system for managing customer requests and complaints that can integrate with various software platforms and websites.
  - Enable customizable ticket creation, routing, and workflow management based on the specific operational needs of each organization.
  - Incorporate artificial intelligence and natural language processing techniques to support efficient analysis and classification of customer request texts.
  - Enhance workflow automation and support processes to improve response times, coordination between support teams, and overall service quality.
- **Approach:**
  - Conduct a needs assessment and gather functional and non-functional requirements from users and stakeholders.
  - Design a structured system architecture that supports ticket management, workflow control, user authentication, and intelligent request handling.

- ❖ Utilize internal development teams for project planning, system development, integration, and testing.
- ❖ Implement and evaluate AI-supported features that assist in ticket analysis and resolution, ensuring alignment with system objectives and user requirements.

- **Roles and Responsibilities:**

Name	**Role**	**Responsibility**
Dr. Riad Sonbol	Project Manager &Sponsor	Provide strategic direction, oversee project funding and resources, resolve high-level issues, and ensure the project meets organizational objectives.
Eng. Anas Abdulaziz	Project Manager	Manage day-to-day project execution, coordinate with team members, track progress, manage risks, and report on project status to Dr. Riad Sonbol.
Mohammad Yasser Bazallah	AI developer & Backend [Django]	Develop and integrate artificial intelligence components within the system, including text analysis and intelligent ticket processing. Contribute to backend development using Django, support API integration between system modules, and assist in implementing data-driven decision-making features.
Bayan Alallan	SE & Backend [Django]	Collaborate on backend system development, manage database design and integrations, develop and maintain RESTful APIs, and ensure data consistency, security, and reliable system performance.
Nuha Tinawi	System analyst & Frontend [React]	Analyze system requirements and translate them into functional designs, design and implement user interfaces, ensure a smooth and user-friendly experience, and integrate frontend components with backend APIs while maintaining usability and consistency.

Table3 Roles and Responsibilities

3. The SOW document:

Statement of Work is a comprehensive document that defines the scope of work for a project. It outlines the specific tasks, deliverables, timeline, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project's objectives, and the criteria for success.

1. Project Description and Objectives:

The project aims to develop an AI-powered ticketing system that facilitates the management of customer support tickets in an organized and efficient manner. The system assists support teams in prioritizing, managing, and resolving tickets more effectively, while contributing to improved user satisfaction through structured and intelligent support processes.

2. Project Scope:

The AI-Powered Ticketing System focuses on automating and enhancing the management of customer requests and complaints. The project scope includes the use of artificial intelligence techniques to support ticket analysis, prioritization, and organization, with the primary goal of improving response times, workflow efficiency, and overall customer satisfaction.

3. Project Goals:

- Develop a software system that simplifies ticket management for support teams and managers.
- Utilize artificial intelligence algorithms to provide automated insights, ticket prioritization, and sentiment analysis.
- Enable a user-friendly interface for submitting and tracking support tickets.
- Provide dashboards and reporting tools for monitoring ticket performance and evaluating support team efficiency.

Deliverables:

- Project plan.
- SRS documentation.
- Final project report.
- Functional AI algorithms for ticket analysis and sentiment detection.
- Backend and frontend system components forming the complete AI-powered ticketing system.

4. Project Requirements:

Technology and Tools:

- Programming Languages: Python, JavaScript, HTML, CSS.
- Frameworks: Django (backend), React (frontend).
- Database: MySQL.

5. Assumptions:

- Regular availability of project team members and stakeholders for feedback.
- Continuous feedback and supervision from the project manager.
- Hardware and software resources required for development and deployment.

6. Project Resources:

Human Resources:

- Dr. Riad Sonbol: Project Manager
- Eng. Anas-Abdulaziz: Project Manager
- Mohammad Yasser Bazallah: AI developer & Backend developer [Django]
- Bayan Alallan: Backend Developer (Django)
- Nuha Tinawi: System analyst & Frontend Developer (React)

7. Schedule:

- ❖ Project Start Date: October 1, 2024
- ❖ Projected Finish Date: December 30, 2025

4. Risk management

Risk Title	Risk Description	Raised Date	Tracking Frequency	State	Impact	Mitigation Plan
Team of only two students	If one of the students stops working for some reason, the whole project progress will be impacted.	24/4/2025	Weekly	Active	High	The one who finishes early will assist the other with their tasks.
Fuzzy project scope	Misunderstanding the project scope at the beginning of the work can lead to major mistakes in implementation.	24/4/2025	Daily	Closed	Medium	Conduct regular team discussions and meetings at the end of each milestone to clarify objectives.
Learning new technologies	Using AI components with limited knowledge can create challenges in development.	15/4/2025	Weekly	Under Mitigation	High	Allocate dedicated learning sessions and use online resources to build understanding.
Separation of work	One member is responsible for the frontend and the other for the backend, which can cause coordination gaps.	1/4/2025	Daily	Active	Medium	Regular integration testing and progress sync-ups to ensure compatibility between backend and frontend.
Integration issues	Problems may arise when integrating AI components with other system modules due to lack of interoperability experience.	12/4/2025	Weekly	Active	High	Test small integrations early and maintain proper documentation for module interfaces.

Risk Title	Risk Description	Raised Date	Tracking Frequency	State	Impact	Mitigation Plan
Deadline pressure	Tight deadlines may lead to shortcuts in development, causing potential quality issues.	30/5/2025	Weekly	Active	Medium	Break tasks into smaller deliverables and prioritize critical functionality to ensure quality work within deadlines.
Lack of testing expertise	Limited experience with testing AI models could result in poorly validated functionality or errors.	20/4/2025	Weekly	Active	High	Assign specific time for testing and involve external peers to review testing plans.
Requirement changes	Late changes in requirements from instructors or stakeholders can disrupt the project timeline and scope.	18/3/2025	Weekly	Active	High	Allocate buffer time in the schedule and ensure effective communication with stakeholders to manage changes.
Deployment challenges	Lack of experience with deployment might delay the final delivery of the project.	25/4/2025	Weekly	Active	High	Research deployment best practices and run a mock deployment to identify potential blockers early.
Sentiment analysis accuracy	Challenges in achieving acceptable sentiment analysis accuracy due to limitations in AI training data or inappropriate model selection.	2/4/2025	Weekly	Active	High	Train the sentiment analysis model with a diverse dataset and validate the performance regularly with test cases.

Table4 Risk management

5. Gantt chart

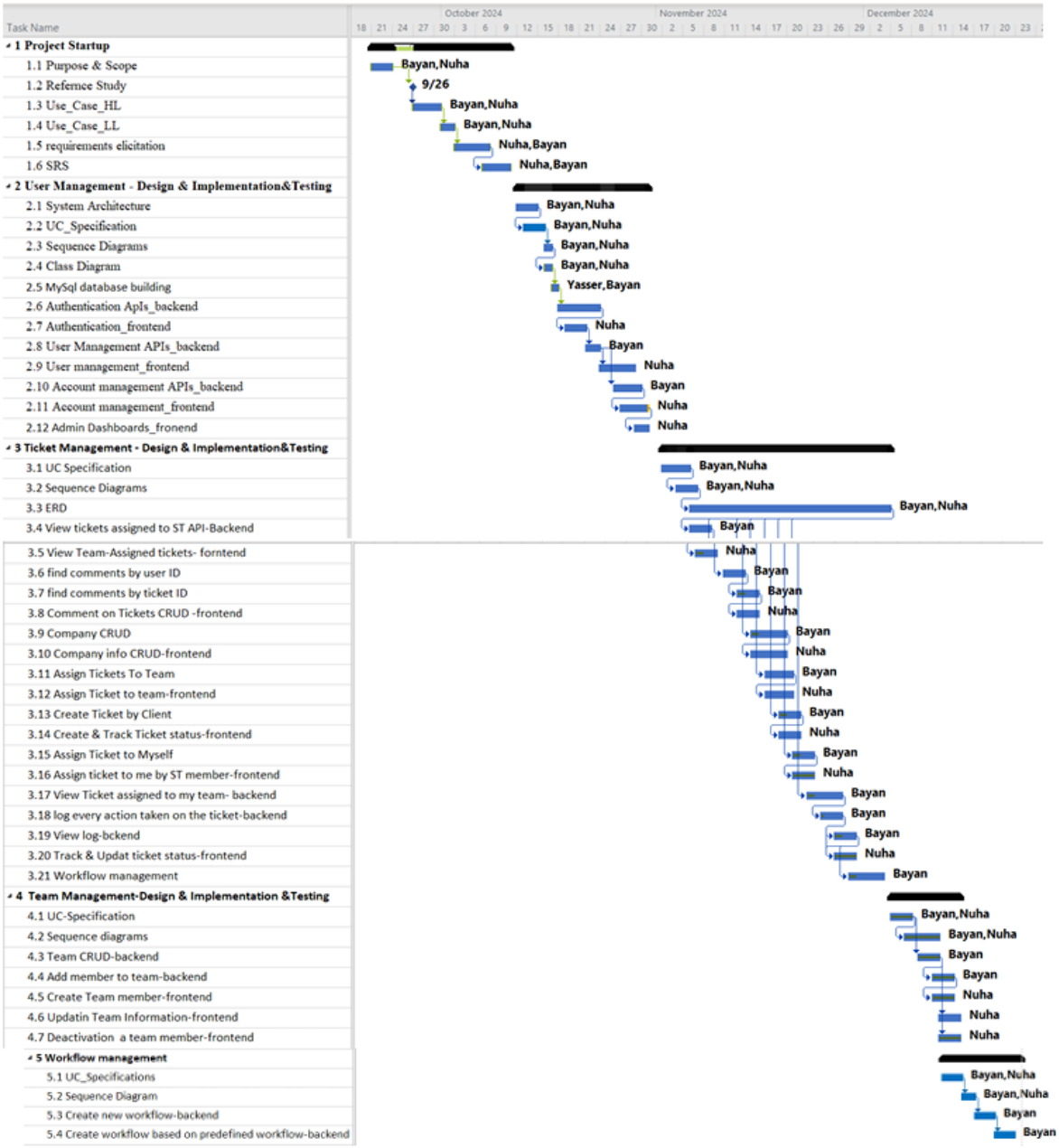


Figure 1 Gantt chart

6. Summary

In conclusion, good project management is vital for successful software system development. It provides structure, ensuring projects are completed within scope and schedule. Overall, strong project management enhances the delivery of high-quality software systems that meet the goals of the project.

Chapter 4 System Analysis

1. Introduction

This chapter focuses on the detailed analysis of the AI-Powered Ticketing System. It defines the system's requirements, functionality, and architecture by examining user needs and the operational context. The chapter includes an exploration of functional and non-functional requirements, use cases, and workflows to provide a comprehensive understanding of how the system will address the identified challenges and achieve the project objectives. This analysis forms the foundation for designing and implementing the proposed solution.

2. Project Timeline

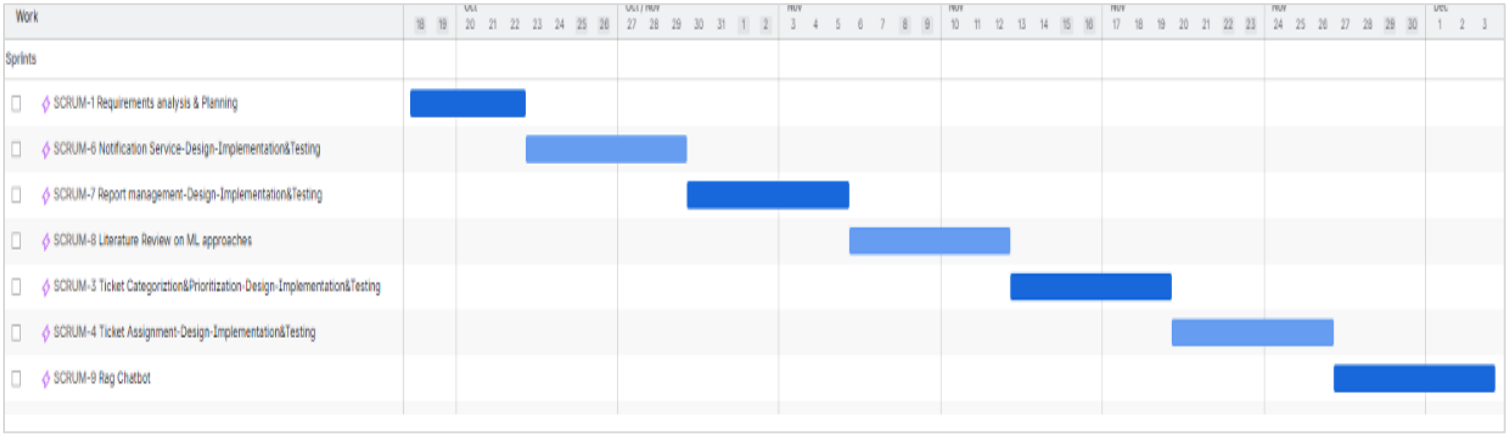


Figure 2 Timeline

3. Software Requirement Specification document (SRS)

Version 0.5

Prepared by:

Nuha Tinawi

Bayan Alallan

Mohammad Yasser Bazallah

Revision history

Date	Reason for change	version
7/10/2024	Add more details about the reports and actors.	0.1
11/10/2024	Add revision history.	0.2
25/12/2024	Updated functional requirements	0.3
23/3/2025	Updated version for senior-1 project	0.4
20/12/2025	Updated version for senior-2 project	0.5

1. Introduction

1.1 Purpose

This document specifies the software requirements and design for AI-Powered Ticketing System, a solution designed to simplify customer request and complaint management.

1.2 Project Scope

The AI-Powered Ticketing System is designed to automate and enhance the management of customer requests and complaints through the use of artificial intelligence technologies. The system focuses on improving response times, organizing ticket handling processes, and increasing customer satisfaction. This specification defines the scope of the system by outlining its high-level functionality, supported features, and the primary actors involved in the ticket management process.

1.2.1 High-Level Requirements:

- **User Management**: This functionality allows administrators to manage users by adding, updating, and deleting user profiles. It also has role-based access control, automatically assigning roles (Support Team Manager, Support Team Member, or Client) based on the actor creating the user or the registration process.

- **Team Management**: The system provides tools to manage support teams, including adding, updating, deleting, and viewing team profiles

- **Ticket Management**: This This requirement covers the core functionality of managing customer support tickets throughout their entire lifecycle. The system shall support ticket creation, updating, assignment, resolution, and status tracking. It shall provide dynamic, company-specific workflows, including a default workflow and customizable workflows with user-defined steps. The system shall support intelligent ticket processing through artificial intelligence techniques, including automated ticket

categorization, priority determination, and automatic assignment of tickets to the appropriate support team based on ticket content analysis. These automation features shall be configurable and may be enabled or disabled by administrators according to organizational needs.
In addition, the system shall support AI-assisted ticket handling by enabling contextual assistance for support teams during ticket resolution, helping improve response quality and efficiency without replacing human decision-making.

- **Member management**: The system allows Support Team Managers to manage team members by adding them to specific teams, activating or deactivating their accounts, and enabling ticket assignment and resolution features.

- **Notification System:** The system shall provide a notification mechanism to inform users about important ticket-related events. Notifications shall be triggered when the status of a ticket changes, such as assignment, progress updates, or resolution.
  The notification functionality shall be provided as local (in-system) notifications and shall be available exclusively to Client users, ensuring they are informed about the progress of their submitted tickets. The notification system shall operate independently from other system components and shall respect role-based access control policies.

- **Workflow Customization:** The system shall support customizable ticket workflows to adapt to different organizational needs. Workflow definition and configuration shall be performed by the Administrator during the creation of a Support Team Manager account. The Administrator shall be able to assign either a predefined default workflow or a custom workflow to each Support Team Manager.
  Once assigned, the workflow shall govern the ticket processing stages available to the corresponding support team and shall not be modifiable directly by the Support Team Manager unless explicitly reconfigured by

the Administrator. This approach ensures consistency, control, and proper governance of ticket handling processes across the system.

- **Reporting and Analytics:** The system shall provide reporting and analytics capabilities to support monitoring and evaluation of ticket management performance. It shall offer visual dashboards that present key metrics such as ticket status distribution, ticket volume over time, team workload, and resolution progress.
  The system shall allow authorized users, specifically Administrators and Support Team Managers, to generate and export reports in XML format for further analysis and documentation purposes. Access to reporting and analytics features shall be restricted based on role-based access control policies to ensure data confidentiality and proper usage.

- **Company Management:** The system provides administrators with tools to manage company profiles, including adding, updating, deleting, searching, and viewing company information.
  Each company can be configured with either a Default Workflow or a Customized Workflow.

- **Comments Management:** The system allows Support Team Managers, Support Team Members, and Clients to collaborate by adding, editing, deleting, and viewing comments on tickets.

- **Account management:** Users can manage their account information, including editing and deleting their profiles.

- **Sign-Up:** The system allows new users to register by providing necessary.

➢ AI HIGH LEVEL REQUIREMENT

- **AI-Powered Ticket Prioritization**: The system shall support automatic prioritization of support tickets based on the analysis of ticket content and customer sentiment. This functionality shall utilize natural language processing techniques to assess the urgency of requests and assign appropriate priority levels. The ticket prioritization process shall operate independently from ticket categorization and team assignment and shall be configurable, allowing administrators to enable or disable this feature as required.

- **AI-Powered Ticket Categorization:** The system shall automatically assign a category to each ticket using a BERT-based natural language processing model to support efficient triaging and routing of customer requests. The categorization process shall be based on a predefined set of ticket categories, namely Request, Feedback, and Complaint, ensuring consistent classification across the system.

- **Auto Team Assignment:** The system shall support automatic assignment of tickets to the appropriate support team based solely on the analysis of the ticket content. This functionality shall utilize a BERT-based natural language processing model trained on a dedicated dataset to determine the most suitable team, such as HR, IT, Sales, or other predefined teams, responsible for handling the ticket.
  The team assignment process shall operate independently from ticket categorization and prioritization mechanisms, ensuring a clear separation between category classification, priority determination, and team routing. The automatic team assignment feature shall be configurable, allowing administrators to enable or disable it based on organizational needs.

- **AI Feature Control (On/Off per Company):** The system shall provide administrators with the ability to enable or disable artificial intelligence features on a per-company basis. These features shall include automated

ticket categorization, automatic team assignment, and AI-powered ticket prioritization. Each AI feature shall operate independently and can be controlled separately without affecting the functionality of other system components. This configurability allows organizations to adapt the system's level of automation according to their operational needs and preferences.

- **RAG-Based AI Chatbot:** The system shall provide an AI-powered chatbot based on Retrieval-Augmented Generation (RAG) to assist support teams during ticket handling and resolution. The chatbot shall analyze user queries and retrieve relevant information from a predefined knowledge base and historical ticket data to generate context-aware responses.
  The chatbot functionality shall serve as a decision-support tool by suggesting possible solutions or guidance without automatically resolving tickets or replacing human judgment. The system shall support response classification to determine whether the generated answer is suitable, requires refinement, or needs human intervention. Access to the chatbot functionality shall be restricted to authorized user roles according to system policies.

1.2.2 Actors:

- **Admin**: The Admin is responsible for overseeing and maintaining the entire system and holds the highest level of access and control over the platform. The Admin shall manage system configurations, users, companies, and global settings. When creating or modifying a company or a Support Team Manager account, the Admin shall configure the applicable ticket workflow by assigning either a predefined default workflow or a customized workflow with defined steps, ensuring controlled and consistent ticket processing across the system.

- **Client:** Clients are end users who submit support requests, feedback, or complaints through the ticketing system. They shall be able to create

tickets, view and track the status of their submitted tickets, add comments or additional information, and manage their personal account information.

The system shall provide Clients with local (in-system) notifications to inform them of ticket status changes, ensuring transparency and continuous awareness of ticket progress throughout the resolution process.

- **Support team manager (STM):** The Support Team Manager is responsible for overseeing the activities of the support team and ensuring efficient ticket resolution. They have access to both ticket management and team management functionalities, including adding and managing team members, and assigning tickets. STMs can move tickets across workflow steps (whether default or customized) and review AI-generated categories.

- **Support team member:** Support team members are responsible for addressing tickets submitted by clients. Their primary focus is on resolving issues efficiently. They can view, assign, and resolve tickets, as well as collaborate with clients and team members by adding comments to tickets. Support Team Members interact with the workflow stages by updating ticket statuses and progressing tickets through defined steps. They also view the ticket's AI-assigned category to better understand the context of the issue.

2. Overall Description

2.1 product perspective

The AI-Powered Ticketing System described in this document is a standalone software product developed as an academic project. It is designed to support the management of customer requests and complaints through structured workflows and intelligent automation. The system is developed independently and is not intended to replace or integrate with an existing ticketing platform.

2.2 Product Features

- **User Management:**
  Create, update, search, and delete user profiles.
  Role-based access control (Admin, Client, Support team Manager).

- **Team Management:**
  Organize and manage support teams, including adding, updating, and deleting teams.

- **Ticket Management:**
  - Create, and track customer support tickets.
  - Automatically assign tickets to the appropriate support team using a BERT-based model trained on ticket content.
  - resolving tickets with Ticket prioritization and status tracking.
  - Support Team Managers can enable automatic ticket priority estimation and manually adjust priorities.
  - The system supports Default and Customized Workflows, allowing each company to define its own ticket progression steps.
  - Tickets are displayed in a Kanban-style view, where team members can move tickets across workflow stages.

- **Sentiment Analysis:**
  Automatically detect urgent issues based on sentiment and prioritize tickets accordingly.

- **Workflow Customization:**
  Admins can define a custom workflow for each company or apply a default workflow. Custom workflows consist of user-defined steps, allowing companies to tailor the lifecycle of a ticket according to their internal processes.

- **Comments management:**
  Allow Support Team Managers, Support Team Members, and Clients to add, and view comments on tickets to enhance collaboration.

- **Company management:**
  Add, update, delete, and view company profiles to manage organizational data seamlessly. Admins can assign each company either a Default Workflow or a Customized Workflow with custom ticket stages.

- **Ticket Categorization:**
  The system automatically categorizes tickets based on their content using a BERT-based natural language processing model. Ticket categories are predefined as Request, Feedback, and Complaint, ensuring consistent classification and improved ticket organization

- **Auto Team Assignment:**
  The system automatically assigns each ticket to the most appropriate support team based on the analysis of the ticket content. This functionality utilizes a BERT-based natural language processing model trained on a dedicated dataset to determine the responsible team (such as HR, IT, Sales, or other predefined teams). The team assignment process operates independently from ticket categorization and priority estimation and can be enabled or disabled by the Admin.

- **RAG-Based AI Chatbot:**
  The system provides an AI-powered chatbot based on Retrieval-Augmented Generation (RAG) to assist support teams during ticket resolution. The chatbot retrieves relevant information from a knowledge base and previous tickets to generate context-aware responses and suggestions. It serves as a decision-support tool and does not automatically resolve tickets, ensuring that final decisions remain under human control.

2.3 User Classes and Characteristics

2.3.1 Admin

Responsibilities: Manages user accounts, configures system settings, assigns roles, oversees integration with other platforms, manages company profiles, and Monitors system-wide reports and analytics via the admin dashboard. Admins can configure custom or default workflows for each company and define the sequence of workflow steps when customizing.

Privileges: User management, company management, access to dashboard reports (including user activity analytics), workflow configuration, and all configuration features.

2.3.2 Client

Responsibilities: Submits and tracks tickets, reviews ticket status and responses, and collaborates on tickets by adding comments.

Privileges: Limited to ticket creation and viewing, and managing their own account information. Clients can view the automatically assigned ticket category for their submitted tickets (read-only).

2.3.3 Support Team Manager

Responsibilities : Oversees the support team, assigns tickets, monitors team performance using dashboard reports, manages team workflows, including defining and modifying ticket workflow stages, enables or adjusts automatic ticket prioritization, and manages team members. The STM can also create users with the role of Support Team Member. Can move tickets between workflow steps and update their status within custom or default workflows. Also, can view and override AI-assigned ticket categories if needed.

Privileges: Access to team management, member management, ticket management, can manually modify ticket priority and track overdue tickets, and Access to performance and workload reports for team members, include managing workflow steps and overriding ticket categorization results.

2.3.4 Support Team Member

Responsibilities: Handles and resolves support tickets, assigns tickets to themselves, and collaborates with other users by adding comments on tickets.

Moves tickets across workflow stages and updates their status as per the defined process.

Can view AI-assigned categories to understand ticket context before handling.

Privileges: Access to assigned tickets and the ability to manage ticket resolution and collaboration, Ability to transition tickets through workflow steps and access ticket metadata including AI-generated category.

3. System features

3.1 functional requirements

3.1.1 User management

REQ-1.1:

The system shall allow the admin to create a new user profile by entering necessary information.

REQ-1.2:

The system shall allow the Admin to edit user information.

REQ-1.3:

The system shall allow the Admin to delete a user.

REQ-1.4:

The system shall allow the Admin to search for a user by name.

REQ-1.5:

The system shall automatically assign a default role to a newly created user based on the actor performing the creation:

- **REQ-1.5.1:** When the Admin creates a new user, the system shall assign the role Support Team Manager by default.
- **REQ-1.5.2:** When a user registers through the system, the system shall assign the role Client by default.

- **REQ-1.5.3:** When a Support Team Manager creates a new user, the system shall assign the role Support Team Member by default.

3.1.2 Team Management

REQ-2.1:

The system shall allow Support Team Manager (STM) to add team to support teams.

REQ-2.2:

STM shall be able to delete team from the support teams.

REQ-2.3:

The system shall allow STM to edit team details.

REQ-2.4:

The system shall allow STM to view a list of all support teams.

REQ-2.5:

The system shall allow STM to search for a team by name.

3.1.3 Ticket management

REQ-3.1:

Upon selecting "Create Ticket" the system shall allow Clients to create a new ticket, entering required details such as issue description.

REQ-3.2:

The system shall provide an option for the STMs to assign tickets to specific team for resolution.

REQ-3.3:

The system shall maintain a ticket log to track the status changes for every ticket.

- **REQ-3.3.1:** The system shall log the creation of a new ticket with details including the creator, timestamp, and initial status.
- **REQ-3.3.2:** The system shall log every status change of the ticket, including the user who initiated the change, the new status, and the timestamp.
- **REQ-3.3.3:** The system shall log the resolution of a ticket with details of the resolver, the final status, and the resolution timestamp.

- **REQ-3.3.4:** The ticket log shall be accessible to authorized users, such as Support Team Manager, and Support Team Member, for review**.**

REQ-3.4:

Upon ticket creation, the system shall employ an NLP-based approach to analyze the content of the ticket and automatically estimate its priority based on keywords, urgency, and sentiment analysis. This functionality shall be configurable and may be enabled or disabled by the STM.

REQ-3.5:

The system shall provide functionality to view ticket details, including priority, category, assigned support team, company name, ticket status, client name, and ticket description.

REQ-3.6:

The system shall allow authorized users to delete a ticket if required, with a record of the deletion captured in the ticket log.

REQ-3.7:

The system shall allow authorized users to update a ticket, with a record of the update captured in the ticket log.

REQ-3.8:

The system shall allow Support Team Managers and Support Team Members to view assigned tickets

REQ-3.9:

The system shall automatically assign a category to the ticket using a BERT-based classifier based on the ticket content. The supported categories shall include Request, Feedback, and Complaint.

REQ-3.10:

The system shall allow Support Team Managers and Support Team Members to change the status of a ticket (e.g., "In Progress," "Done")

REQ-3.11:

The system shall support two types of workflows for ticket progression:

- Default Workflow (e.g., Open → Assign to Team → Assign to Member → In Progress → Done)

- Customized Workflow, defined per company, with user-defined steps such as(Step 1 → Step 2 → Step 3...).

REQ-3.12:

The system shall support automatic assignment of tickets to the appropriate support team based on ticket content analysis using a dedicated BERT-based model.

3.1.4 Member management

REQ-4.1:

The system shall allow Support Team members to view all tickets that have been assigned to their team.

REQ-4.2:

The system shall allow Support Team members to resolve tickets that have been assigned to them.

REQ-4.3:

The system shall allow the Support Team Manager (STM) to add new members to the system.

REQ-4.4:

The system shall enable the STM to add members to a specific support team.

REQ-4.5:

The STM shall be able to change the status of a team member, including activating or deactivating the member.

REQ-4.6:

Support team members shall be able to assign a ticket to themselves.

3.1.5 Account information management

REQ-5.1:

The system shall allow all users to edit their account information.

REQ-5.2:

Users shall be able to delete their accounts.

3.1.6 Company management

REQ-6.1:

The system shall allow the Admin to add a company.

REQ-6.2:

The Admin shall be able to delete a company.

REQ-6.3:

The system shall allow the Admin to edit company details.

REQ-6.4:

The Admin shall be able to view a list of all companies.

REQ-6.5:

The system shall allow the Admin to search for a company by name.

3.1.7 Comments management

REQ-7.1:

The system shall allow Support Team Members and Clients to add comments to tickets.

REQ-7.2:

The system shall allow STM, Support Team Member, and Client to delete their own comments.

REQ-7.3:

The system shall allow STM, Support Team Member, and Client to view a list of comments associated with their tickets.

3.1.8 Workflow Customization

REQ-8.1:

The system shall allow the admin to choose between a Default Workflow or a Customized Workflow when adding or editing a company profile.

REQ-8.2:

When selecting Customized Workflow, the system shall allow the Admin to define a sequence of custom steps (e.g., Step 1, Step 2, Step 3) for ticket progression.

REQ-8.3:

Support Team Managers and Members shall be able to transition tickets between steps based on the company's configured workflow.

REQ-8.4:

The system shall visually display the workflow as a Kanban-style board, allowing authorized users to move tickets across the defined steps.

REQ-8.5:

The system shall store the workflow configuration for each company and apply it automatically to all tickets submitted under that company.

3.1.9 Ticket Categorization

REQ-9.1:

The system shall use a BERT-based machine learning model to analyze the textual content of a ticket upon creation.

REQ-9.2:

Based on the content analysis, the system shall automatically assign a predefined category to the ticket. The predefined categories shall include Request, Feedback, and Complaint.

REQ-9.3:

The system shall display the assigned category as part of the ticket's metadata, visible to all relevant users.

REQ-9.4:

The system shall store the categorized data for analytics and future model improvement.

3.1.10 Sign up

REQ-10.1:

the system shall allow new users to register by providing necessary information.

3.1.11 Auto Team Assignment

REQ-11.1:

The system shall analyze the textual content of a ticket upon creation using a dedicated BERT-based machine learning model trained for support team identification.

REQ-11.2:

Based on the content analysis, the system shall automatically assign the ticket to the most appropriate support team (such as HR, IT, Sales, or other predefined teams).

REQ-11.3:

The auto team assignment process shall operate independently from ticket categorization and ticket prioritization mechanisms, ensuring that each AI-based function produces a separate and distinct output.

REQ-11.4:

The system shall allow the Admin to enable or disable the automatic team assignment feature on a per-company basis.

REQ-11.5:

When the auto team assignment feature is disabled, the system shall allow manual assignment of tickets to support teams by authorized users.

3.1.12 RAG-Based AI Chatbot

REQ-12.1:

The system shall provide an AI-powered chatbot based on Retrieval-Augmented Generation (RAG) to assist support teams during ticket handling and resolution.

REQ-12.2:

The chatbot shall retrieve relevant information from a predefined Knowledge Base and historical ticket data to generate context-aware responses related to the current ticket or user query.

REQ-12.3:

The system shall provide Knowledge Base management functionality, allowing authorized users to upload, update, view, and delete files used by the RAG-based chatbot.

REQ-12.4:

The system shall ensure that any updates or deletions of Knowledge Base files are reflected in the chatbot's retrieval and response generation process.

REQ-12.5:

The system shall support two RAG-based interaction modes:

- **Chat Engine,** which provides interactive, multi-turn conversational assistance.
- **Solution Engine,** which generates structured and actionable solution suggestions focused on resolving a specific ticket.

REQ-12.6:

The Chat Engine shall support conversational question-answering, allowing users to ask follow-up questions while maintaining contextual continuity.

REQ-12.7:

The Solution Engine shall generate resolution-oriented outputs, such as suggested steps or guidance, based on the ticket content and retrieved knowledge base information.

REQ-12.8:

The chatbot shall generate responses intended to support and guide support team members without automatically resolving tickets or performing actions on their behalf.

REQ-12.9:

The system shall classify chatbot responses into predefined response types, including normal, soft refusal, hard refusal, and needs human intervention, to ensure safe and appropriate usage of generated content.

REQ-12.10:

When a response is classified as requiring human intervention, the system shall flag the interaction for review and allow authorized users to take appropriate action.

REQ-12.11:

Access to the RAG-based chatbot and Knowledge Base management functionalities shall be restricted to authorized user roles according to role-based access control policies.

3.1.13 Notification System

REQ-13.1:

The system shall provide a local (in-system) notification mechanism to inform users about ticket-related events.

REQ-13.2:

The system shall generate notifications for Clients when the status of their submitted tickets changes, such as assignment, progress updates, or resolution.

REQ-13.3:

Notifications shall be displayed within the system interface and shall not rely on external communication channels.

REQ-13.4:

The notification system shall operate in accordance with role-based access control policies, ensuring that notifications are visible only to the intended Client users.

3.1.14 Reporting and Analytics

REQ-14.1:

The system shall provide reporting and analytics features to support monitoring and evaluation of ticket management activities.

REQ-14.2:

The system shall display visual dashboards presenting key metrics, including ticket status distribution, ticket volume over time, team workload, and ticket resolution progress.

REQ-14.3:

The system shall allow Administrators and Support Team Managers to generate detailed reports based on ticket data and system activity.

REQ-14.4:

Authorized users shall be able to export generated reports in XML format for documentation, analysis, or integration purposes.

REQ-14.5:

Access to reporting and analytics features shall be restricted according to role-based access control policies to ensure data confidentiality and proper usage.

3.1.15 AI Feature Control (On/Off per Company)

REQ-AI-CTRL-1:

The system shall allow the Admin to enable or disable artificial intelligence features on a per-company basis.

REQ-AI-CTRL-2:

The controllable AI features shall include:

- AI-powered ticket categorization
- Auto team assignment
- AI-based ticket prioritization

Each feature shall be configurable independently.

REQ-AI-CTRL-3:

Disabling an AI feature shall not affect the core ticket management functionality or other enabled AI features.

REQ-AI-CTRL-4:

When an AI feature is disabled for a company, the system shall rely on manual processes or default system behavior for the corresponding functionality.

REQ-AI-CTRL-5:

The system shall store and apply AI feature configuration settings separately for each company to ensure customization and isolation between organizations.

4. Non-Functional Requirement

4.1 Scalability

NF-1.1: The system shall be scalable to support an increasing number of users and tickets without performance degradation.

NF-1.2: The system architecture shall allow the addition of new resources or services to handle increased workloads dynamically.

4.2 Availability

NF-2.1: The system shall maintain a high level of availability to ensure continuous access to ticket management and support functionalities.

4.3 interoperability

NF-3.1: The system shall be capable of integrating with third-party customer support platforms.

NF-3.2: The system shall ensure data compatibility with external systems.

4.4 Security

NF-4.1: The system shall enforce secure authentication mechanisms, requiring users to log in using strong passwords and role-based access control policies.

NF-4.2: The system shall encrypt all sensitive data, including passwords, user information, and ticket-related data, to ensure data confidentiality and protection.

4.5 Extensibility

NF-5.1: The system shall be extensible to support multiple languages, allowing for easy integration of new language modules in the future without significant architectural changes.

NF-5.2: The system shall be designed to allow seamless integration of additional artificial intelligence components, such as new NLP models or machine learning algorithms, enabling future enhancements without major redevelopment.

4. System Requirements

Table 5 System Requirements

Req-ID	Requirement Title	Category	Priority
REQ-1.1	The system shall allow the admin to create a new user profile by entering necessary information.	User Management	High
REQ-1.2	The system shall allow the Admin to edit user information.	User Management	High
REQ-1.3	The system shall allow the Admin to delete a user.	User Management	High
REQ-1.4	The system shall automatically assign a default role to a newly created user based on the actor performing the creation.	User Management	High
REQ-1.4.1	When the Admin creates a new user, the system shall assign the role Support Team Manager by default.	User Management	Medium
REQ-1.4.2	When a user registers through the system, the system shall assign the role Client by default.	User Management	Medium
REQ-1.4.3	When a Support Team Manager creates a new user, the system shall assign the role Support Team Member by default.	User Management	Medium
REQ-2.1	The system shall allow Support Team Manager (STM) to add team to support teams.	Team Management	High
REQ-2.2	STM shall be able to delete team from the support teams.	Team Management	High
REQ-2.3	The system shall allow STM to edit team details.	Team Management	High
REQ-2.4	The system shall allow the STM to view a list of all support teams.	Team Management	Medium
REQ-3.1	Upon selecting 'Create Ticket' the system shall allow Clients to create a new ticket, entering required details such as issue description.	Ticket Management	High

Req-ID	Requirement Title	Category	Priority
REQ-3.2	The system shall provide an option for the STMs to assign tickets to specific teams for resolution.	Ticket Management	High
REQ-3.3	The system shall maintain a ticket log to track the status changes for every ticket.	Ticket Management	High
REQ-3.3.1	The system shall log the creation of a new ticket with details including the creator, timestamp, and initial status.	Ticket Management	Medium
REQ-3.3.2	The system shall log every status change of the ticket, including the user who initiated the change, the new status, and the timestamp.	Ticket Management	Medium
REQ-3.3.3	The system shall log the resolution of a ticket with details of the resolver, the final status, and the resolution timestamp.	Ticket Management	Medium
REQ-3.3.4	The ticket log shall be accessible to authorized users for review.	Ticket Management	Medium
REQ-3.4	Upon ticket creation, the system shall employ an NLP-based approach to analyze the content of the ticket and automatically estimate its priority based on keywords, urgency, and sentiment analysis.	Ticket Management	Medium
REQ-3.5	The system shall provide functionality to view ticket details, including priority, category, company name, status of ticket, client name, and ticket description	Ticket Management	High
REQ-3.6	The system shall allow authorized users to delete a ticket if required, with a record of the deletion captured in the ticket log.	Ticket Management	Medium
REQ-3.7	The system shall allow authorized users to update a ticket, with a record of the deletion captured in the ticket log.	Ticket Management	High
REQ-3.8	The system shall allow Support Team Managers and Support Team Members to view assigned tickets		

Req-ID	Requirement Title	Category	Priority
REQ-3.9	The system shall automatically assign a category to the ticket using a BERT-based classifier based on the ticket content.	Ticket Management	High
REQ-3.10	The system shall allow Support Team Managers and Support Team Members to change the status of a ticket (e.g., "In Progress," "Done")	Ticket Management	High
REQ-3.11	The system shall support two types of workflows for ticket progression: -Default Workflow (e.g., Open → Assign to Team → Assign to Member → In Progress → Done) -Customized Workflow, defined per company, with user-defined steps such as Step 1 → Step 2 → Step 3.	Ticket Management	High
REQ-4.1	The system shall allow Support Team members to view all tickets that have been assigned to their team.	Member Management	High
REQ-4.2	The system shall allow Support Team members to resolve tickets that have been assigned to them.	Member Management	High
REQ-4.3	The system shall allow the Support Team Manager (STM) to add new members to the system.	Member Management	High
REQ-5.1	The system shall allow all users to edit their account information.	Account Information Management	Medium
REQ-5.2	Users shall be able to delete their accounts.	Account Information Management	Medium
REQ-6.1	The system shall allow the Admin to add a company.	Company Management	High
REQ-6.2	The Admin shall be able to delete a company.	Company Management	High
REQ-7.1	The system shall allow members and clients to add a comment on a ticket.	Comments Management	Medium

Req-ID	Requirement Title	Category	Priority
REQ-7.2	The system shall allow STM, Support Team Members, and Clients to view a list of comments associated with their tickets.	Comments Management	Medium
REQ-8.1	The system shall allow the Admin to choose between a Default Workflow or a Customized Workflow when adding or editing a company profile.	Workflow Customization	High
REQ-8.2	When selecting Customized Workflow, the system shall allow the Admin to define a sequence of custom steps (e.g., Step 1, Step 2, Step 3) for ticket progression.	Workflow Customization	Medium
REQ-8.3	Support Team Managers and Members shall be able to transition tickets between steps based on the company's configured workflow.	Workflow Customization	Medium
REQ-8.4	The system shall visually display the workflow as a Kanban-style board, allowing authorized users to move tickets across the defined steps.	Workflow Customization	
REQ-8.5	The system shall store the workflow configuration for each company and apply it automatically to all tickets submitted under that company.	Workflow Customization	
REQ-9.1	The system shall use a BERT-based machine learning model to analyze the textual content of a ticket upon creation.	Ticket Categorization	
REQ-9.2	Based on the content analysis, the system shall automatically assign a predefined category to the ticket.	Ticket Categorization	
REQ-9.3	The system shall display the assigned category as part of the ticket's metadata, visible to all relevant users.	Ticket Categorization	
REQ-9.4	The system shall store the categorized data for analytics and future model improvement.	Ticket Categorization	

Req-ID	Requirement Title	Category	Priority
REQ-10.1	The system shall allow new clients to register by providing necessary information.	Registration Management	High
REQ-10.2	The system shall provide a login page for all users, requiring valid credentials.	Login Management	High
REQ-10.3	The system shall validate user credentials against the database during login attempts.	Login Management	High
NF-1.1	The system shall be scalable to support an increasing number of users and tickets.	Scalability	High
NF-1.2	The system architecture shall allow the addition of new resources or services to handle increased workloads dynamically.		
NF-2.1	The system shall maintain a high level of availability to ensure continuous access to ticket management and support functionalities.	Availability	High
NF-3.1	The system shall be capable of integrating with third-party customer support platforms.	Interoperability	Medium
NF-4.1	The system shall enforce secure authentication mechanisms, requiring users to log in using strong passwords and role-based access control policies.	Security	High
NF-4.2	The system shall encrypt all sensitive data, including passwords, user information, and ticket-related data, to ensure data confidentiality and protection.	Security	High
NF-5.1	The system shall be extensible to support multiple languages.	Extensibility	Medium
NF-5.2	The system shall be designed to allow seamless integration of additional artificial intelligence components, such as new NLP models or machine learning algorithms, enabling future enhancements without major redevelopment.	Extensibility	Medium

Senior2 requirement

Req-ID	Requirement Title	Category	Priority
REQ-14.1:	The system shall provide reporting and analytics features to support monitoring and evaluation of ticket management activities.	Reporting and Analytics	Medium
REQ-14.2:	The system shall display visual dashboards presenting key metrics, including ticket status distribution, ticket volume over time, team workload, and ticket resolution progress.	Reporting and Analytics	Medium
REQ-14.3:	The system shall allow Administrators and Support Team Managers to generate detailed reports based on ticket data and system activity.	Reporting and Analytics	Medium
REQ-14.4:	Authorized users shall be able to export generated reports in XML format for documentation, analysis, or integration purposes.	Reporting and Analytics	Medium
REQ-14.5:	Access to reporting and analytics features shall be restricted according to role-based access control policies to ensure data confidentiality and proper usage.	Reporting and Analytics	Medium
REQ-13.1:	The system shall provide a local (in-system) notification mechanism to inform users about ticket-related events.	Notification System	Medium
REQ-13.2:	The system shall generate notifications for Clients when the status of their submitted tickets changes, such as assignment, progress updates, or resolution.	Notification System	Medium
REQ-13.3:	Notifications shall be displayed within the system interface and shall not rely on external communication channels.	Notification System	Medium
REQ-13.4:	The notification system shall operate in accordance with role-based access control policies, ensuring that notifications are visible only to the intended Client users.	Notification System	Medium

Req-ID	Requirement Title	Category	Priority
REQ-12.1:	The system shall provide an AI-powered chatbot based on Retrieval-Augmented Generation (RAG) to assist support teams during ticket handling and resolution.	RAG-Based AI Chatbot	High
REQ-12.2:	The chatbot shall retrieve relevant information from a predefined Knowledge Base and historical ticket data to generate context-aware responses related to the current ticket or user query.	RAG-Based AI Chatbot	High
REQ-12.3:	The system shall provide Knowledge Base management functionality, allowing authorized users to upload, update, view, and delete files used by the RAG-based chatbot.	RAG-Based AI Chatbot	High
REQ-12.4:	The system shall ensure that any updates or deletions of Knowledge Base files are reflected in the chatbot's retrieval and response generation process.	RAG-Based AI Chatbot	High
REQ-12.5:	The system shall support two RAG-based interaction modes: • Chat Engine, which provides interactive, multi-turn conversational assistance. • Solution Engine, which generates structured and actionable solution suggestions focused on resolving a specific ticket.	RAG-Based AI Chatbot	High
REQ-12.6	The Chat Engine shall support conversational question-answering, allowing users to ask follow-up questions while maintaining contextual continuity.	RAG-Based AI Chatbot	High
REQ-12.7	The Solution Engine shall generate resolution-oriented outputs, such as suggested steps or guidance, based on the ticket content and retrieved knowledge base information.	RAG-Based AI Chatbot	High

Req-ID	Requirement Title	Category	Priority
REQ-12.8	The chatbot shall generate responses intended to support and guide support team members without automatically resolving tickets or performing actions on their behalf.	RAG-Based AI Chatbot	High
REQ-12.9	The system shall classify chatbot responses into predefined response types, including normal, soft refusal, hard refusal, and needs human intervention, to ensure safe and appropriate usage of generated content.	RAG-Based AI Chatbot	High
REQ-12.10	When a response is classified as requiring human intervention, the system shall flag the interaction for review and allow authorized users to take appropriate action.	RAG-Based AI Chatbot	High
REQ-12.11	Access to the RAG-based chatbot and Knowledge Base management functionalities shall be restricted to authorized user roles according to role-based access control policies.	RAG-Based AI Chatbot	High
REQ-10.1:	The system shall analyze the textual content of a ticket upon creation using a dedicated BERT-based machine learning model trained for support team identification.	Auto Team Assignment	Medium
REQ-10.2:	Based on the content analysis, the system shall automatically assign the ticket to the most appropriate support team (such as HR, IT, Sales, or other predefined teams).	Auto Team Assignment	Medium
REQ-10.3:	The auto team assignment process shall operate independently from ticket categorization and ticket prioritization mechanisms, ensuring that each AI-based function produces a separate and distinct output.	Auto Team Assignment	Medium
REQ-10.4:	The system shall allow the Admin to enable or disable the automatic team assignment feature on a per-company basis.	Auto Team Assignment	Medium
REQ-10.5:	When the auto team assignment feature is disabled, the system shall allow manual	Auto Team Assignment	Medium

	assignment of tickets to support teams by authorized users.		
REQ-AI-CTRL-1:	The system shall allow the Admin to enable or disable artificial intelligence features on a per-company basis.	AI Feature Control	Medium
REQ-AI-CTRL-2	The controllable AI features shall include: AI-powered ticket categorization Auto team assignment AI-based ticket prioritization Each feature shall be configurable independently.	AI Feature Control	Medium
REQ-AI-CTRL-3	Disabling an AI feature shall not affect the core ticket management functionality or other enabled AI features.	AI Feature Control	Medium
REQ-AI-CTRL-4	When an AI feature is disabled for a company, the system shall rely on manual processes or default system behavior for the corresponding functionality.	AI Feature Control	Medium
REQ-AI-CTRL-5	The system shall store and apply AI feature configuration settings separately for each company to ensure customization and isolation between organizations.	AI Feature Control	Medium

5. Requirements modeling

4.1 Basic UML Diagrams

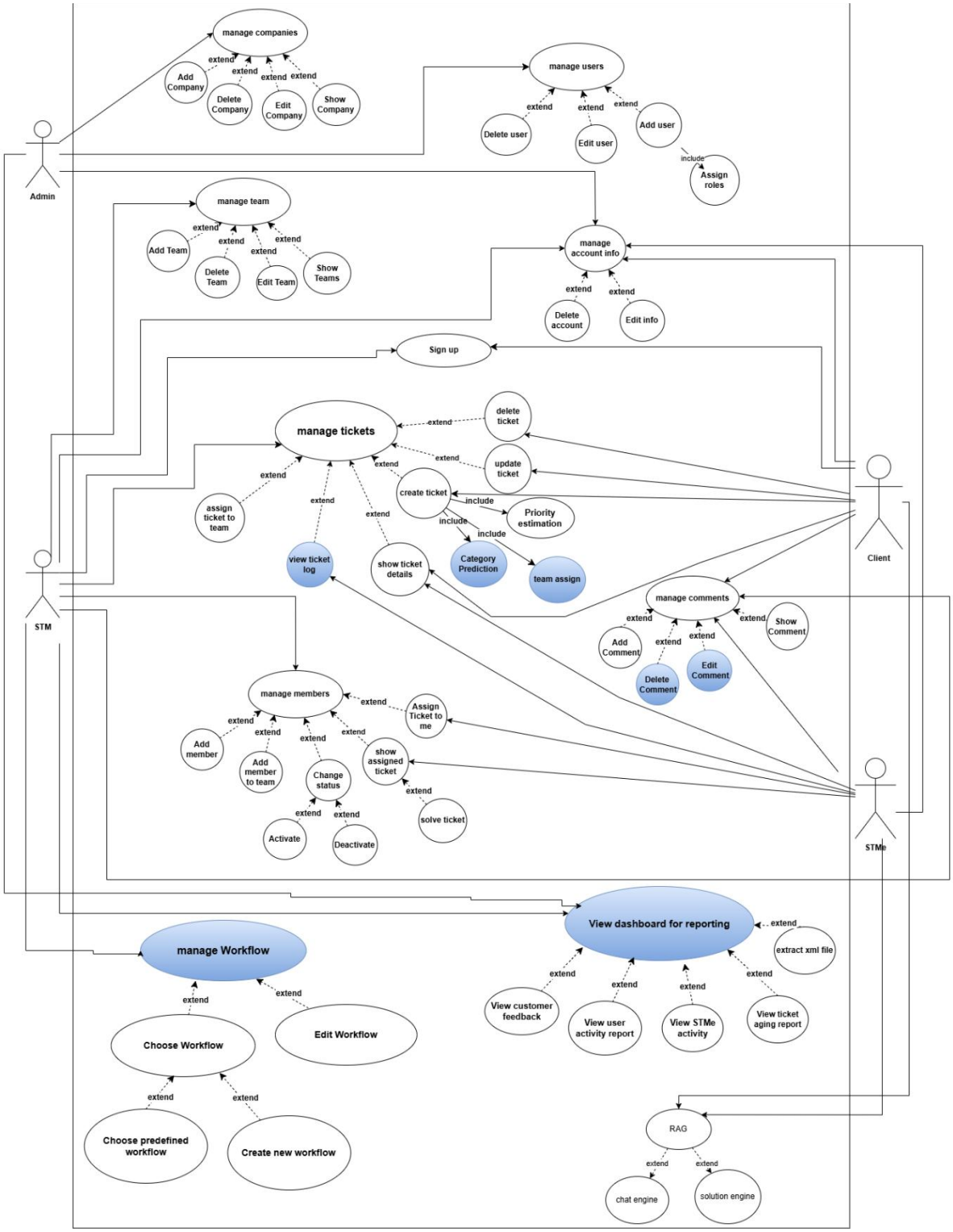


Figure 3 use case diagram

System features (use case specifications - Sequence Diagrams):

- **User management**

Figure4 user management - sequence diagram

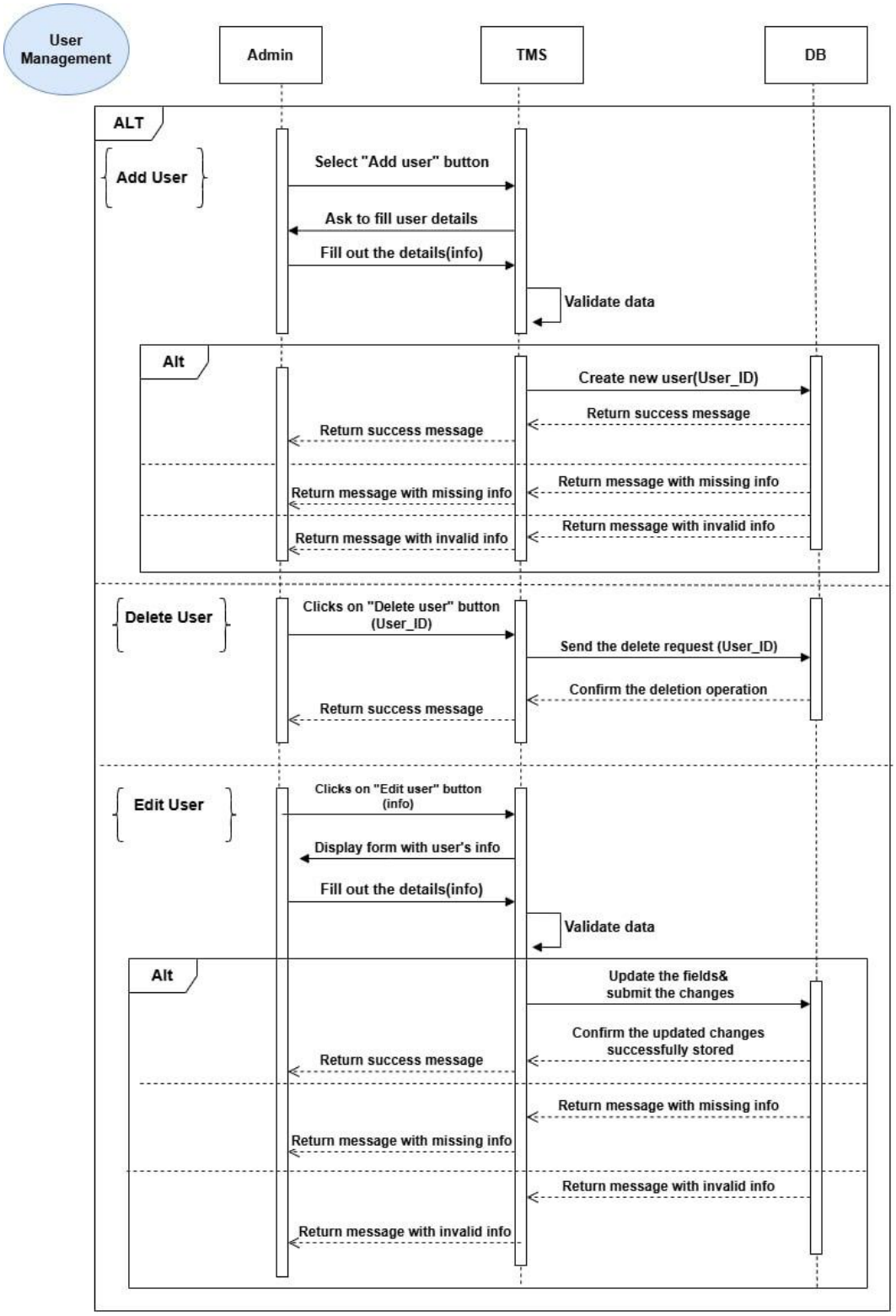


Table 6 user management - usecse specification

Use case name	**User management**
Description	This use case allows the admin manager to manage users.
Actors	Admin
Preconditions	Admin must be logged into the system.
Main Scenario	▪ Admin navigate to the **User management page.** **A. If Admin request to add a new user:** 1. STM selects the “Add user” option. 2. The system displays a form for entering new user details. 3. Admin fills in the details and submits the form. 4. System validates the data. 5. After successful validation, the system creates the new user. 6. The system displays success message. **B. If Admin request to delete user:** 1. The admin clicks the "Delete User" button on the dashboard. 2. The system requests to delete a specific team 3. The system displays a confirmation message to the admin 4. The admin confirms the deletion by selecting the "Confirm "option in the confirmation message. 5. The system deletes the in the database. **C. If Admin request to Edit user status:** 1. admin selects the “Edit user” option. 2. The system display form with the user’s current information 3. Admin updates the desired fields. 4. System validates the updated information and saves changes. 5. The updated information is displayed in the team list.
Alternative Scenario	**(A1) Missing Fields:** 1. The flow starts after point [4.A, 4.C] of the main flow. 2. The system show that the field is missing. 3. The system requests from STM to complete the fields. 4. The flow returns to point 4 and the use case succeeds.
Postconditions	The user is successfully managed.

- **Registration**

Figure5 Registration - sequence diagram

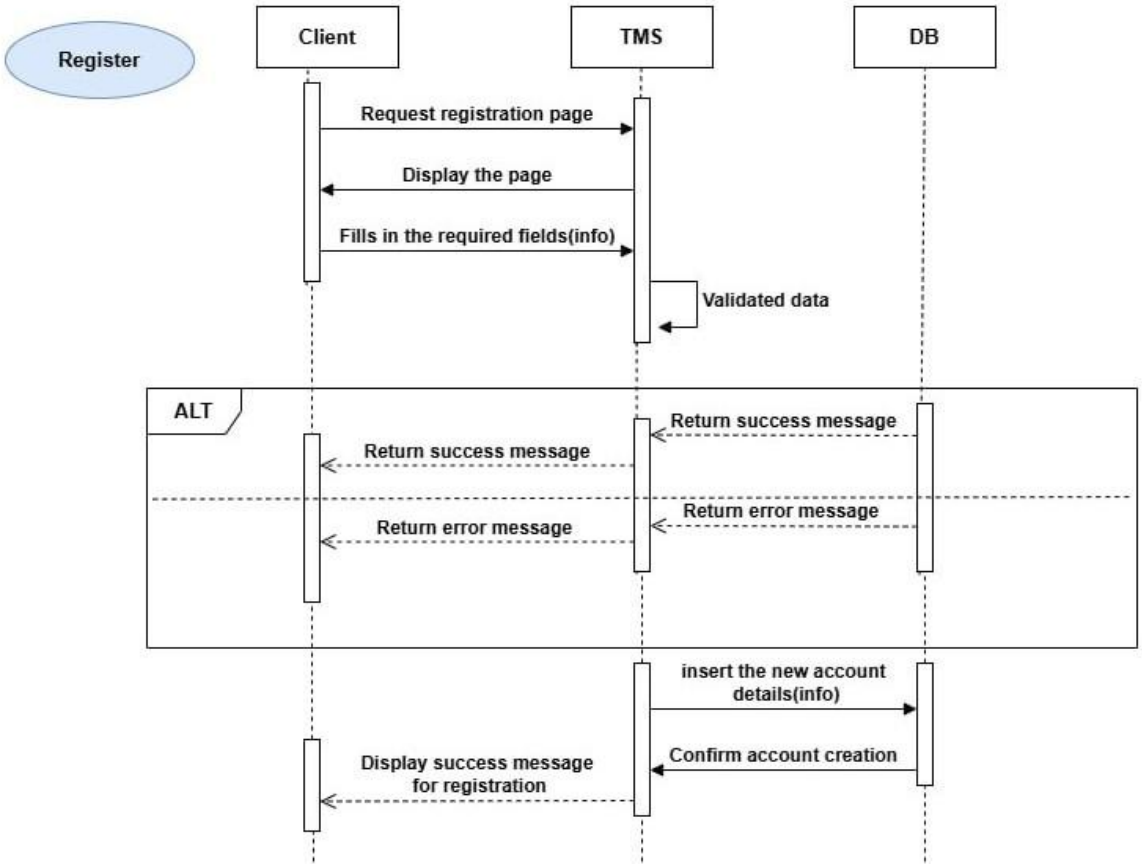


Table 7 Registration - usecase specification

Use Case Name	**Register**
Actor	Client
Precondition	The client must have access to the application or website.
Main Scenario	1. The client launches the application and opens the registration page. 2. The client fills in the required details (e.g., name, email, password, and other relevant fields). 3. The client clicks on the "Register" button. 4. The system validates the input. 5. The system creates the account and displays a success message.
Alternative Scenario	If the user provides invalid or incomplete input (e.g., missing fields, weak password, duplicate email), the system displays an appropriate error message and prompts the user to correct the input.
Postcondition	The client account is created successfully, and the user can proceed to log in.

- **Sign in**

Figure6 sign in - sequence diagram

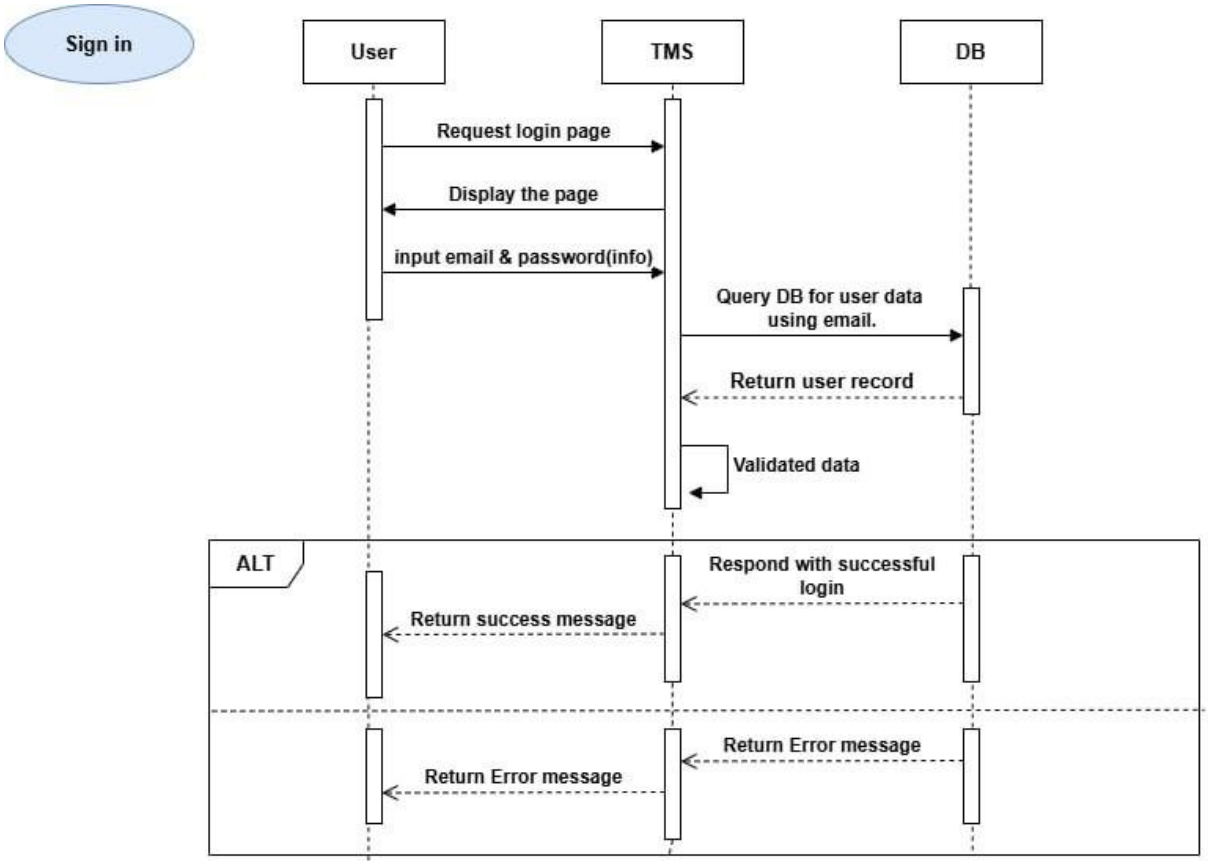


Table 8 sign in - usecaes specification

Use Case Name	**Sign in**
Actor	STM, STMe, Admin, Client
Precondition	The client must have access to the application or website.
Main Scenario	1. The user launches the application and opens the login page. 2. The user enters their email and password. 3. The user clicks on the "Sign In" button. 4. The system validates the entered credentials. The system logs the user in successfully.
Alternative Scenario	If the email or password is incorrect, the system displays an error message ("Invalid credentials. Please try again.") and does not log the user in.
Postcondition	The user is successfully logged into the system.

- **Account info management**

Figure 7 account info management - sequence diagram

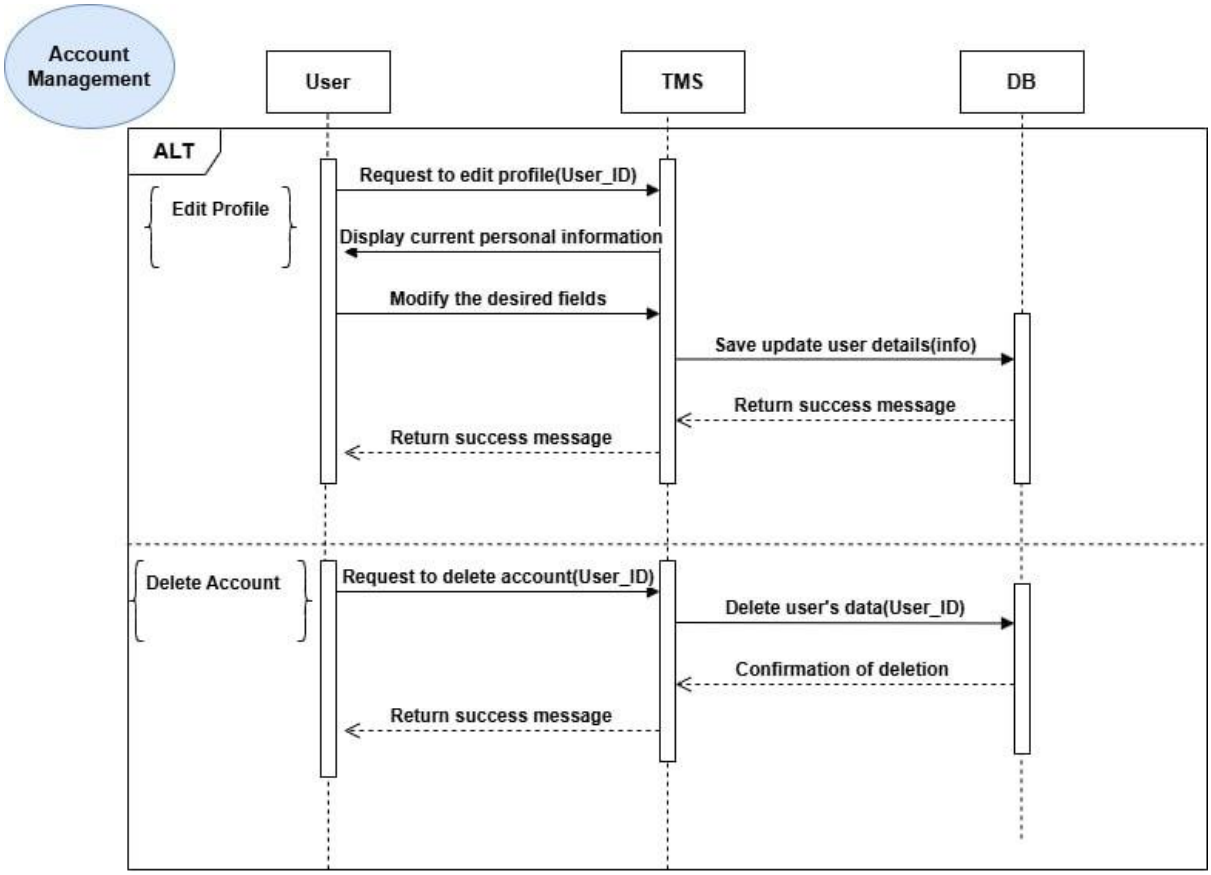


Table 9 Account info management - usecase specification

Use case name	**Account Info management**
Description	This use case allows users to manage their account information.
Actors	All users
Preconditions	user must be logged into the system.
Main Scenario	▪ User navigates to the **Profile information page.** **A. If User request to update its own account:** 1. The user navigates to the **Profile information** page. 2. The system displays the user's current personal information. 3. The user modifies the desired fields directly on the page. 4. The user clicks the **"Edit My Profile"** button to save the changes. 5. The system validates the updated information for correctness and completeness. 6. After successful validation, the system saves the updated information in the database. 7. The system displays a success message: **"Your account information has been updated successfully."** **B. If User request to delete its own account:** 1. The user navigates to the **Account Info Management** page. 2. The user clicks the **"Delete Account"** button. 3. The system deletes the user's account from the database. 4. The system logs the user out automatically. 5. The system displays a success message: **"Your account has been successfully deleted."**
Alternative Scenario	**Invalid Input:** 1. If the user enters invalid information (e.g., an incorrectly formatted email or missing required fields), the system highlights the errors and displays an error message. 2. The user revises the information and clicks the **"Edit My Profile"** button again to save the changes.
Postconditions	The user's account information is successfully updated or deleted based on the action performed.

- **Team management**

Figure8 team management - sequence diagram

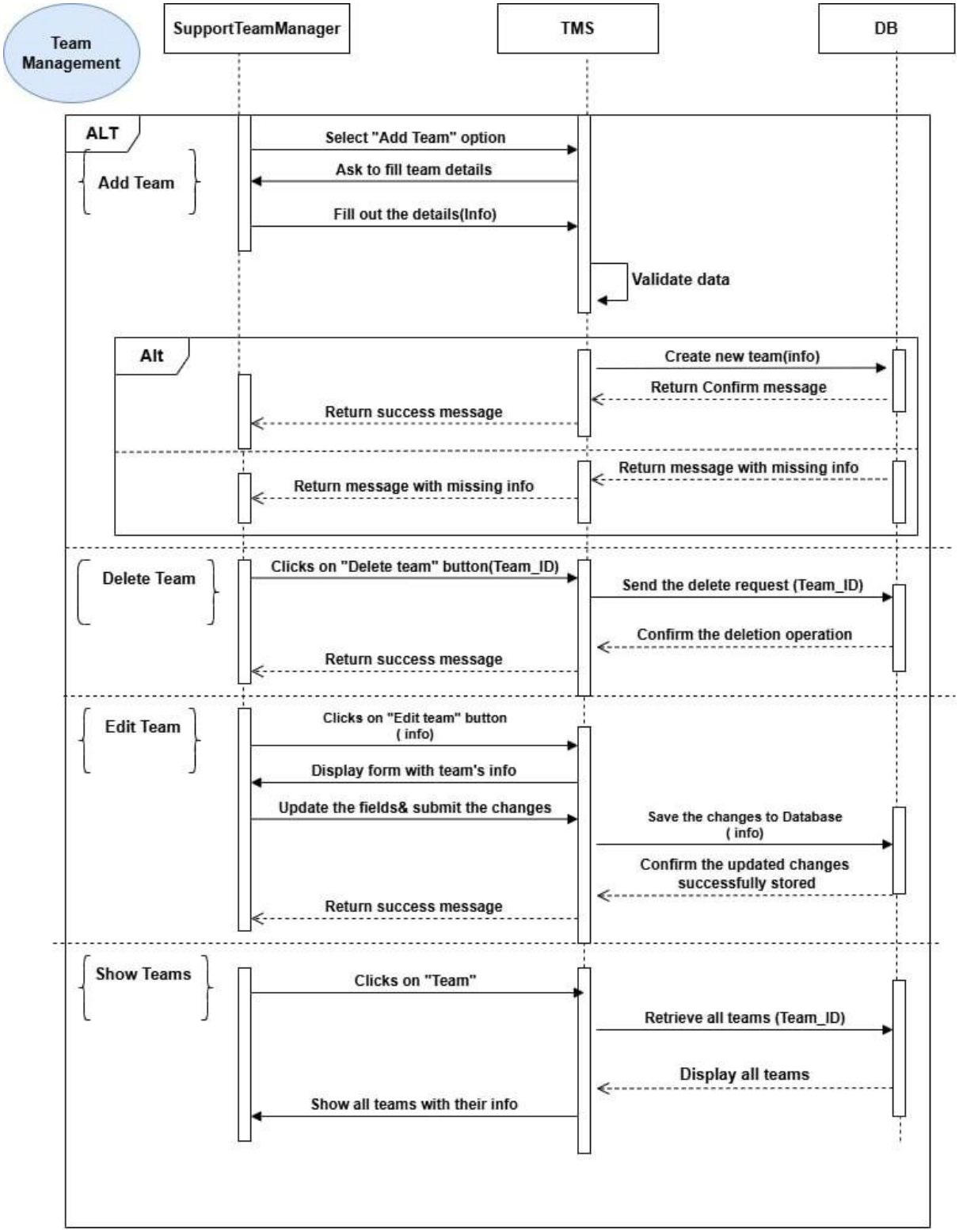


Table 10 Team management - usecase specification

Use case name	**Team management**
Description	This use case allows the support team manager to manage support team, including adding, deleting, updating statuses and viewing all teams.
Actors	Support Team manager (STM)
Preconditions	Support Team Manager must be logged into the system.
Main Scenario	▪ Support team manager navigate to the **Team management page.** **A. If STM request to add a new team:** 1. STM selects the "Add Team" option. 2. The system displays a form for entering new team details., such as category, and description. 3. Support Team Manager fills in the details and submits the form. 4. System validates the data. 5. After successful validation, the system creates the new team. 6. The updated team list is displayed, showing the newly added team &display success message. **B. If STM request to delete team:** 1. The STM clicks the "Delete Team" button on the dashboard. 2. The system requests to delete a specific team 3. The system displays a confirmation message to the STM 4. The STM confirms the deletion by selecting the "Confirm" option in the confirmation message. 5. The system deletes the in the database. **C. If STM request to update team status:** 1. STM selects the "Edit team" option 2. The system display form with the team's current information 3. Support Team Manager updates the desired fields. 4. System validates the updated information and saves changes. 5. The updated information is displayed in the team list. **D. If STM request to show teams:** 1. The STM clicks the "Team" in the side bar. 2. The system displays the list of all teams with their details.
Alternative Scenario	**(A1) Missing Fields:** 1. The flow starts after point [4.A, 4.C] of the main flow (A, B and C). 2. The system show that the field is missing. 3. The system requests from STM to complete the fields. 4. The flow returns to point 4 and the use case succeeds.
Postconditions	The team is successfully managed (added, deleted, updated, or showed teams).

Figure9 add new member - sequence diagram

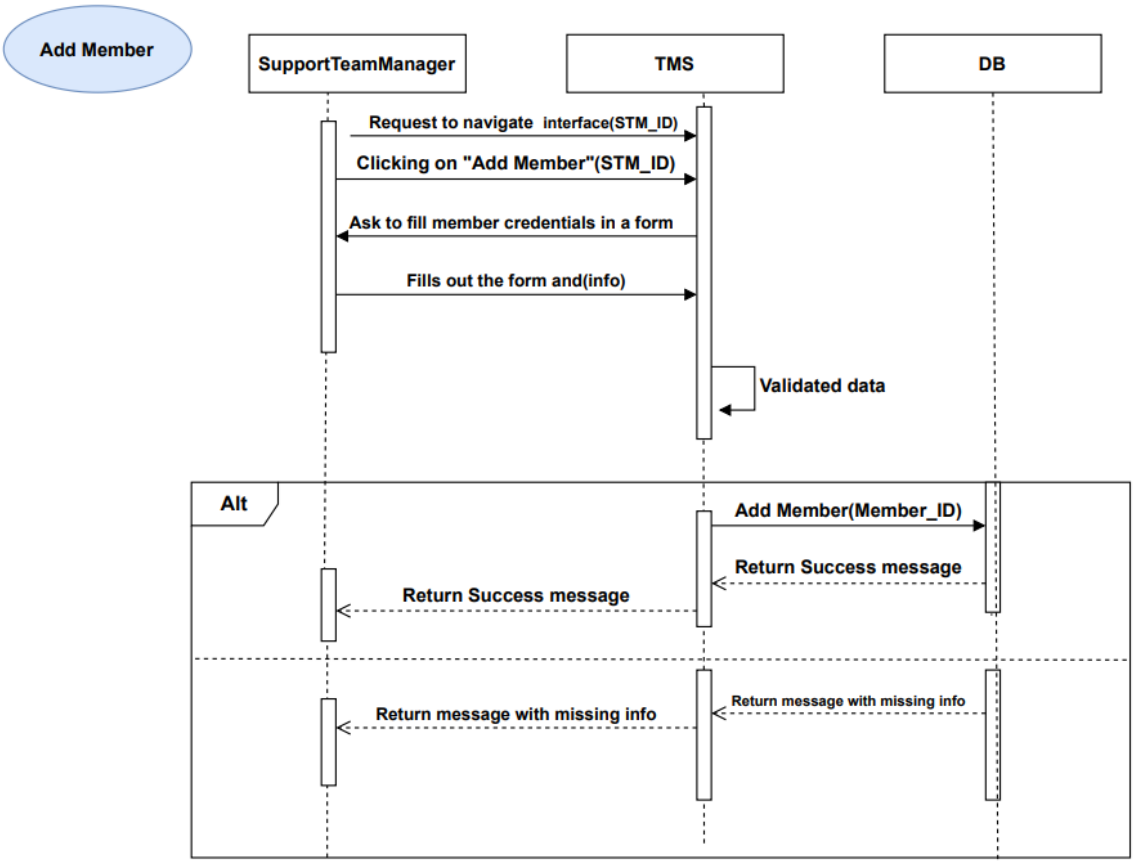


Table 11 Add new member - usecase specification

Use Case Name	**Add Member**
Actor	STM
Precondition	The Support Team Manager is logged into the system.
Main Scenario	1. The STM navigates to the interface 2. The STM clicks on the "Add Member" button. 3. The system displays a form with fields for the new member's details: **Name, Phone Number, Email, and Company**. 4. The STM fills out the form with the new member's details and submits it. 5. The system validates the entered data. 6. After successful validation, the system creates the new member and displays a success message: **"Team member added successfully."**
Postcondition	The new member is added to the system and is available to be assigned to a team.

Figure10 add member to team - sequence diagram

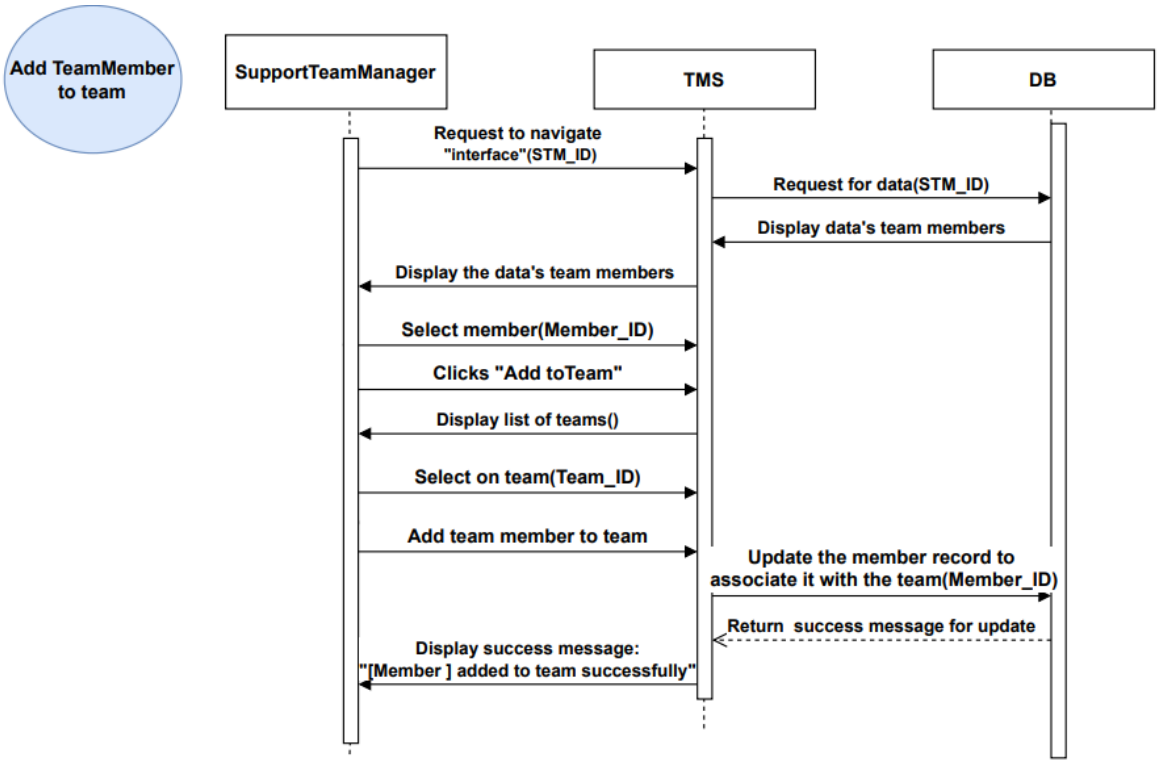


Table 12 add member to team - usecase specification

Use Case Name	**Add Member to Team**
Actor	Support Team Manager (STM)
Precondition	Support Team Manager is logged into the system, the new member must already exist in the system.
Main Scenario	1. The STM navigates to the interface 2. The STM selects the member to assign from the list of available members. 3. The STM clicks the "Add to Team" button. 4. The system displays a dropdown list of teams. 5. The STM selects the desired team from the dropdown list. 6. The STM clicks the "Confirm" button. 7. The system assigns the member to the selected team and updates the team member list. 8. The system displays a success message: **"Team member added to the team successfully."**
Postcondition	The selected member is successfully added to the chosen team.

Figure11 Change Status of Team Member - sequence diagram

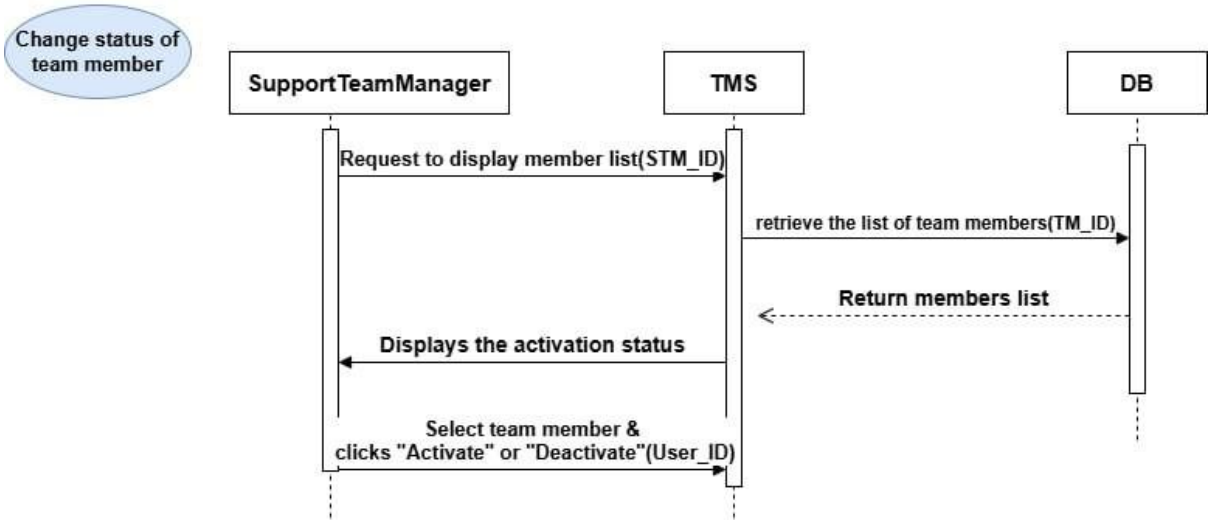


Table 13 Change Status of Team Member - usecase specification

Use Case Name	**Change Status of Team Members**
Actor	Support Team Manager (STM)
Precondition	The STM is logged into the system.
Main Scenario	1. The STM navigates to the interface. 2. The system displays the list of team members with their current activation status. 3. The Support Team Manager selects the team member and clicks the "Activate" switch. 4. The system updates the activation status of the selected team member. 5. A confirmation message is displayed, indicating the status change.
Postcondition	The selected team member's activation status is updated in the system database.

- **Ticket management**

Figure 12 create ticket - sequence diagram

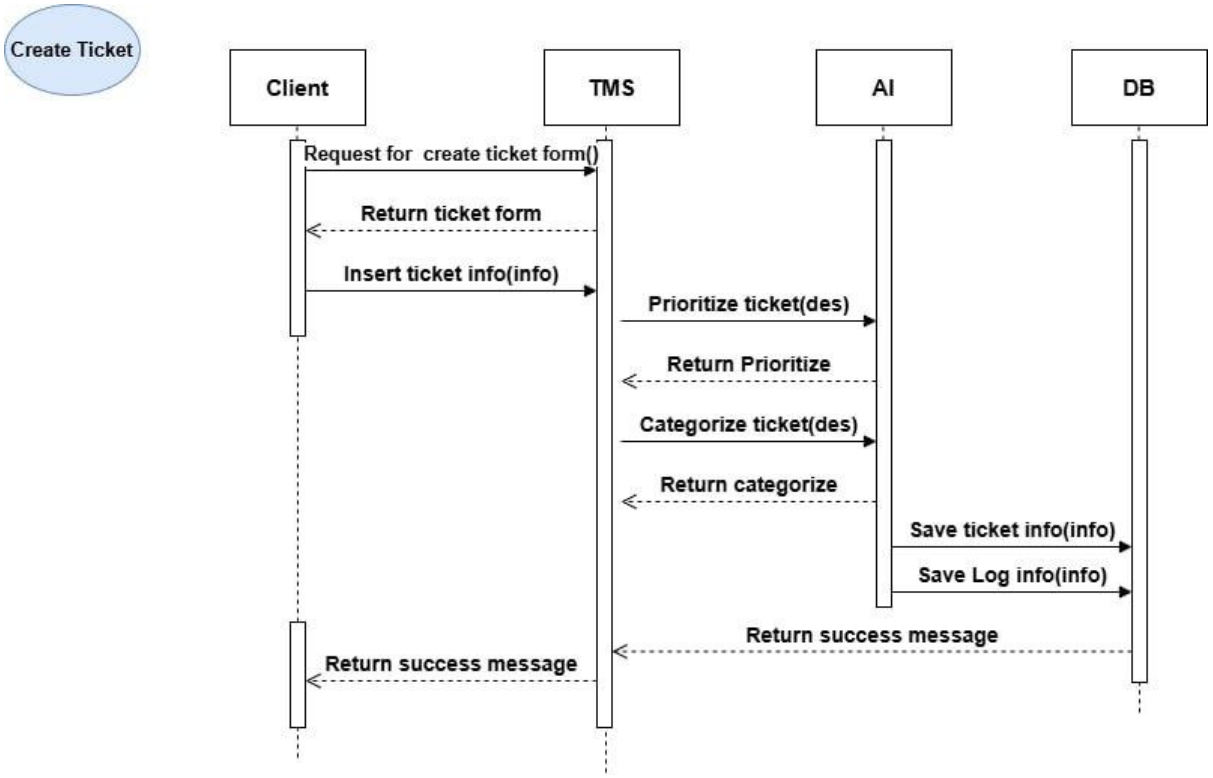


Table 14 create ticket - usecase specification

Use Case Name	**Create Ticket**
Actor	Client
Precondition	The client is logged into the system.
Main Scenario	1. Client requests to "Create Ticket." 2. System displays a form with fields for ticket details (title, description). 3. Client fills in the required fields and submits the form. 4. System saves the ticket, assigns it a unique ID, and provides a success message. The system automatically determines the tickets priority & categorized.
Alternative Scenario	**A1: Client submits the form with incomplete information.** 1. The flow starts after point **3** of the main flow. 2. The system show that the field is missing. 3. The System prompts the client to fill in the missing The flow returns to point **3** and the use case succeed.
Postcondition	A new ticket with a unique ID is saved in the system, and priority is assigned.

Figure13 assign ticket to team - sequence diagram

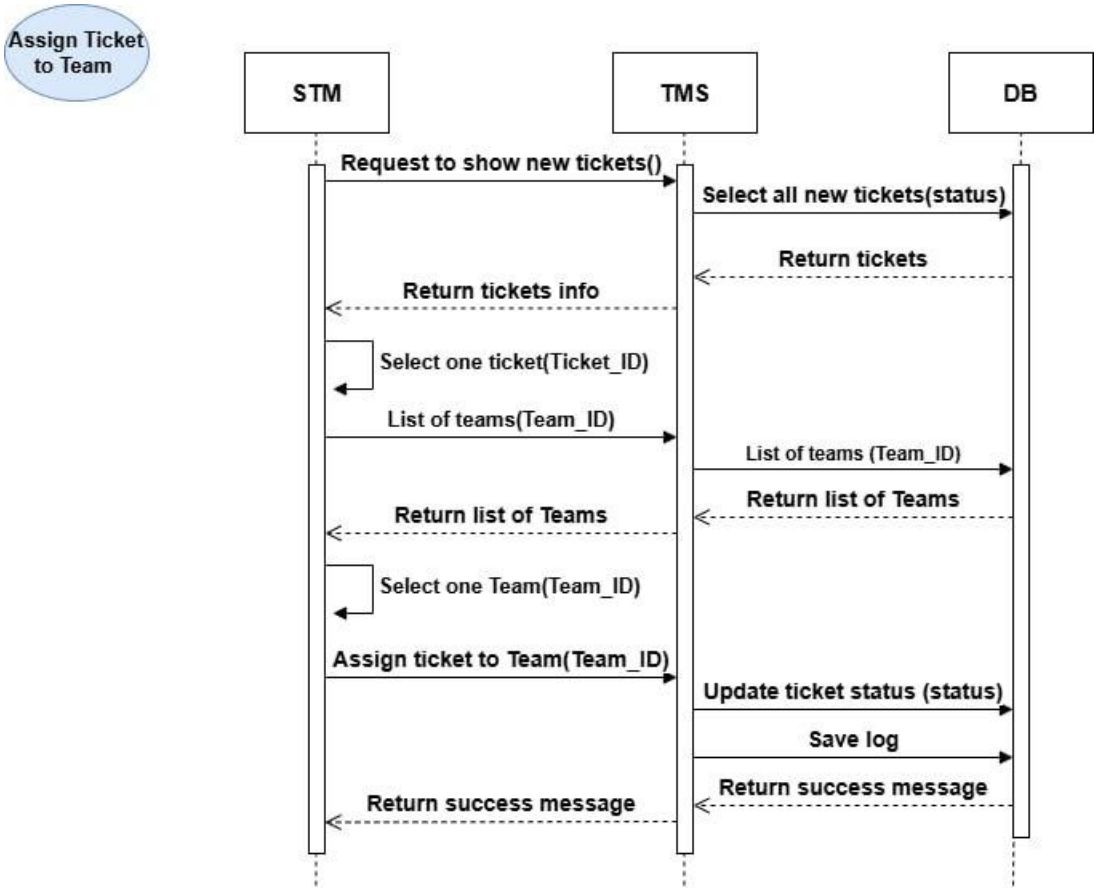


Table 15 Assign Ticket to Team - usecase specification

Use Case Name	**Assign Ticket to Team**
Actor	Support Team Manager (STM)
Precondition	The STM is logged into the system.
Main Scenario	1. STM selects a specific ticket to assign. 2. System displays a list of available support team. 3. STM selects a team to assign the ticket. 4. System updates the ticket's assigned team and displays a confirmation message.
Postcondition	The ticket is successfully assigned to a team.

Figure 14 show assigned tickets - sequence diagram

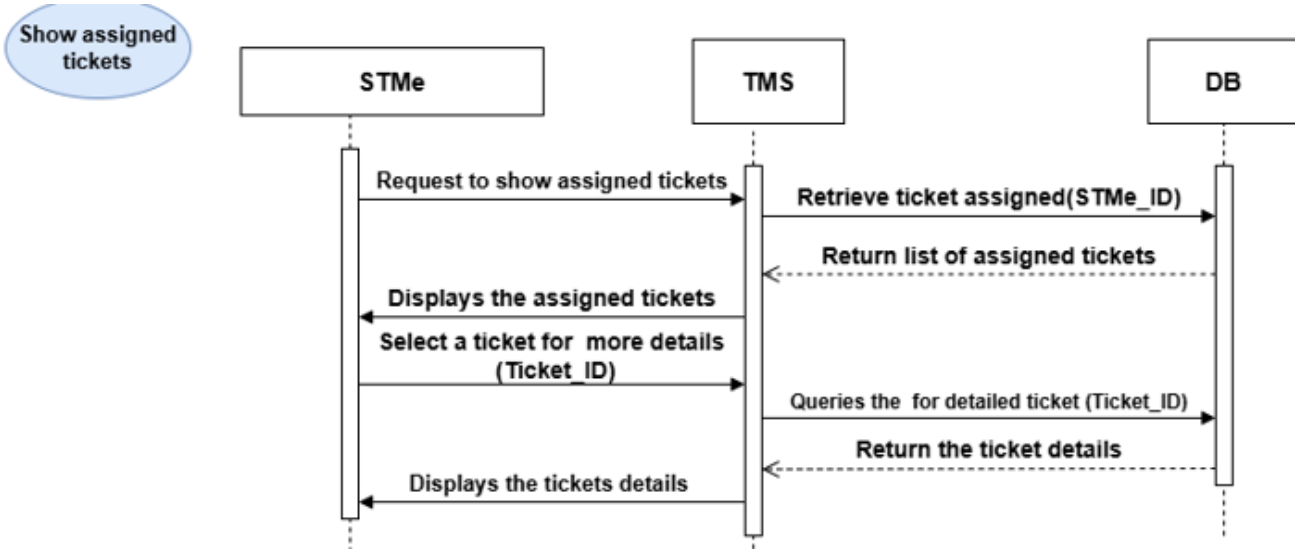


Table 16 Show Assigned Ticket - usecaes specification

Use Case Name	**Show Assigned Ticket**
Actor	Support Team Member
Precondition	Support team member is logged into the system.
Main Scenario	1. STMe request to "Show Assigned Tickets." 2. System filters and displays tickets assigned to the selected member. 3. STMe clicks on a ticket for more details. 4. System displays the details
Postcondition	The system successfully displays assigned tickets

Figure 15 solve ticket - sequence diagram

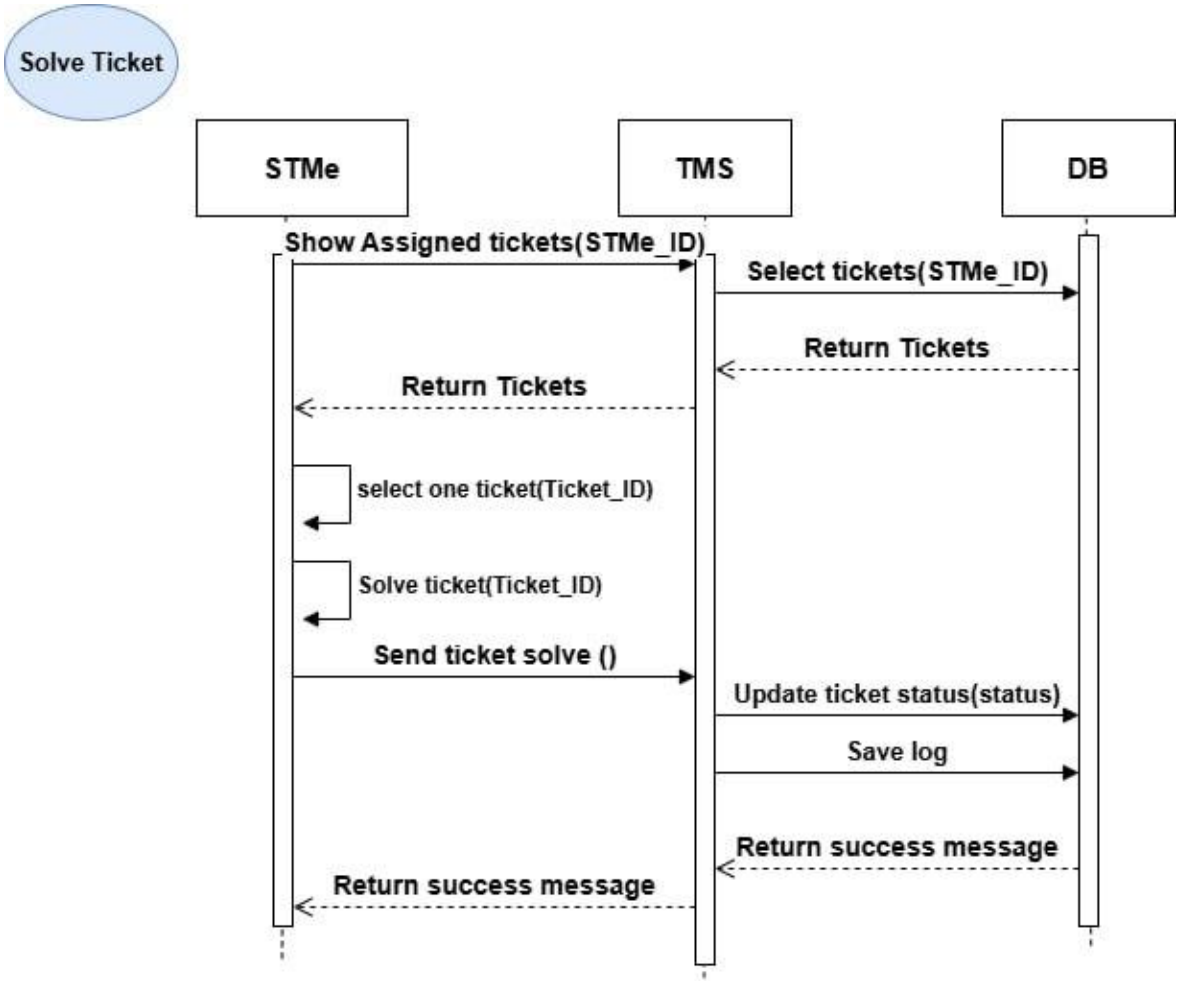


Table 17 solve ticket - usecase specification

Use Case Name	**Solve Ticket**
Actor	Support Team Member
Precondition	Support team member is logged into the system.
Main Scenario	1. Support team member request to the "show Ticket". 2. The system first displays the details of the assigned ticket. 3. System checks the current status of the selected ticket. 4. If the ticket status is "Open" or "In Progress," the system prompts the user to confirm the action to mark the ticket as resolved. After confirmation, the system updates the ticket status to "Resolved" and displays a success message.
Alternative Scenario	**A1: The selected ticket status is already set to "Resolved" or "Closed.".** 1. The system detects that the ticket is in Unmodifiable Status (e.g., "Resolved" or "Closed"). 2. The system displays a message: "This ticket is already resolved and cannot be updated further." 3. The system provides the user with options to: • View the details of the resolved ticket. • Return to the list of tickets. • Search for another ticket to work on. **3.1** If the user selects "View Details," the system displays the ticket's details, including the resolution status, resolution date, assigned team member, and any resolution comments. **3.2** If the user selects "Return to Ticket List," the system redirects them back to the list of all tickets. If the user selects "Search for Another Ticket," the system prompts the user to enter search criteria for a different ticket.
Postcondition	The ticket solved successfully.

Figure16 assign ticket to me - sequence diagram

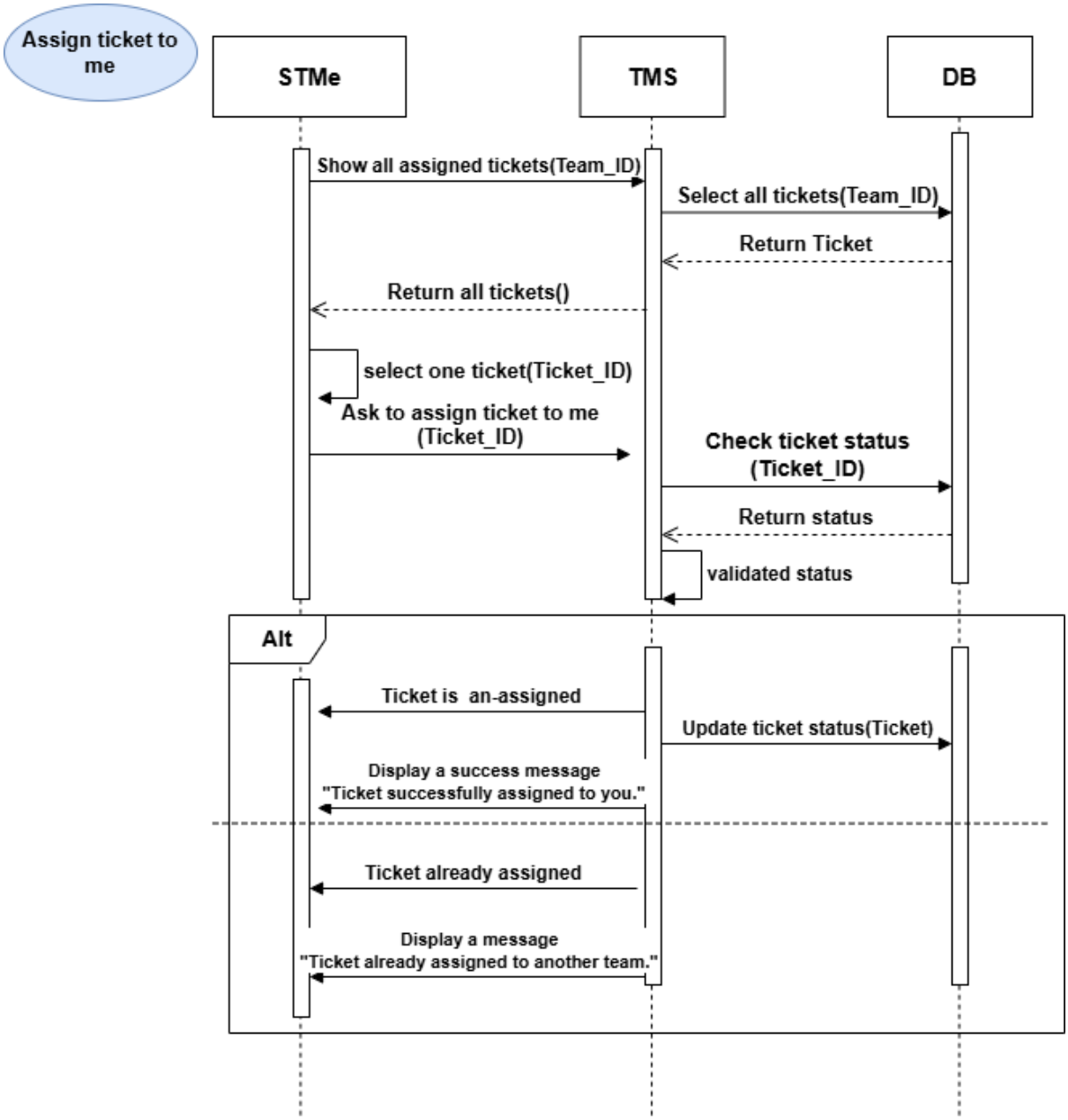


Table 18 Assign Ticket to me - usecase specification

Use Case Name	**Assign Ticket to me**
Actor	Support Team Member
Precondition	The **Support Team Member** is logged into the system.
Main Scenario	1. Support Team member request to view a list of tickets assigned to their team. 2. Support Team member selects an unassigned ticket from the list. 3. Support Team member requests to "Assign Ticket to Me." 4. System checks the status of the selected ticket to confirm it is unassigned. 5. If the ticket is unassigned, the System assigns the ticket to Support Team Member and updates the ticket's status. 6. System displays a confirmation message.
Alternative Scenario	**A1: Ticket already assigned** 1. The flow starts after step **4** of the main scenario. 2. System finds that the ticket is already assigned to another team member. 3. System displays a message to the Team Member indicating that the ticket is already assigned and cannot be reassigned. 4. The use case ends.
Postcondition	The ticket is assigned to the Support team member, and its status is updated in the system.

Figure 17 show ticket details - sequence diagram

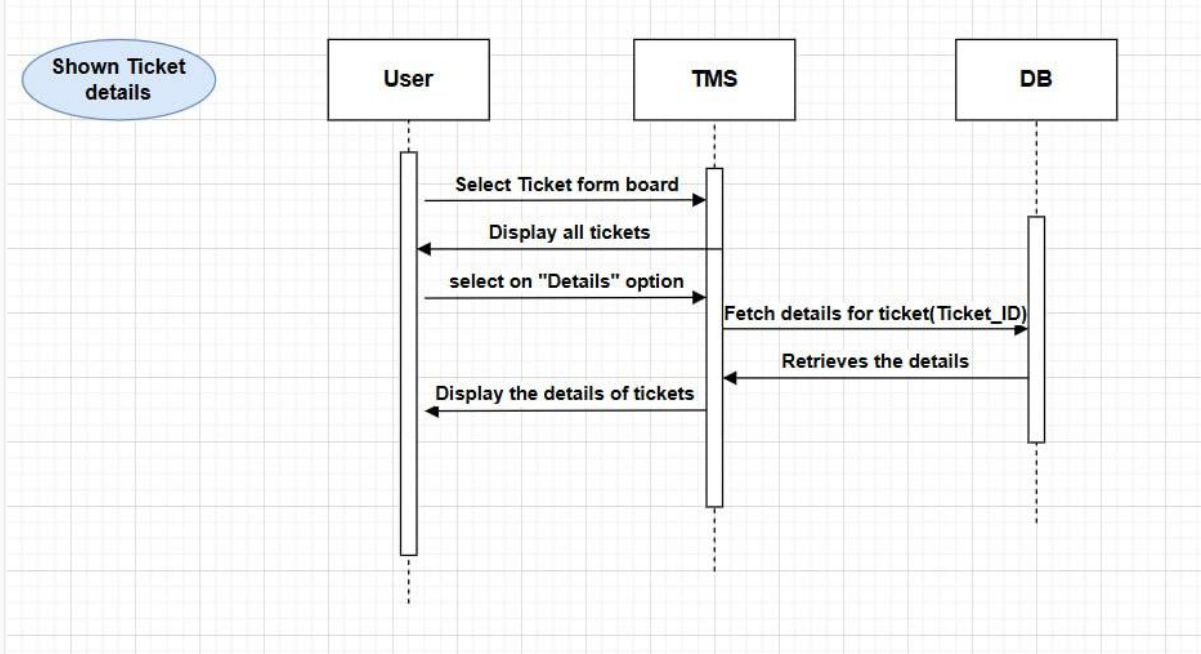


Table 19 show ticket details - usecase specification

Use Case Name	**Show Ticket details**
Actor	STM, STMe, client
Precondition	The user is logged into the system.
Main Scenario	1. The user navigates to the ticket interface. 2. The ticket board displays a list of tickets. 3. The user selects one ticket from the board. 4. The actor clicks on the "Details" button for the selected ticket. 5. The system retrieves the ticket details and displays them on the screen.
Postcondition	The system displays the detailed information of the selected ticket.

Figure 18 delete ticket - sequence diagram

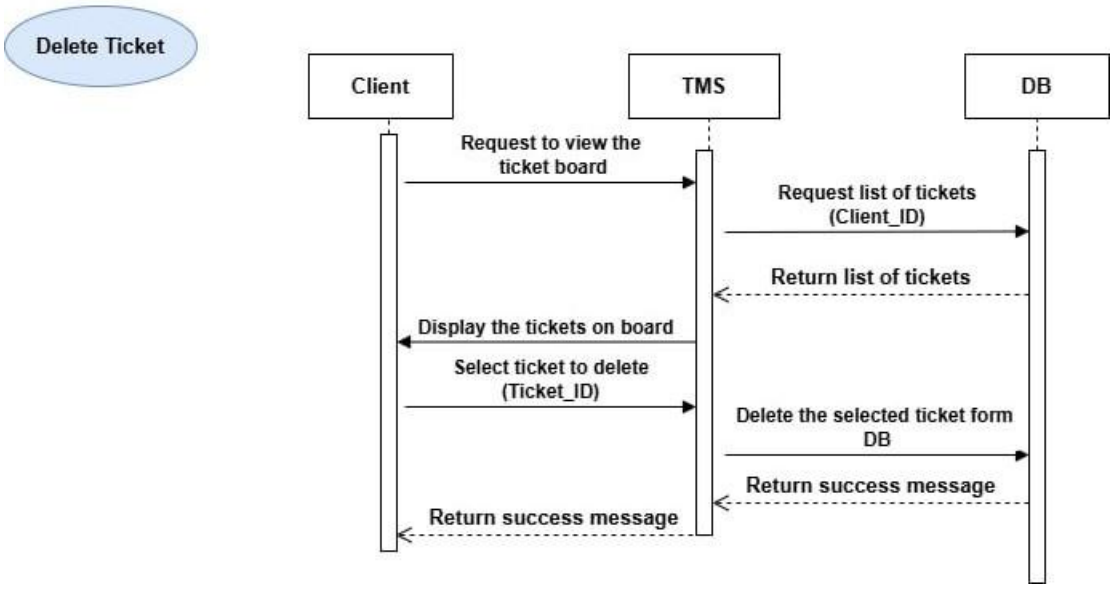


Table 20 delete ticket - usecase specification

Use Case Name	**Delete Ticket**
Actor	Client
Precondition	The client is logged into the system.
Main Scenario	1. The client views the ticket board. 2. The client selects a ticket from the board. 3. The client clicks on the "Delete" button. 4. The client confirms the deletion. 5. The system removes the ticket from the database. 6. The system displays a success message.
Postcondition	The Ticket is deleted successfully.

Figure 19 update ticket - sequence diagram

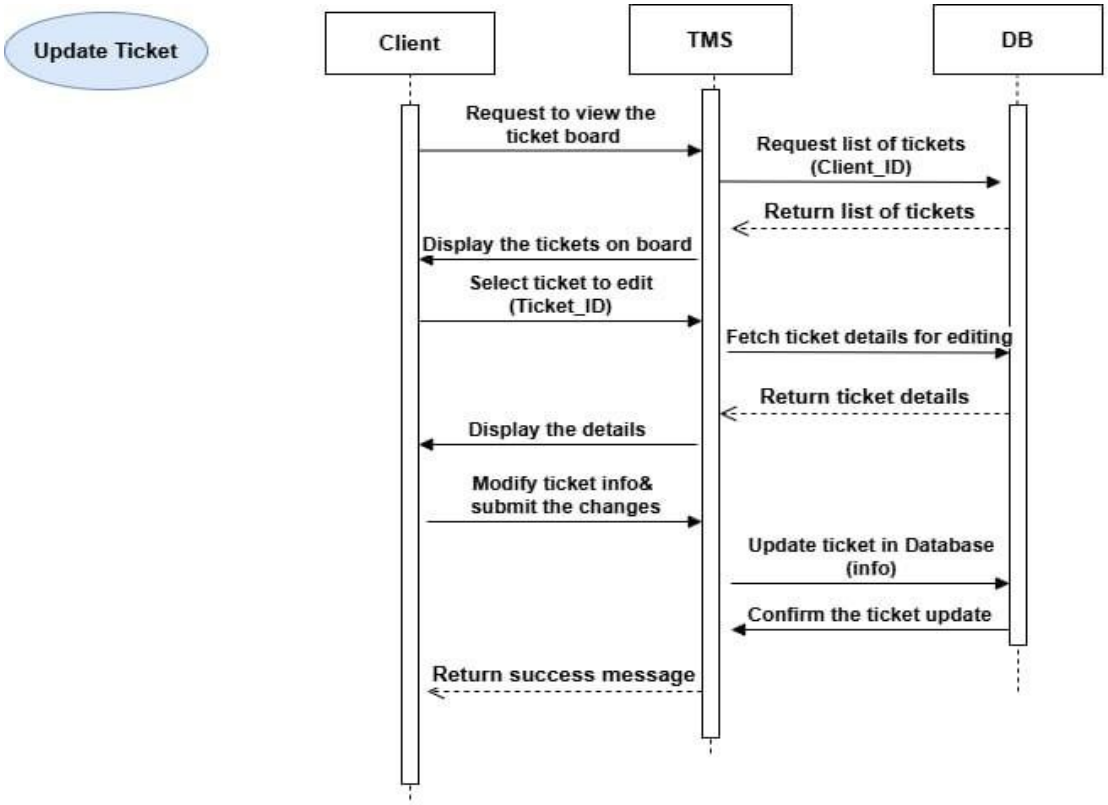


Table 21 update ticket usecase - specification

Use Case Name	**Update Ticket**
Actor	Client
Precondition	The client is logged into the system.
Main Scenario	1. The client launches the website and navigates to the "My Team Ticket" interface. 2. The client request to views the ticket. 3. The system displays the tickets on the board. 4. The client selects a ticket from the board. 5. The client clicks on the "Edit" button. 6. The client modifies the ticket. 7. The system validates the input and updates the ticket in the database. The system displays a success message.
Alternative Scenario	**A1: Missing fields** 1. The client modifies the ticket information but leaves required fields blank. 2. The system detects the missing fields. 3. The system displays an error message specifying the required fields. 4. The ticket is not updated, and the client is prompted to correct the errors
Postcondition	The Ticket is updated successfully.

Figure20 change ticket status - sequence diagram

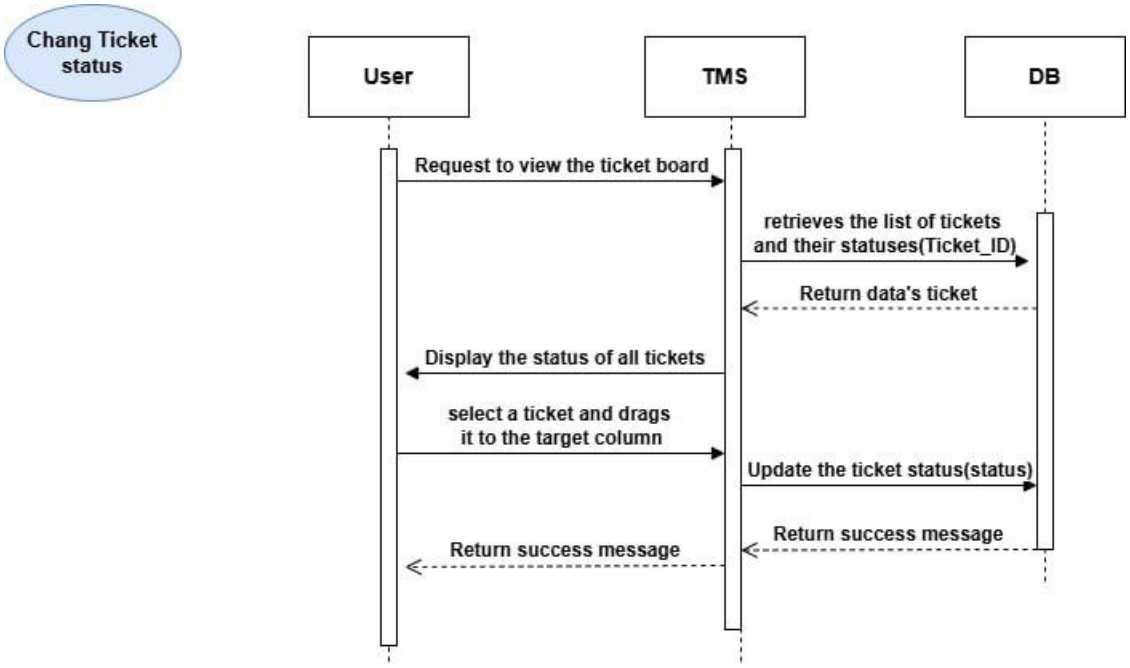


Figure 21 change ticket status - usecase specification

Use Case Name	**Change Ticket Status**
Actor	STM, STMe,
Precondition	The STM , STMe is logged into the system.
Main Scenario	1. The user navigates to the ticket management interface. 2. The user selects a specific ticket from the list of tickets. 3. The actor clicks and drags the ticket from the **current status column** 4. The actor drops the ticket into the **target column** 5. The system updates the ticket's status in the database. A success message is displayed to the user.
Postcondition	The ticket is moved to the new column, and the updated status is visible to all users.

- **Company management**

Figure22 company management - sequence diagram

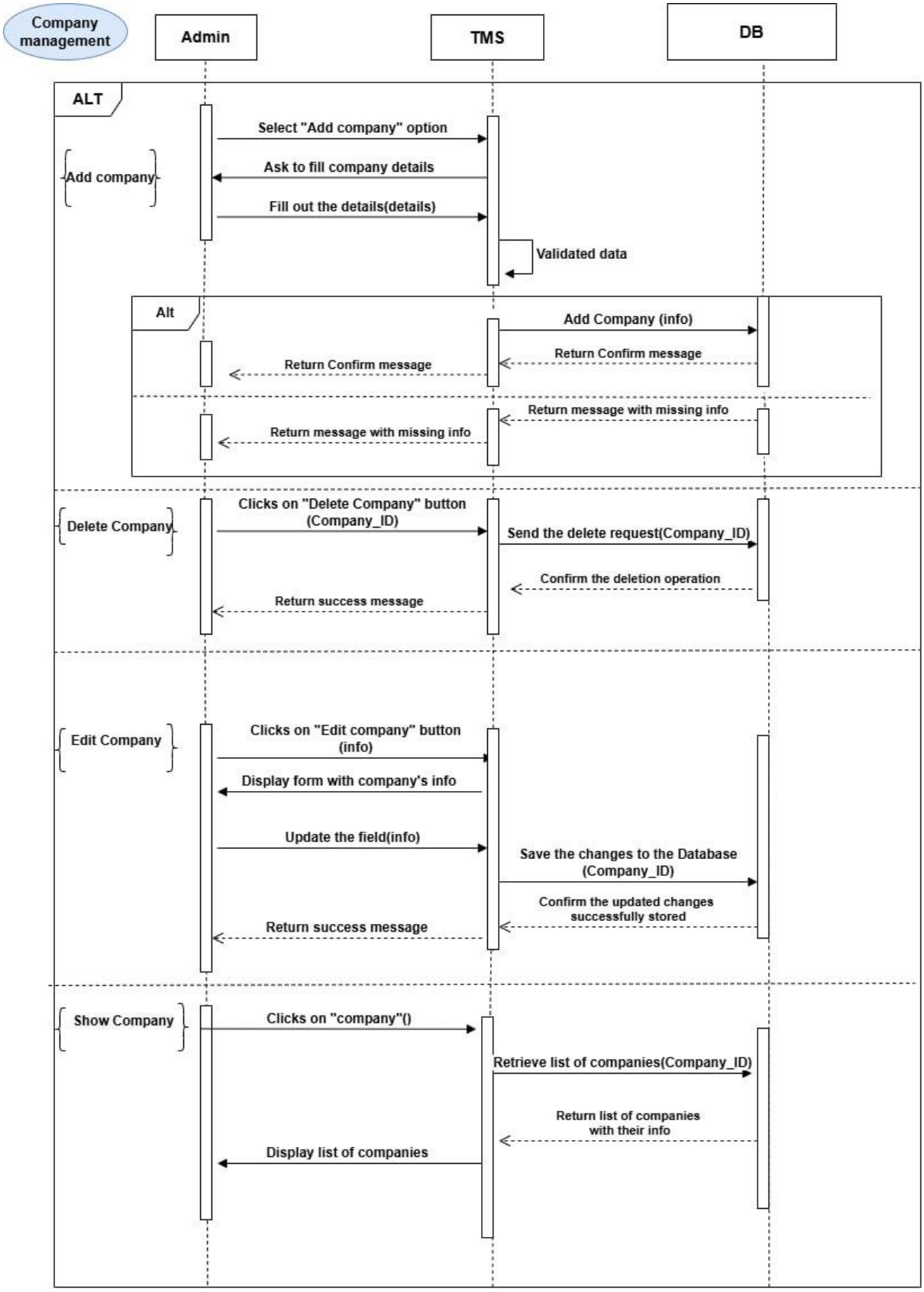


Table 22 Company management usecase - specification

Use case name	**Company management**
Description	This use case allows the admin to manage company, including adding, and deleting, updating, viewing all companies
Actors	Admin
Preconditions	Admin must be logged into the system.
Main Scenario	▪ Admin navigate to the Company management page. **E. If Admin request to add a new company:** 1. Admin selects the “Add company” option. 2. The system displays a form for entering new company details, such as ID, name, address. 3. Admin fills in the details and submits the form. 4. System validates the data. 5. After successful validation, the system creates the new company. 6. The system displays success message. **F. If Admin request to delete company:** 1. The admin clicks the "Delete company" button on the dashboard. 2. The system requests to delete a specific company 3. The system displays a confirmation message to the admin 4. The admin confirms the deletion by selecting the "Confirm" option in the confirmation message. 5. The system deletes the designated company from the company list in the database. **G. If STM request to update company info:** 1. Admin selects the “Edit company” option 2. The system display form with the company’s current information 3. Admin updates the desired fields. 4. System validates the updated information and saves changes. 5. The updated information is displayed in the company list. **H. If Admin request to show company:** 1. The admin clicks the "Company" component in the side bar. 2. The system displays the list of all companies with their details.
Alternative Scenario	**(A1) Missing Fields:** 1. The flow starts after point [4.A, 4.C,] of the main flow. 2. The system show that the field is missing. 3. The system requests from Admin to complete the fields. 4. The flow returns to point 4 and the use case succeeds.
Postconditions	The company is successfully managed.

- **Comment management**

Figure23 comment management - sequence diagram

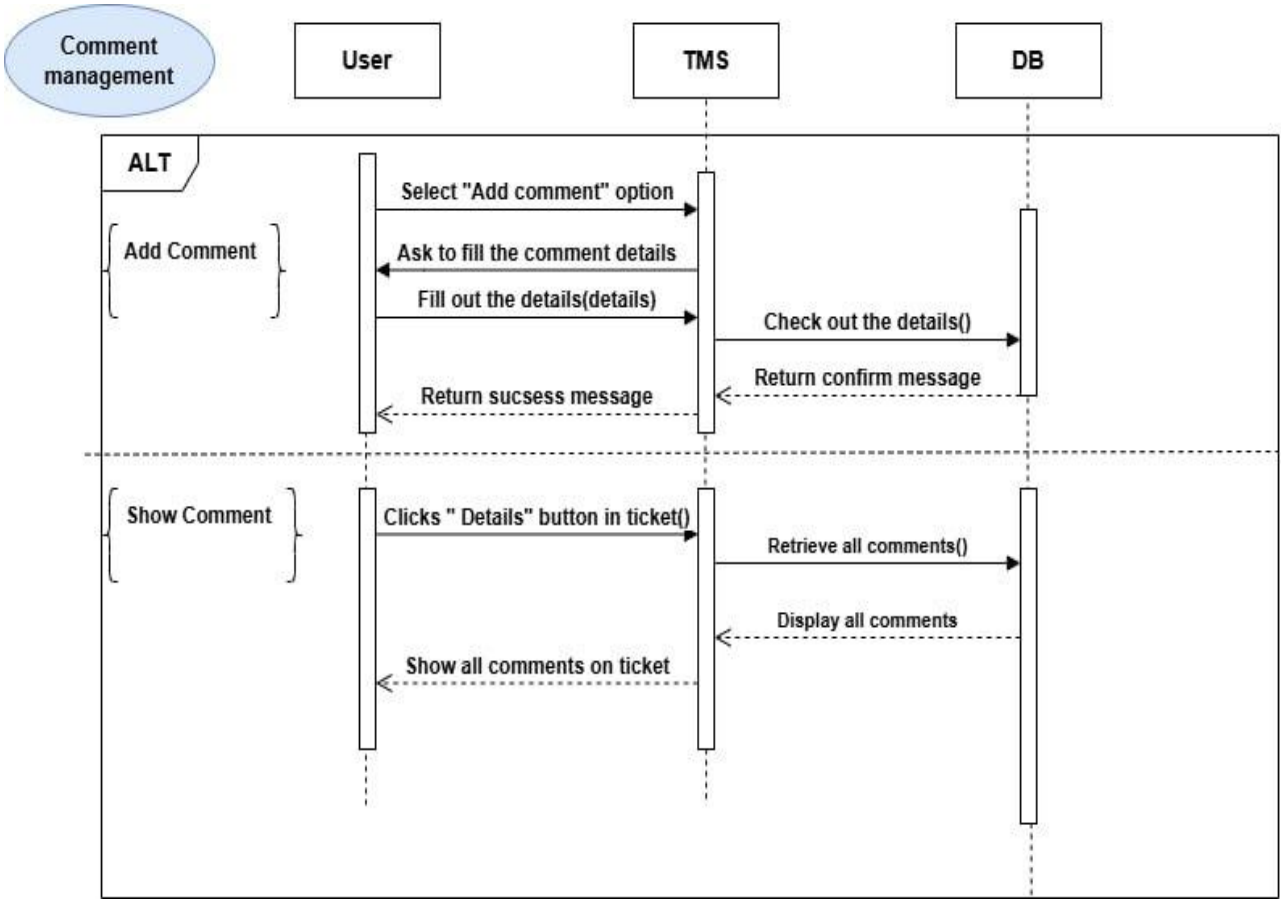


Table 23 Comment management - usecase specification

Use case name	**Comment management**
Description	This use case allows the **Client, Support Team Manager, and Support Team Member** to manage comment, including adding, and viewing all comments.
Actors	Client, Support Team Manager, Support Team Member
Preconditions	User must be logged into the system.
Main Scenario	▪ User navigate to the Comment management page. **A. If User request to add a new comment:** 1. Admin selects the "Add comment" option. 2. The system displays a form for entering new comment details, such as description. 3. User fills in the details and submits the form. 4. System validates the data. 5. After successful validation, the system creates the new comment. 6. The system displays success message. **B. If Admin request to show comments:** 1. The User clicks the "Details" button in the ticket. 2. The system displays all comments.
Postconditions	The comment is successfully managed.

- **Workflow management**

Figure24 choose default workflow - sequence diagram

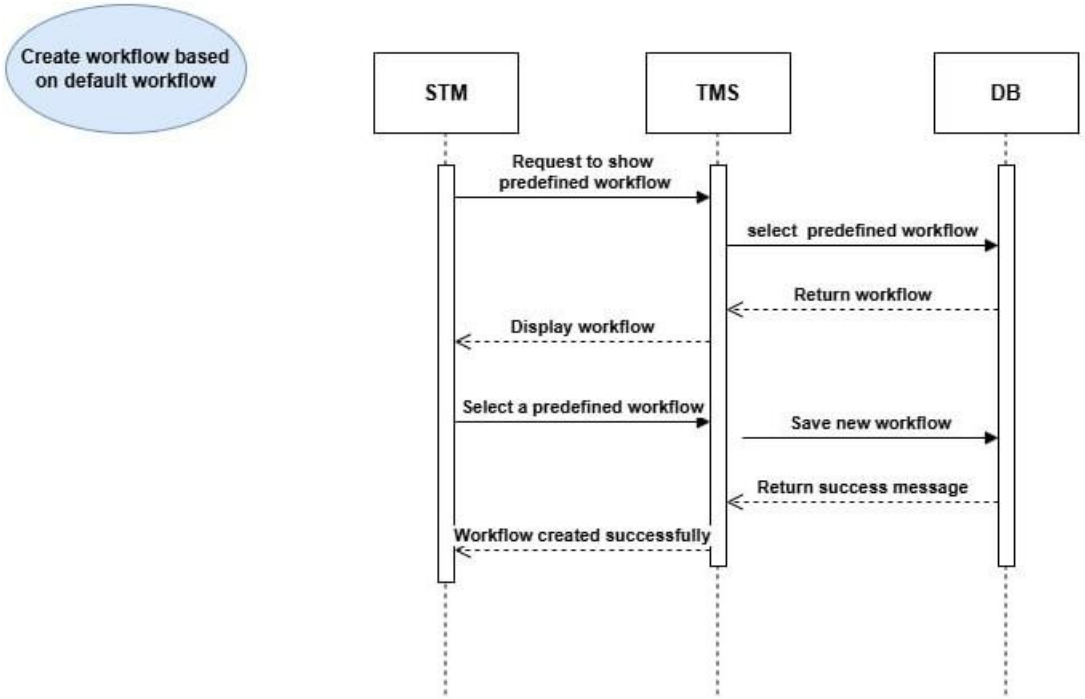


Table 24 choose default workflow - usecase specification

Use Case Name	**Create workflow based on default workflow**
Actor	STM
Precondition	The STM must be logged into the system.
Main Scenario	1. The STM request to show predefined workflow. 2. The system displays the predefined workflow. 3. The STM select specific workflow. 4. The system displays a form pre-filled with details from the selected predefined workflow. 5. The STM reviews and modifies the workflow details as needed. 6. The STM submits the modified workflow. 7. The system creates the new workflow based on the predefined workflow and displays a success message: "Workflow created successfully.
Postcondition	The predefined workflow was created

Table25 Create workflow from scratch - sequence diagram

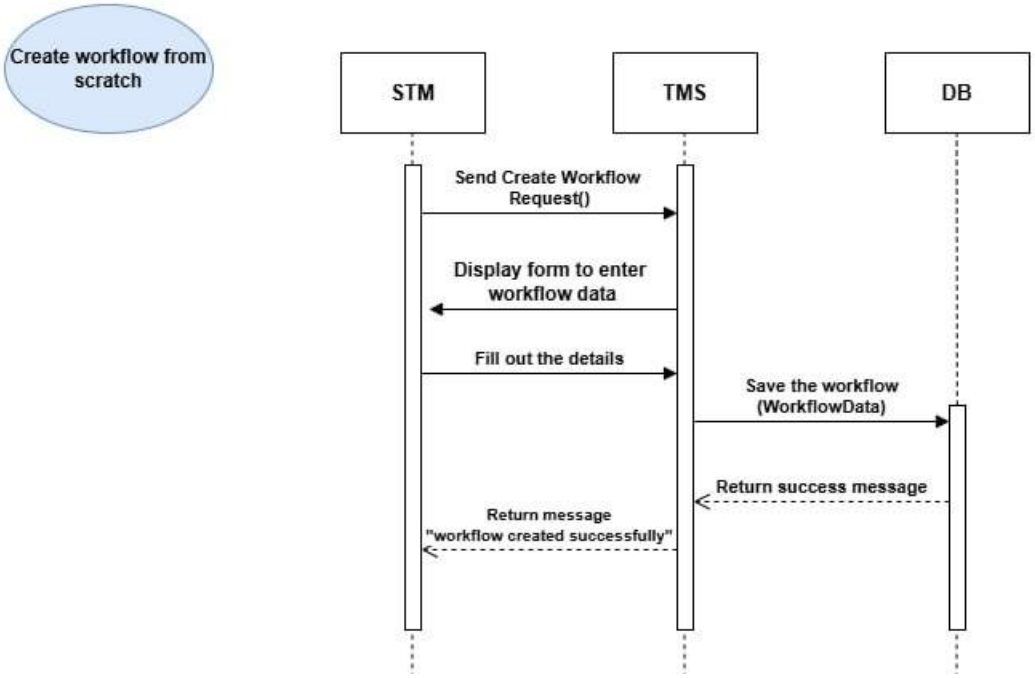


Table 26 Create workflow from scratch - usecase specification

Use Case Name	**Create workflow from scratch**
Actor	STM
Precondition	The STM must be logged into the system.
Main Scenario	1. The STM navigates to the interface. 2. The STM clicks on the "Create workflow" button. 3. The system displays a form with fields for the new workflow details. 4. The STM fills out the form with the new workflow details and submits it. 5. The system validates the entered data. 6. After successful validation, the system creates the new workflow and displays a success message: **"workflow created successfully."**
Postcondition	The new workflow was created

4.2 Analysis Class Diagram

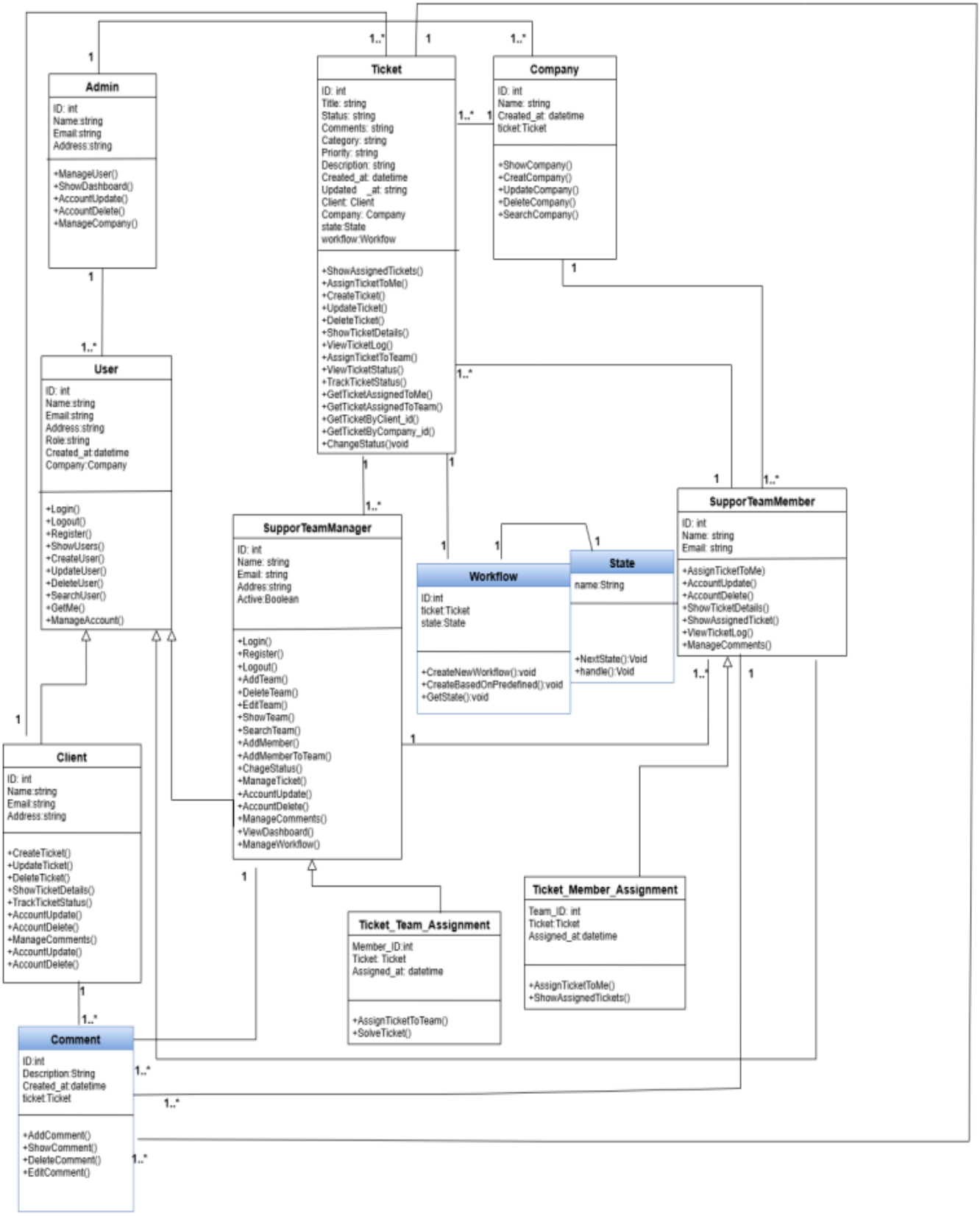


Figure 25 Class Diagram

4.3ERD Diagram

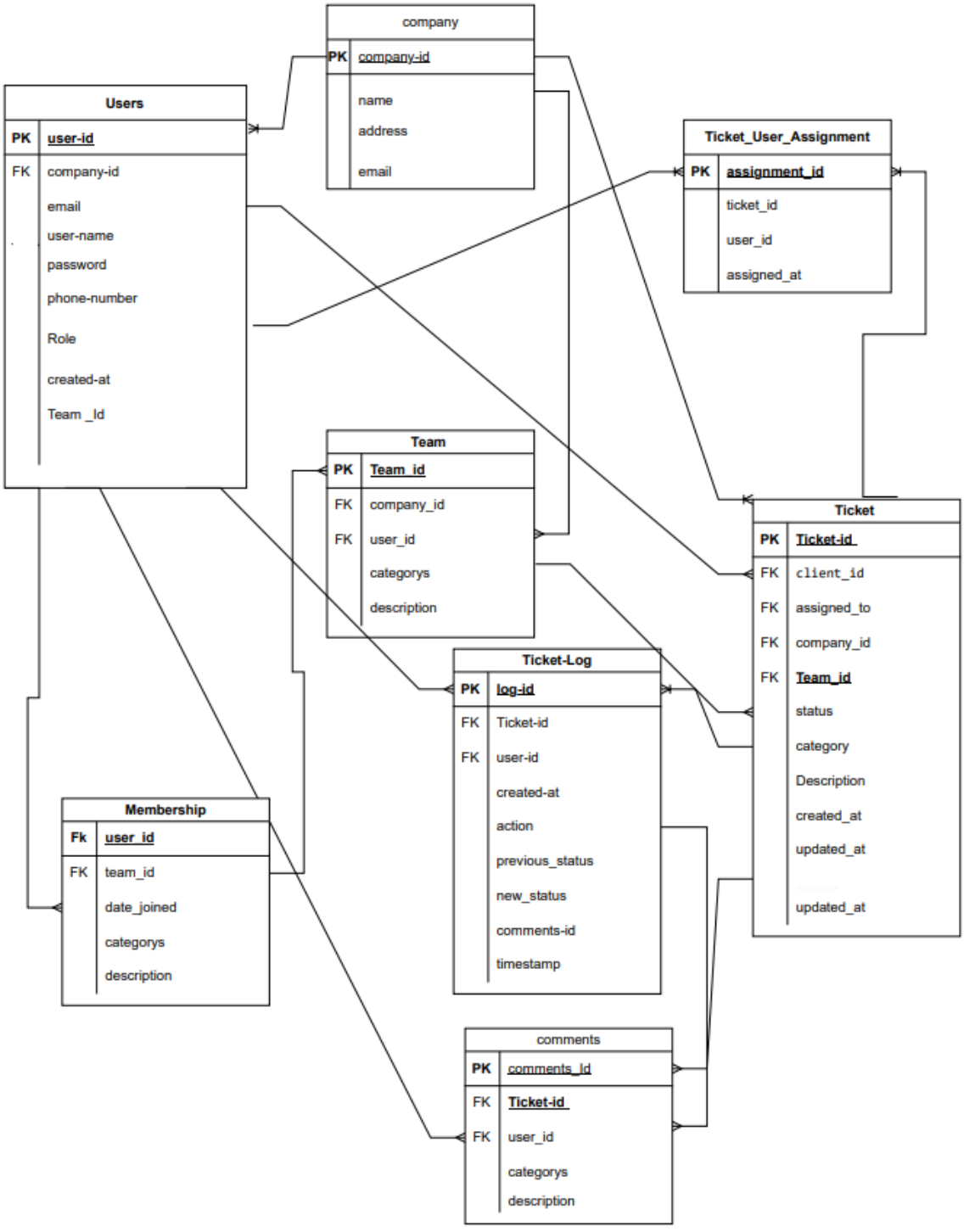


Figure 26 ERD diagram

6. Initial Test Cases

Table 27 Test cases for User Management by Admin

Test Case Scenario:	**User Management by Admin**		
Test case id	Test case title	Test steps	Expected result
Tc-01	Check results on display the user's dashboard.	1. Launch the website 2. Launch the user's dashboard interface.	Display user's dashboard
Tc-02	Check results on add new user (STM)	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on add user button 4. insert user info	User added successfully
Tc-03	Check results on add new user (STM) but user already exist	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on add user button 4. insert user info	Error message user already exist with this email
Tc-04	Check results on add new user with Missing Fields	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on add user button 4. insert user info	Error message fields required
Tc-05	Check results on add new user with invalid information	1. Launch the website 2. Launch the user's dashboard interface. 3. click on add user button 4. insert user info	Error message invalid information
Tc-06	Check results on update user info	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on Edit button 4. update user info	User edited successfully
Tc-07	Check results on update user info with invalid information	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on Edit button 4. update user info	Error message invalid information
Tc-08	Check results on delete user	1. Launch the website. 2. Launch the user's dashboard interface. 3. click on delete user button 4. confirm delete	User deleted successfully

Table 28 Test cases for Account Info management

Test Case Scenario:	**Account Info management**		
Test case id	Test case title	Test steps	Expected result
Tc-09	Check results on update account info.	1. Launch the website. 2. Launch the profile interface. 3. Display user info 4. Click on Edit My profile	user info updated successfully
Tc-10	Check results update account info with invalid information	1. Launch the website. 2. Launch the profile interface. 3. Display user info 4. Click on Edit My profile	Error message invalid information
Tc-11	Check results on update account info without any changes	1. Launch the website. 2. Launch the profile interface. 3. Display user info 4. Click on Edit My profile	message No changes made

Table 29 Test cases for Company management by admin

Test Case Scenario:	**Company management by admin**		
Test case id	Test case title	Test steps	Expected result
Tc-12	Check results on display the Companies dashboard of the admin.	1. Launch the website by the system admin. 2. Launch the Companies dashboard interface.	Display Companies dashboard
Tc-13	Check results on add new Company	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on add Company button 4. insert Company info	Company added successfully
Tc-14	Check results on add new Company but Company already exist	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on add Company button 4. insert Company info	Error message Company already exist with this email
Tc-15	Check results on add new Company with Missing Fields	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on Company user button 4. insert Company info	Error message fields required
Tc-16	Check results on add new Company with invalid information	1. Launch the website 2. Launch the Companies dashboard interface. 3. click on add Company button 4. insert Company info	Error message invalid information

Test Case Scenario:		**Company management by admin**	
Test case id	Test case title	Test steps	Expected result
Tc-17	Check results on update Company info	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on Edit button 4. update Company info	Company edited successfully
Tc-18	Check results on update Company info with invalid information	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on Edit button 4. update Company info	Error message invalid information
Tc-19	Check results on delete Company	1. Launch the website. 2. Launch the Companies dashboard interface. 3. click on delete button 4. confirm delete	Companies deleted successfully

Table 30 Test cases for Team management by STM

Test Case Scenario:		**Team management by STM**	
Test case id	Test case title	Test steps	Expected result
Tc-20	Check results on add new team.	1. Launch the website. 2. Launch the Teams interface. 3. Display Team dashboard 4. Click on add new team 5. Insert team info	Team info updated successfully
Tc-21	Check results on add new Team with Missing Fields	1. Launch the website. 2. Launch the Teams dashboard interface. 3. click on add Team button 4. insert Team info	Error message fields required
Tc-22	Check results on update Team info	1. Launch the website. 2. Launch the Teams dashboard interface. 3. click on Edit button 4. update Team info	Team edited successfully
TC-23	Check results on delete Team	1. Launch the website. 2 Launch the Teams dashboard interface. 3. click on delete Team button 4. confirm delete	Team deleted successfully

Table 31 Test cases for Team Members management by STM

Test Case Scenario:	**Team Members management by STM**		
Test case id	Test case title	Test steps	Expected result
Tc-24	Check results on add new team Member.	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on add new Team Member 5. Insert Team Member info	Team Member info updated successfully
Tc-25	Check results on add new team Member but team Member already exist	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on add new Team Member 5. Insert Team Member info	Error message user already exist with this email
Tc-26	Check results on add new team Member with Missing Fields	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on add new Team Member 5. Insert Team Member info	Error message fields required
Tc-27	Check results on add new team Member with invalid information	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on add new Team Member Insert Team Member info	Error message invalid information
Tc-28	Check results on add team member to team	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on Add Member To Team 5. Select a team from the selection list. 6. Add member to team	Member added successfully
Tc-29	Check results on update team for team member	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on Edit 5. Select a team from the selection list. 6. Add member to new team	Members team updated successfully
Tc-30	Check results on Change status for team member	1. Launch the website. 2. Launch the Team Members interface. 3. Display Team Members dashboard 4. Click on Change status switch	Status updated successfully

Table 32 Test cases for Ticket management

Test Case Scenario:		**Ticket management**	
Test case id	Test case title	Test steps	Expected result
Tc-31	Check results on show Ticket Details.	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on Details button	Display Ticket Details successfully
Tc-32	Check results on add comment on Ticket	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on comment button 6. Add comment	comment added successfully
Tc-33	Check results on see all comments on Ticket	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on comment button	Display Ticket comments Successfully
Tc-34	Check results on Assign Ticket to Team Transaction by STM	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. drag ticket from open column and drop it in Assign to team column 6. choose team from selection list 7. confirm the transaction	Ticket change status and moved to assign to team column
Tc-35	Check results on Assign to members or in progress or done Transaction by STM	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. drag ticket from open column and drop it in Assign to members or in progress or done column	The system rejects this Transaction.
Tc-36	Check results on Assign Ticket to me Transaction by STMe in My Team Ticket interface	1. Launch the website. 2. Launch the My Team Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. drag ticket from Assign to Team column and drop it in Assign to Member column 6. confirm the transaction	Ticket change status and moved to assign to Member column
Tc-37	Check results on from assign to member to In Progress or Done Transaction by STMe in My Team Ticket interface	1. Launch the website. 2. Launch the My Team Ticket interface. 3. Display Ticket board 4. choose on ticket from board 5. drag ticket from Assign to Member column and drop it in Assign to In Progress or Done column	The system rejects this Transaction

Test Case Scenario:		Ticket management	
Test case id	Test case title	Test steps	Expected result
Tc-38	Check results on from assign to team to In Progress Transaction by STMe in Ticket interface	1. Launch the website. 2. Launch the My Team Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. drag ticket from Assign to Member column and drop it in In Progress column 6. confirm the transaction	Ticket change status and moved to In Progress column
Tc-39	Check results on from assign to team to Done Transaction by STMe in Ticket interface	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. drag ticket from Assign to Member column and drop it in Done column	The system rejects this Transaction
Tc-40	Check results on from In Progress to Done Transaction by STMe in Ticket interface	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose on ticket from board 5. drag ticket from In Progress column and drop it in Done column 6. confirm the transaction	Ticket change status and moved to Done column
Tc-41	Check results on add new Ticket by Client	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. Click on Add new Ticket button 5. Insert Ticket info	Ticket added successfully
Tc-42	Check results on add new Ticket with Missing Fields By Client	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. Click on Add new Ticket button 5. Insert Ticket info	Error message fields required
Tc-43	Check results on update Ticket info By Client	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on edit button 6. Edit Ticket info	Ticket edited successfully
Tc-44	Check results on update Ticket info Missing Fields By Client	1. Launch the website. 2. Launch the My Team Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on edit button 6. Edit Ticket info	Error message fields required
Tc-45	Check results on delete Ticket by Client	1. Launch the website. 2. Launch the Ticket interface. 3. Display Ticket board 4. choose one ticket from board 5. Click on delete button 6. confirm delete	Ticket deleted successfully
Tc-46	Check results on show Ticket Activity logs by STM and STMe.	1. Launch the website. 2. Launch the Activity logs interface. 3. Display Activity logs dashboard	Display Activity logs dashboard

Table 33 Test cases for Sign in Functionality

Test Case Scenario:		**Check Sign in Functionality.**	
Test case id	Test case title	Test steps	Expected result
Tc-47	Check results on entering a valid email and password.	1. Launch the application on the login page. 2. Enter your email and password. 3. Choose Sign in.	The login should be successful
Tc-48	Check results on entering an invalid email, or password.	1. Launch the application on the login page. 2. Enter an email and password. 3. Choose Sign in.	Error message “invalid email or password.”
Tc-49	Check results when a user email or password are empty, and the “login” button is pressed.	1. Launch the application on the login page. 2. Enter an email or password and password. 3. Choose Sign in.	Error message “a field is missing”

Table 34 Test cases for Register Functionality

Test Case Scenario:		**Check Register Functionality.**	
Test case id	Test case title	Test steps	Expected result
Tc-50	Check results on completing the register form correctly, and select sign-up	1. Launch the application on the login page. 2. Complete the form of your information. 3. Choose Register	The Register should be successful
Tc-51	Check results on entering an existing email.	1. Launch the application on the login page. 2. Complete the form of your information. 3. Choose Register	Error message “the email is already existed”.
Tc-52	Check results when a user chose a weak password (less than 8 characters).	1. Launch the application on the login page. 2. Complete the form of your information. 3. Choose Register	Error message “ Must contain at least 8 characters”
Tc-53	Check results when Client try to register with Missing Fields	1. Launch the application on the login page. 2. Complete the form of your information. 3. Choose Register	Error message “a field is missing”
Tc-54	Check results when Client try to register and company not found	1. Launch the application on the login page. 2. Complete the form of your information. 3. Choose Register	Error message “Company not found “

Table 35 Test cases for workflow management

Test Case Scenario:		**Check Register Functionality.**	
Test case id	Test case title	Test steps	Expected result
Tc-55	Check if Admin can assign a Default Workflow to a company	1. Launch the website 2. Login as Admin 3. Go to Company Management 4. Click on Add Company 5. Select "Default Workflow" and save	Company created with Default Workflow assigned
Tc-56	Check if Admin can define a Customized Workflow	1. Launch the website 2. Login as Admin 3. Go to Company Management 4. Click on Add Company 5. Select "Customized Workflow" 6. Add steps: Step 1, Step 2, Step 3.. 7. Save the workflow	Company saved with customized workflow steps
Tc-57	Check if workflow steps appear on ticket board for companies with Customized Workflow	1. Login as STM 2. Choose a company using Customized Workflow 3. Open Ticket Management	Customized steps (e.g., Step 1 → Step 2 → Step 3) appear as columns
Tc-58	Move a ticket from Step 1 to Step 2 in Customized Workflow	1. Login as STM 2. Open Ticket Board 3. Drag ticket from Step 1 and drop in Step 2 4. Confirm status change	Ticket status updated to Step 2
Tc-59	Move a ticket through all stages of Default Workflow	1. Login as Support Team Member 2. Open Ticket Board 3. Drag ticket through: Open → Assign to Team → Assign to Member → In Progress → Done	Ticket reaches final status "Done"
Tc-60	Reject invalid workflow transition (e.g., skip a step)	1 .Login as STM 2 .Open Ticket Board with Customized Workflow 3. Attempt to drag ticket from Step 1 directly to Step 3 (skipping Step 2)	System rejects the move and shows validation error

7. Initial Requirement Trackability Matrix (RTM)

Table 36 Initial Requirement Trackability Matrix (RTM)

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-FR-UM-1.1	The system shall allow the admin to create a new user profile by entering necessary information	FR(Add_User)		Seq1.1				
RE-FR-UM-1.2	The system shall allow the Admin to edit user information	FR(Delete_User)		Seq1.2				
RE-FR-UM-1.3	The system shall allow the Admin to delete a user	FR(Edit_User)		Seq1.3				
RE-FR-UM-1.4	The system shall automatically assign a default role to a newly created user based on the actor performing the creation							
RE-FR-UM-1.4.1	When the Admin creates a new user, the system shall assign the role Support Team Manager by default							
RE-FR-UM-1.4.2	When a user registers through the system, the system shall assign the role Client by default							
RE-FR-UM-1.4.3	When a Support Team Manager creates a new user, the system shall assign the role Support Team Member by defaul							
RE-FR-TEM-2.1	The system shall allow Support Team Manager (STM) to add team to support teams	FR(Add_Team)		Seq2.1				
RE-FR-TEM-2.2	STM shall be able to delete team from the support teams	FR(Delete_Team)		Seq2.2				
RE-FR-TEM-2.3	The system shall allow STM to edit team details	FR(Edit_Team)		Seq2.3				
RE-FR-TEM-2.4	The system shall allow the STM to view a list of all support teams	FR(Show_Team)		Seq2.4				
RE-FR-TIM-3.1	The system shall allow Clients to create anew ticket, entering required details such as issue description	FR(Create_Ticket)		Seq3.1				
RE-FR-TIM-3.2	The system shall provide an option for the STMs to assign tickets to specific team for resolution.	FR(Assign_Ticket_ToTeam)		Seq3.2				

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-FR-TIM-3.3	The system shall employ an NLP-based approach to analyze the content of the ticket and automatically estimate its priority based on keywords, urgency, and sentiment analysis.							
RE-FR-TIM-3.4	The system shall provide functionality to view ticket details, including priority, category, company name, status of ticket, client name, and ticket description	FR(View_Ticket_Details)		Seq3.3				
RE-FR-TIM-3.5	The system shall allow authorized users to delete a ticket if required, with a record of the deletion captured in the ticket log	FR(Delete_Ticket)		Seq3.4				
RE-FR-TIM-3.6	The system shall allow authorized users to update a ticket, with a record of the deletion captured in the ticket log	FR(Update_Ticket)		Seq3.5				
RE-FR-TIM-3.7	The system shall allow Support Team Managers and Support Team Members to change the status of a ticket (e.g., "In Progress," "Done")	FR(Change_Ticket_Status)		Seq3.6				
RE-FR-TIM-3.8	The system shall allow Support Team members to view all tickets that have been assigned to their team	FR(Show_assigned_Tickets)		Seq3.7				
RE-FR-TIM-3.9	The system shall allow Support Team members to resolve tickets that have been assigned to them	FR(Solve_Ticket)		Seq3.8				
RE-FR-MM-4.1	The system shall allow the Support Team Manager (STM) to add new members to the system.	FR(Add_Member)		Seq4.1				
RE-FR-MM-4.2	The system shall enable the STM to add members to a specific support team	FR(Add_Member_ToTeam)		Seq4.2				
RE-FR-MM-4.3	The STM shall be able to change the status of a team member, including activating or deactivating the member	FR(Change_Status_Of_TeamMember)		Seq4.3				
RE-FR-MM-4.4	Support team members shall be able to assign a ticket to themselves.	FR(Assign_Ticket_to_Me)		Seq4.4				
RE-FR-AIM-5.1	The system shall allow all users to edit their account information.	FR(Update_Account)		Seq5.1				
RE-FR-AIM-5.2	Users shall be able to delete their accounts.	FR(Delete_Account)		Seq5.2				
RE-FR-CPM-6.1	The system shall allow the Admin to add a company	FR(Add_Company)		Seq6.1				

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-FR-CPM-6.2	The system shall allow the Admin to delete a company	FR(Delete_Company)		Seq6.2				
RE-FR-CPM-6.3	The system shall allow the Admin to edit company details	FR(Update_Company_Info)		Seq6.3				
RE-FR-CPM-6.4	The Admin shall be able to view a list of all companies	FR(Show_Company)		Seq6.4				
RE-FR-COM-7.1	The system shall allow STM, Support Team Member, and Client to add comments on ticket.	FR(Add_Comment)		Seq7.1				
RE-FR-COM-7.2	The system shall allow STM, Support Team Member, and Client to viewa list of comments associated with their tickets	FR(Show_Comment)		Seq7.2				
RE-FR-RM-8.1	The system shall allow new clients to register by providing necessary information	FR(Register)		Seq8.1				
RE-FR-SU-9.1	The system shall provide a login page for all users, requiring valid credentials (email and password) to access the system	FR(Login)		Seq9.1				
RE-FR-SU-9.2	The system shall validate user credentials against the database during login attempts and display appropriate error messages for incorrect username/email or password	FR(Login)		Seq9.2				
RE-FR-SU-9.3	The system shall redirect users to their respective dashboards or homepages based on their roles after a successful login	FR(Login)		Seq9.3				
RE-FR-SU-9.4	The system shall provide a logout option for users, terminating their session securely	FR(Login)		Seq9.4				
RE-FR-SU-9.5	The system shall automatically log out inactive users after a predefined period of inactivity for security purposes	FR(Login)		Seq9.5				
RE-FR-SU-9.6	The system shall implement security features for login, including password hashing using a secure algorithm	FR(Login)		Seq9.6				
RE-NF-SC-1.1	The system shall be scalable to support an increasing number of users and tickets without performance degradation.							
RE-NF-SC-1.2	The system architecture shall allow the addition of new servers or resources to handle increased workloads dynamically							

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detail Design	Coding	Test Cases	Changed Requests No
RE-NF-AV-2.2	The system shall maintain uptime to ensure continuous availability of ticket management and support features							
RE-NF-INT-3.1	The system shall be capable of integrating with third-party customer support platforms							
RE-NF-INT-3.2	The system shall ensure data compatibility with external systems							
RE-NF-SEC-4.1	The system shall enforce secure authentication, requiring users to log in using strong passwords							
RE-NF-SEC-4.2	The system shall encrypt all sensitive data (e.g., passwords, user information)							
RE-NF-EX-5.1	The system shall be extensible to support multiple languages, allowing for easy integration of new language modules in the future without significant architectural changes							
RE-NF-EX-5.2	The system shall be designed to allow for seamless integration of additional AI components							
RE-F-WF-8.1	The system shall allow the Admin to choose between a Default Workflow or a Customized Workflow							
RE-F-WF-8.2	Admin can define a sequence of steps in Customized Workflow							
RE-F-WF-8.3	STM and Members can transition tickets based on company's workflow							
RE-F-WF-8.4	Workflow visually displayed in Kanban-style interface							
RE-F-WF-8.5	Workflow configuration saved and applied per company							
RE-F-CAT-9.1	System uses BERT-based ML to analyze ticket content							
RE-F-CAT-9.2	System auto-assigns a category based on analysis							
RE-F-CAT-9.3	Category appears in ticket metadata							
RE-F-CAT-9.4	Categorization data saved for analytics and model improvement							

Chapter 5 System Design

1. Introduction

This chapter offers a comprehensive overview of the design phase steps , encompassing system design and architecture determination, as well as detailed design and modeling. It underscores the importance of these steps in creating a robust system for collecting, analyzing, and utilizing user feedback to continuously enhance the software system.

2. System Architecture

This section presents the overall system architecture and design of the AI-Powered Ticketing System. System architecture defines the structural organization of the software and illustrates how different components and services interact to ensure efficient data flow, scalability, and seamless integration. The proposed architecture adopts a microservices-based approach, allowing the system to separate responsibilities across independent services while maintaining clear communication between them.

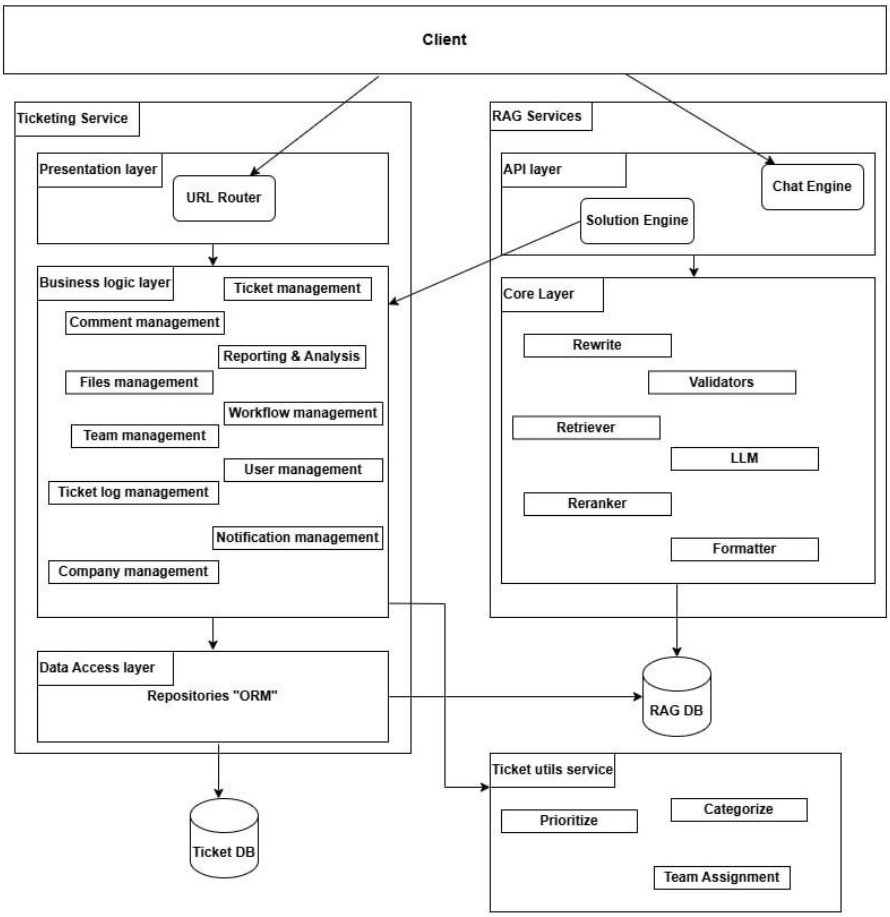


Figure 27 system architecture

Component functionalities:

1) **Client Interface:**

   Represents the user-facing layer where Clients, Support Team Members, Support Team Managers, and Admins interact with the system through a web-based interface to create tickets, manage workflows, access reports, and use AI-assisted features.

2) **Ticketing Service:**

   This core service is responsible for managing all ticket-related operations and business logic. It includes functionalities such as ticket creation, tracking, assignment, status updates, and resolution. The Ticketing Service also manages comments, notifications, workflow execution, reporting, and company configurations.

3) **Presentation Layer (URL Router):**

   Acts as the entry point for client requests within the Ticketing Service. It routes incoming requests to the appropriate business logic components and ensures proper request handling.

4) **Business Logic Layer:**

   Contains the main functional modules of the system, including:

   - Ticket Management
   - Team Management
   - User Management
   - Workflow Management
   - Comment Management
   - Reporting and Analysis
   - Notification Management
   - Company Management

   This layer coordinates system operations and enforces business rules.

5) **RAG Service:**

A dedicated microservice responsible for AI-assisted knowledge retrieval and response generation. It provides two main engines:

- **Chat Engine,** which supports conversational, multi-turn interactions.
- **Solution Engine**, which generates structured, actionable solution suggestions for resolving tickets.

The RAG Service includes a core processing layer consisting of rewriting, retrieval, validation, reranking, and language model components.

6) **Knowledge Base (RAG DB):**

Stores structured and unstructured documents used by the RAG Service. Uploaded files are indexed and retrieved to support context-aware AI responses.

7) **Ticket utils Services:**

Independent AI microservices responsible for specific automated tasks, including:

- Ticket Categorization
- Ticket Prioritization
- Auto Team Assignment

These services operate independently and produce separate outputs that support decision-making within the Ticketing Service.

8) **Data Access Layer (Repositories / ORM):**

Provides an abstraction layer for database interactions. It handles data persistence and retrieval for tickets, users, teams, workflows, logs, and company data.

9) **Databases:**

- **Ticket DB:** Stores core system data such as tickets, users, teams, workflows, logs, and reports.
- **RAG DB:** Stores knowledge base documents and retrieval-related data used by the RAG Service.

System Architecture

We used layered architecture with client-server architecture.

1. Client Side:

The client side represents the user interface layer, allowing users to interact with the system through dashboards, ticket views, reporting screens, and AI-assisted tools.

2. Server-Side Layers:

- **Ticketing Service:** Implements layered architecture (Presentation, Business Logic, and Data Access layers) to manage core system functionality.
- **RAG Service:** Operates as an independent AI microservice focused on retrieval-augmented response generation.
- **Mini AI Services:** Provide specialized AI capabilities for categorization, prioritization, and team assignment.

3. **Database Layer:**
   Persistent data is stored in dedicated databases and accessed through repository abstractions, ensuring data consistency and separation of concerns.

Design Patterns in Use

1. Repository Pattern:

- **Description**:
  The Repository Pattern is used within the Data Access Layer to abstract database operations. It provides a consistent and clean interface for accessing and manipulating data, while hiding the underlying database and ORM implementation details.

- **Purpose :**

  This pattern decouples the business logic from data persistence concerns, improving maintainability, testability, and flexibility. It also allows the system to adapt easily to changes in the database technology without affecting higher layers.

2. Dependency Injection Pattern

- **Description:**

  The system components, such as controllers, services, and AI modules, depend on each other to perform their tasks. Dependency Injection is used to supply these dependencies externally rather than creating them directly within components.

- **Purpose:**

  This pattern promotes loose coupling between system components, enhances testability, and supports modular development, which is particularly important in a microservices-based architecture.

3. Microservices Architectural Pattern

- **Description**:

  The system follows a microservices-based architectural pattern, where core functionalities are divided into independent services, such as the Ticketing Service, RAG Service, and Mini AI Services.

- **Purpose**:

  This pattern improves scalability, fault isolation, and maintainability by allowing each service to be developed, deployed, and scaled independently according to system demands.

3. Detailed design for system component

In this section we will dive into the detailed design for each component of the system, also mentions the design principles and patterns used to build a strong software system that fulfills its requirements.

Design class diagram

- **User Authentication**

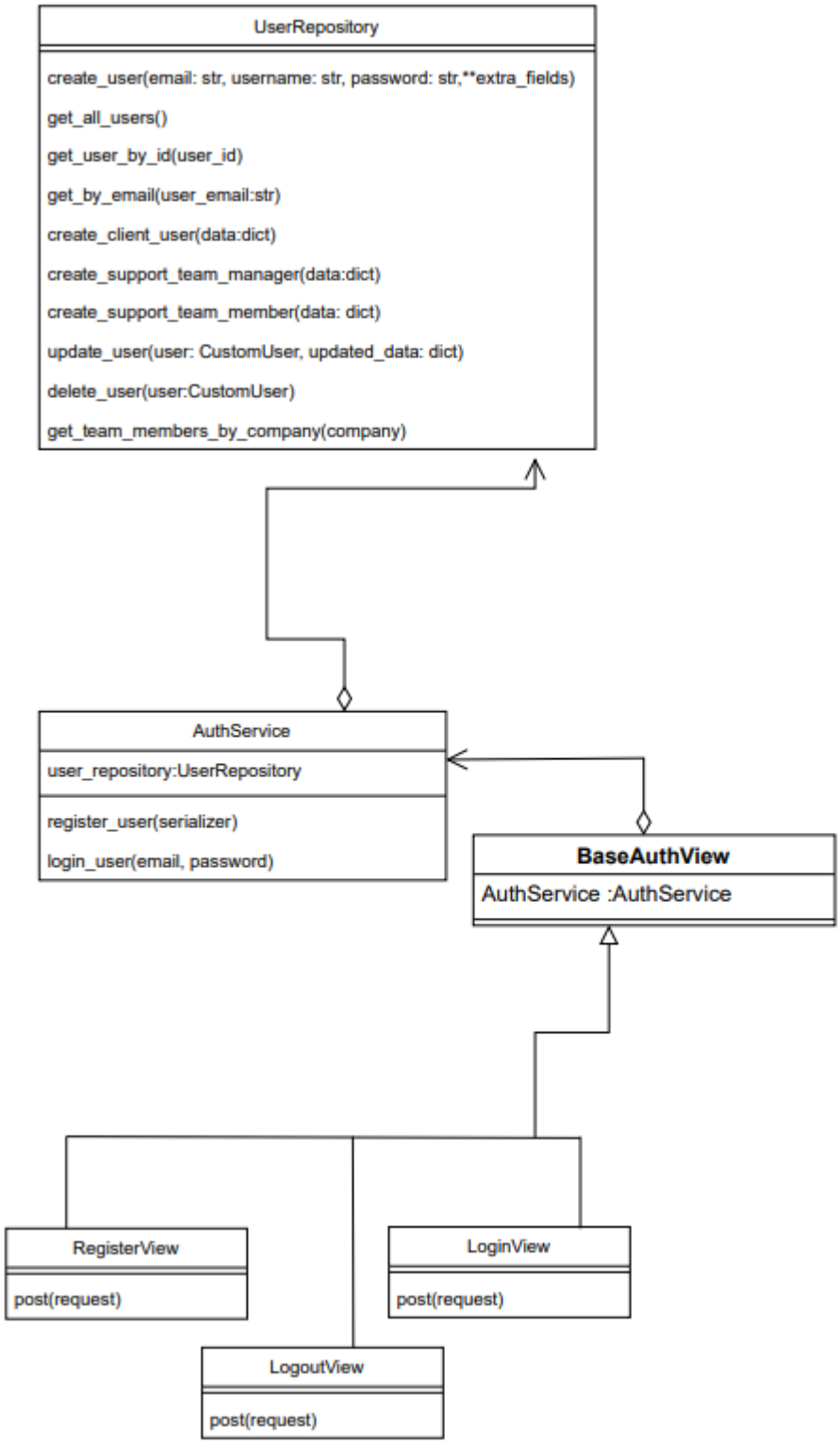
  This class diagram illustrates the relationships and responsibilities between the components in a user authentication and management system. It includes the following primary entities:

  1. **UserRepository:** Handles database operations for user management.
  2. **AuthService:** Provides authentication services using the repository.
  3. **BaseAuthView**: A base class for views requiring authentication services.
  4. **RegisterView, LoginView, and LogoutView:** Represent endpoints for user registration, login, and logout, respectively.

  #### Relationships

  1. **Dependency:**
     - AuthService depends on UserRepository for all user data operations.
     - **BaseAuthView** depends on AuthService for authentication services.
  2. **Inheritance:**
     - RegisterView, LoginView, and LogoutView inherit from **BaseAuthView**.

Figure 28 user authentication - class diagram



- **User management**

  This class diagram represents a detailed structure of a user management system with role-based access control (RBAC) and custom pagination. It demonstrates the relationships between components responsible for handling user-related operations, permissions, and pagination in a layered architecture.

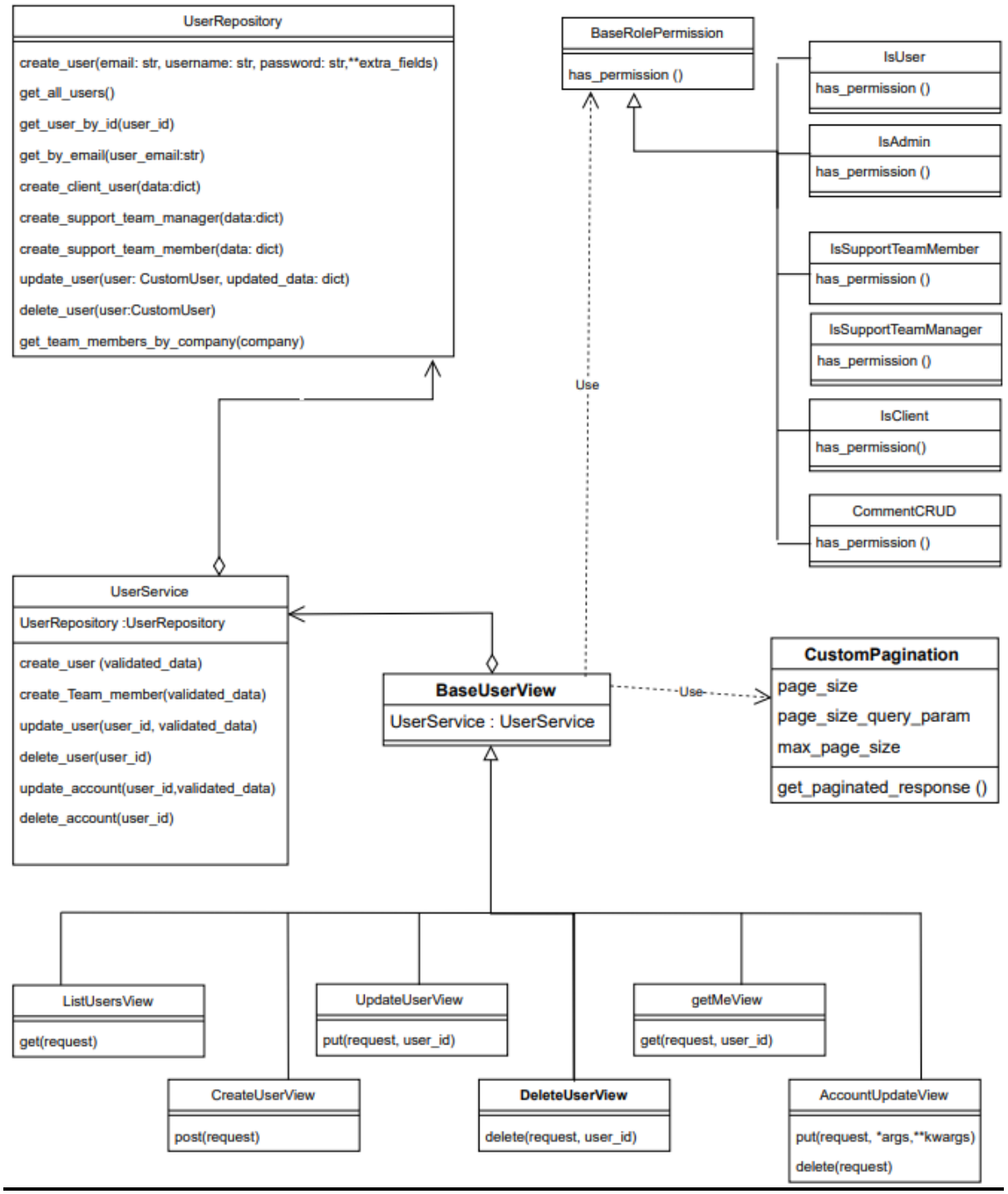
  ### Relationships

  1. **Dependency:**
     - **UserService** depends on **UserRepository** for database operations.
     - **BaseUserView** uses **UserService** to manage user-related logic.
     - Views (e.g., ListUsersView, CreateUserView) extend **BaseUserView**.
     - **BaseRolePermission** is used by role-specific permission classes.
     - Views may use CustomPagination for paginated responses.
  2. **Inheritance:**
     - All permission classes inherit from **BaseRolePermission**.
     - Views like ListUsersView, CreateUserView inherit from **BaseUserView**

  ### Key Features

  1. **Role-Based Access Control:**
     - Implements granular access control based on roles using permission classes.
  2. **Layered Architecture:**
     - Separation of concerns between data access (UserRepository), business logic (UserService), and presentation layer (views).
  3. **Pagination:**
     - Allows efficient handling of large datasets with customizable pagination.

Figure 29 user management - class diagram



- **Team management**

  This class diagram represents a detailed structure of a team management system with role-based access control (RBAC) and custom pagination. It illustrates the relationships between components responsible for managing teams, user assignments, permissions, and paginated responses in a layered architecture.

Relationships

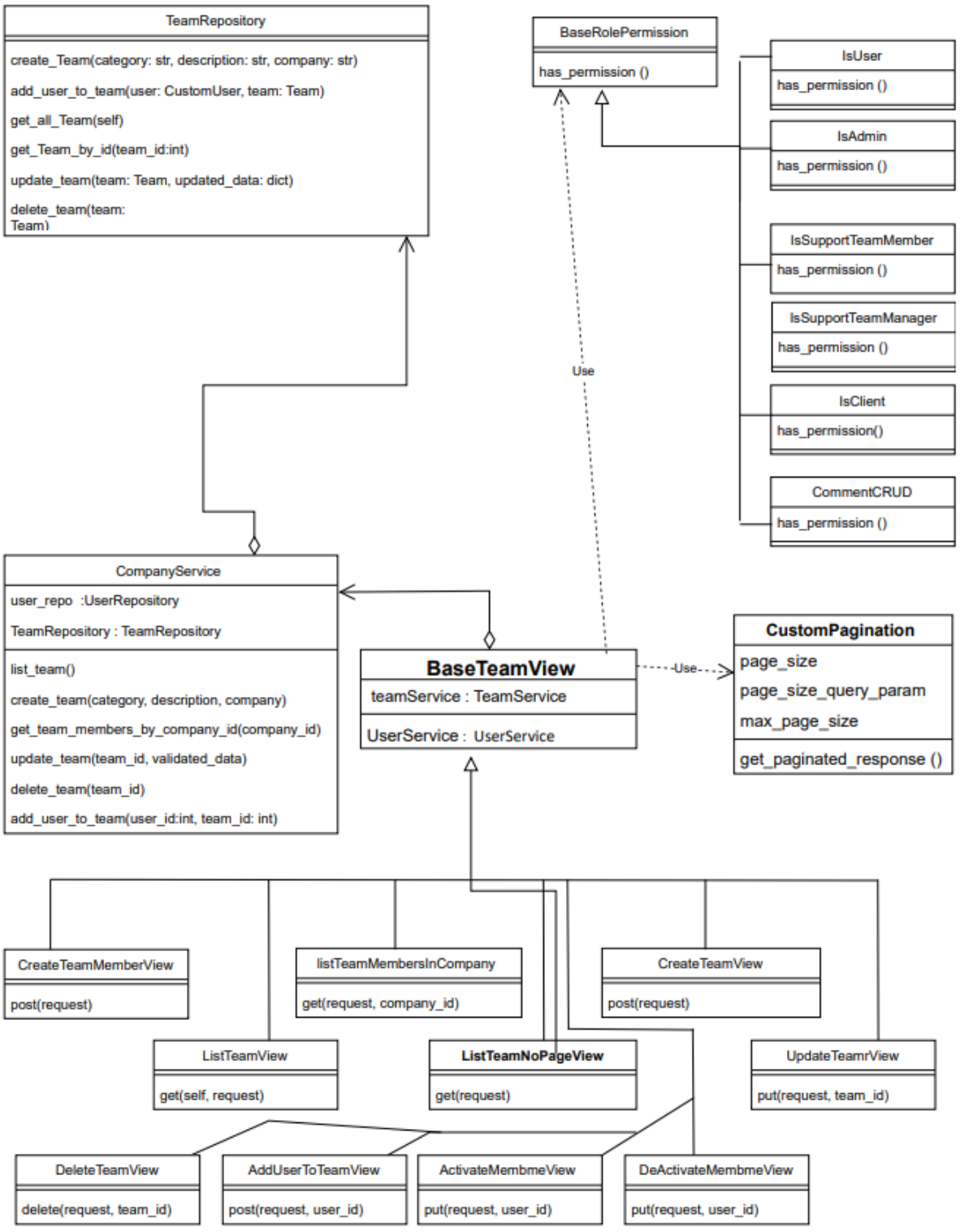
1. **Dependency**
   - **TeamService** depends on **TeamRepository** for team-related database operations.
   - **CompanyService** relies on **TeamRepository** and **UserRepository** to manage team and user interactions.
   - **BaseTeamView** uses **TeamService** and **UserService** to handle team and user-related logic.
   - Views like ListTeamView and CreateTeamView extend functionality from **BaseTeamView**.
   - **BaseRolePermission** is utilized by role-specific permission classes (e.g., IsAdmin, IsSupportTeamMember).
   - Views can use **CustomPagination** to manage large responses effectively.

2. **Inheritance**
   - All role-specific permission classes (e.g., IsAdmin, IsClient) inherit from **BaseRolePermission**.
   - Views like CreateTeamView, ListTeamView, and UpdateTeamView inherit from **BaseTeamView**.

Key Features

1. **Role-Based Access Control (RBAC)**
    - Implements granular access control using permission classes like IsAdmin and IsSupportTeamMember, ensuring only authorized users can access specific operations.
2. **Layered Architecture**
    - Ensures separation of concerns:
    - Repositories (e.g., TeamRepository) handle database interactions.
    - Services (e.g., CompanyService) manage business logic.
    - Views (e.g., DeleteTeamView, AddUserToTeamView) handle client-facing operations.
3. **Custom Pagination**
    - Enables efficient handling of large datasets using CustomPagination, with configurable page sizes and limits for scalability.

Figure30 team management - class diagram



- **Company management**

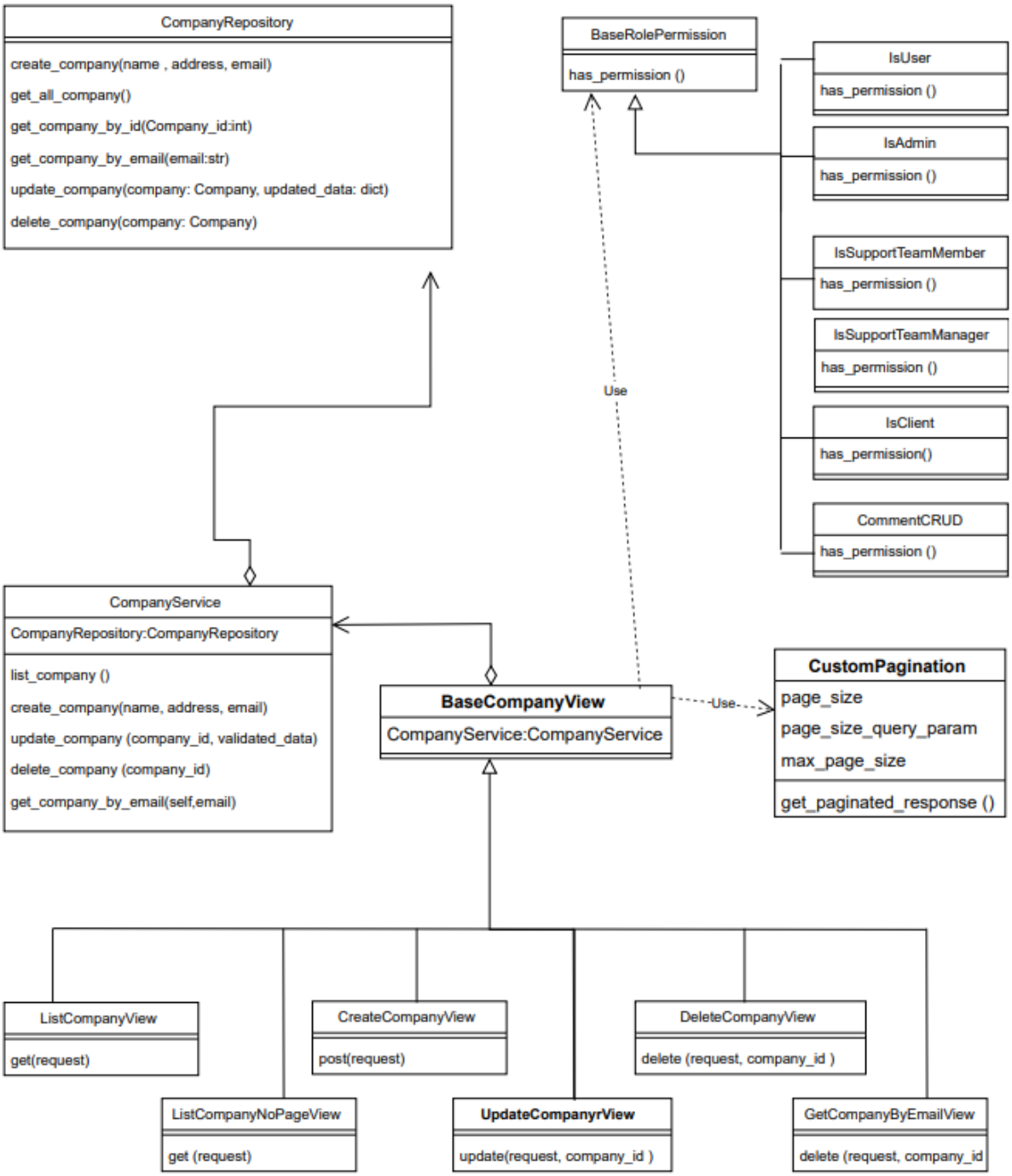
  This class diagram represents the architecture of the company management module within the system. It highlights the relationships between components responsible for handling company-related operations, role-based access control (RBAC), and custom pagination for efficient data handling.

Relationships

1. **Dependency**
   - **CompanyService** depends on **CompanyRepository** for interacting with the database.
   - **BaseCompanyView** relies on **CompanyService** for business logic related to companies.
   - API views like `ListCompanyView`, `CreateCompanyView`, and `UpdateCompanyView` extend the functionality of **BaseCompanyView**.
   - **BaseRolePermission** is utilized by role-specific permission classes (e.g., `IsAdmin`, `IsClient`) to enforce RBAC.
   - Views use **CustomPagination** to manage paginated responses when handling large datasets.

2. **Inheritance**
   - All role-specific permission classes (e.g., `IsUser`, `IsAdmin`) inherit from **BaseRolePermission** to implement access control.
   - Views like `CreateCompanyView` and `ListCompanyView` inherit from **BaseCompanyView** to reuse core functionality.

Figure31 company management - class diagram



- **Ticket management**

  This class diagram highlights the relationships between components responsible for ticket creation, assignment, updates, and role-based access control (RBAC). The system is designed with modularity and scalability, incorporating permissions, business logic, and custom pagination for managing large datasets.

Relationships

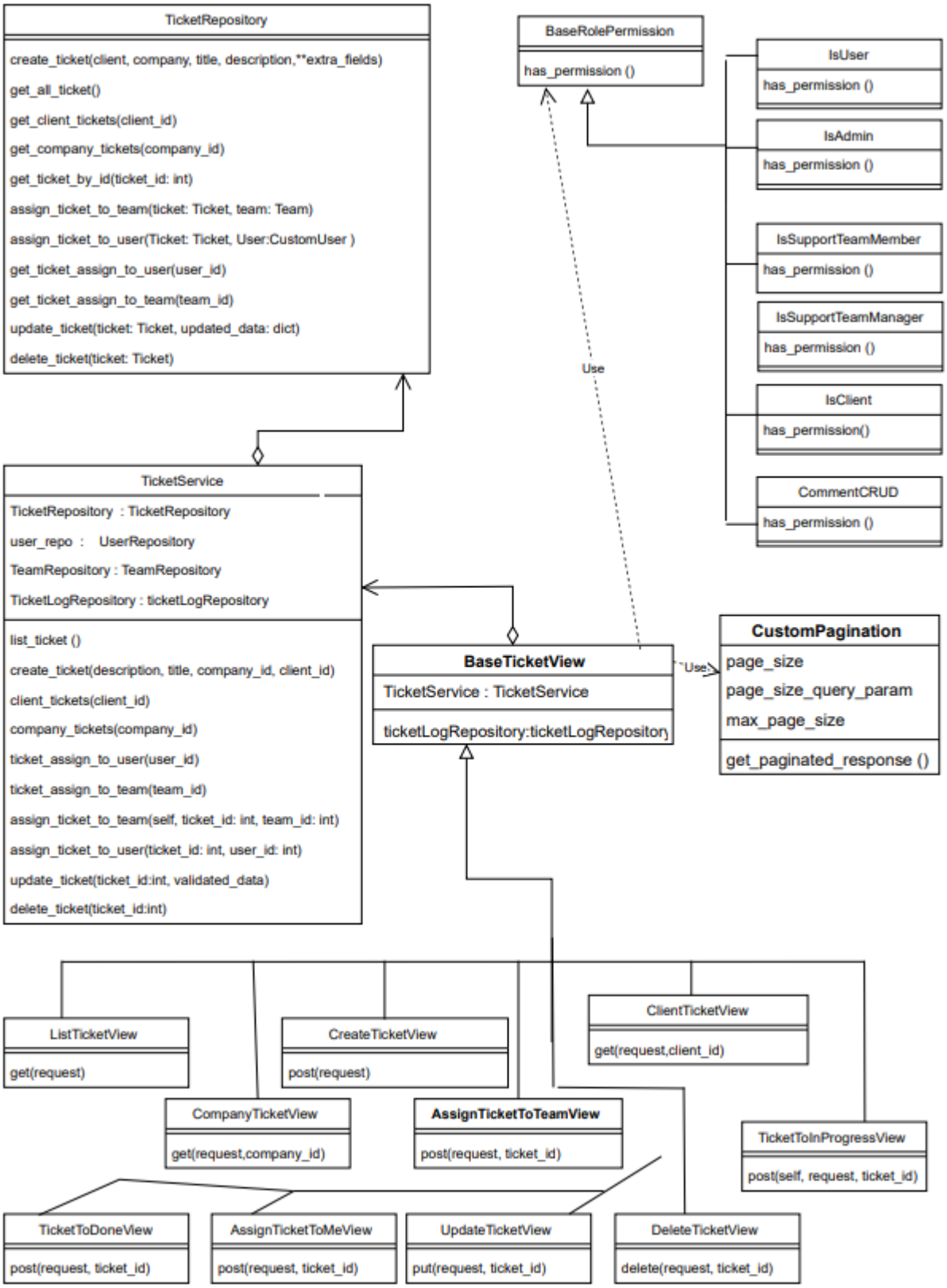
1. **Dependency**
   - **TicketService** depends on:
     - **TicketRepository** for ticket database operations.
     - **UserRepository** and TeamRepository for user and team management.
     - **TicketLogRepository** to maintain ticket logs.
   - **BaseTicketView** relies on **TicketService** for executing business logic.
   - Role-based permissions (e.g., `IsAdmin`, `IsSupportTeamMember`) extend **BaseRolePermission** for enforcing access control.
   - **CustomPagination** is used by views to enable efficient paginated data retrieval.

2. **Inheritance**
   - Views such as `ListTicketView` and `CreateTicketView` inherit from **BaseTicketView**, reusing its core logic.
   - Permission classes like `IsAdmin` and `IsClient` extend **BaseRolePermission** to enforce specific rules for different roles.

Key Features

1. **Role-Based Access Control (RBAC)**
    - Permission classes ensure that only authorized users can perform specific actions, providing fine-grained control.
2. **Layered Architecture**
    - Clear separation of responsibilities:
        - Repositories handle database-level operations.
        - Services encapsulate business logic.
        - Views expose API endpoints to interact with tickets.
3. **Custom Pagination**
    - Enables efficient handling of large ticket datasets with configurable page size and limits.
4. **Ticket Workflow Management**
    - Includes functionality for ticket lifecycle management, such as assigning tickets, updating status, and managing ownership.

Figure32 ticket management - class diagram



- **Ticket log management system**

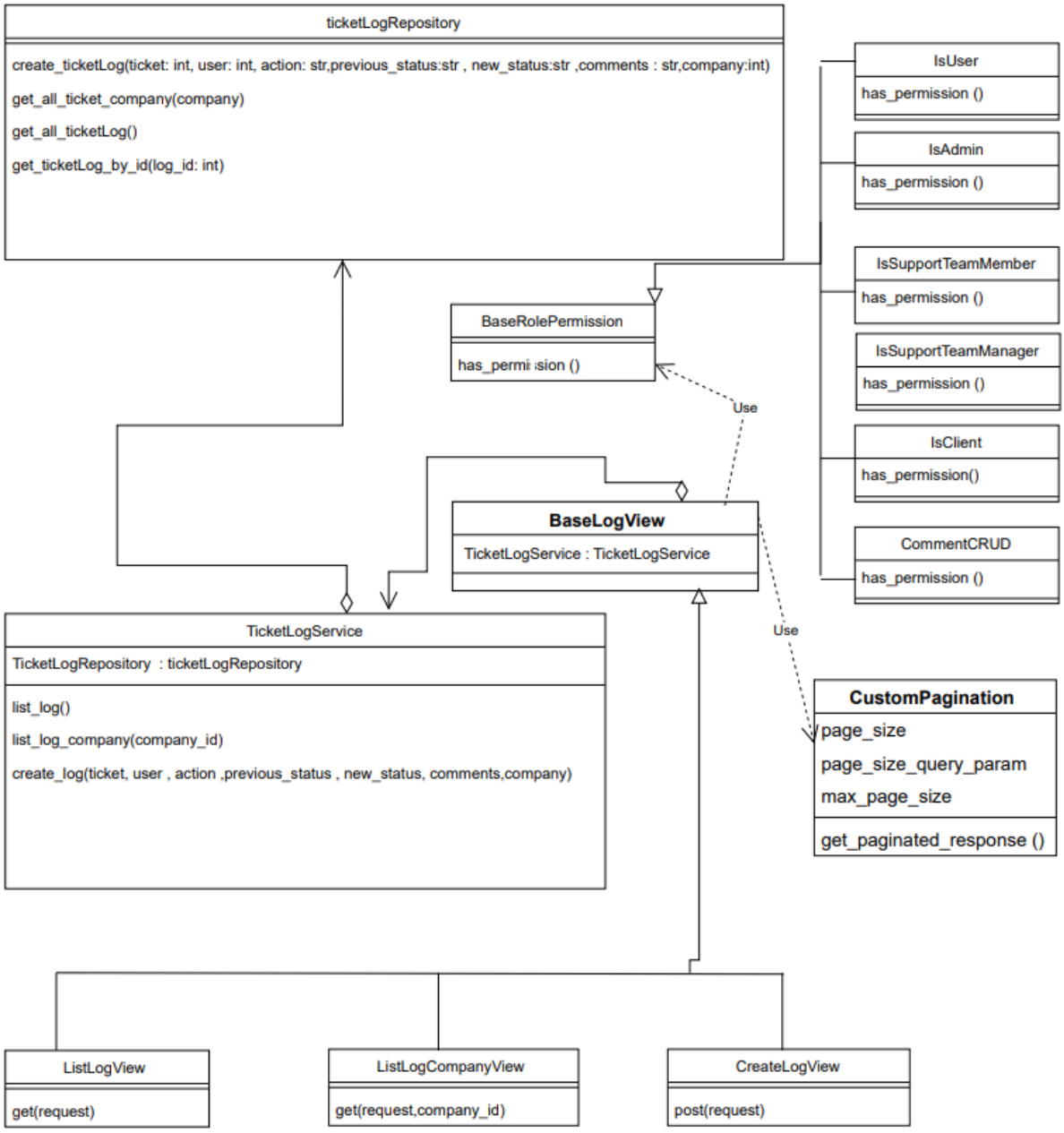
  This class diagram represents the architecture of a ticket log management system. It demonstrates the relationships between components responsible for managing ticket logs, which record the actions and status transitions of tickets in the system. The architecture incorporates modular components for role-based access control (RBAC), data access, business logic, and pagination.

Relationships

1. **Dependency**
   - **TicketLogService** depends on **ticketLogRepository** for database operations related to logs.
   - **BaseLogView** uses **TicketLogService** to process and serve log data to API endpoints.
   - **Role-based Permissions** (e.g., `IsAdmin`, `IsSupportTeamMember`) extend **BaseRolePermission** for enforcing access control rules.
   - **CustomPagination** is used to handle paginated log responses.

2. **Inheritance**
   - **BaseLogView** is the parent class for views like `ListLogView`, `ListLogCompanyView`, and `CreateLogView`, which inherit and extend its core logic.
   - Permission classes (e.g., `IsUser`, `IsAdmin`) inherit from **BaseRolePermission**, providing fine-grained control over access.

Figure33 Ticket log management system - class diagram



- **Comment management**

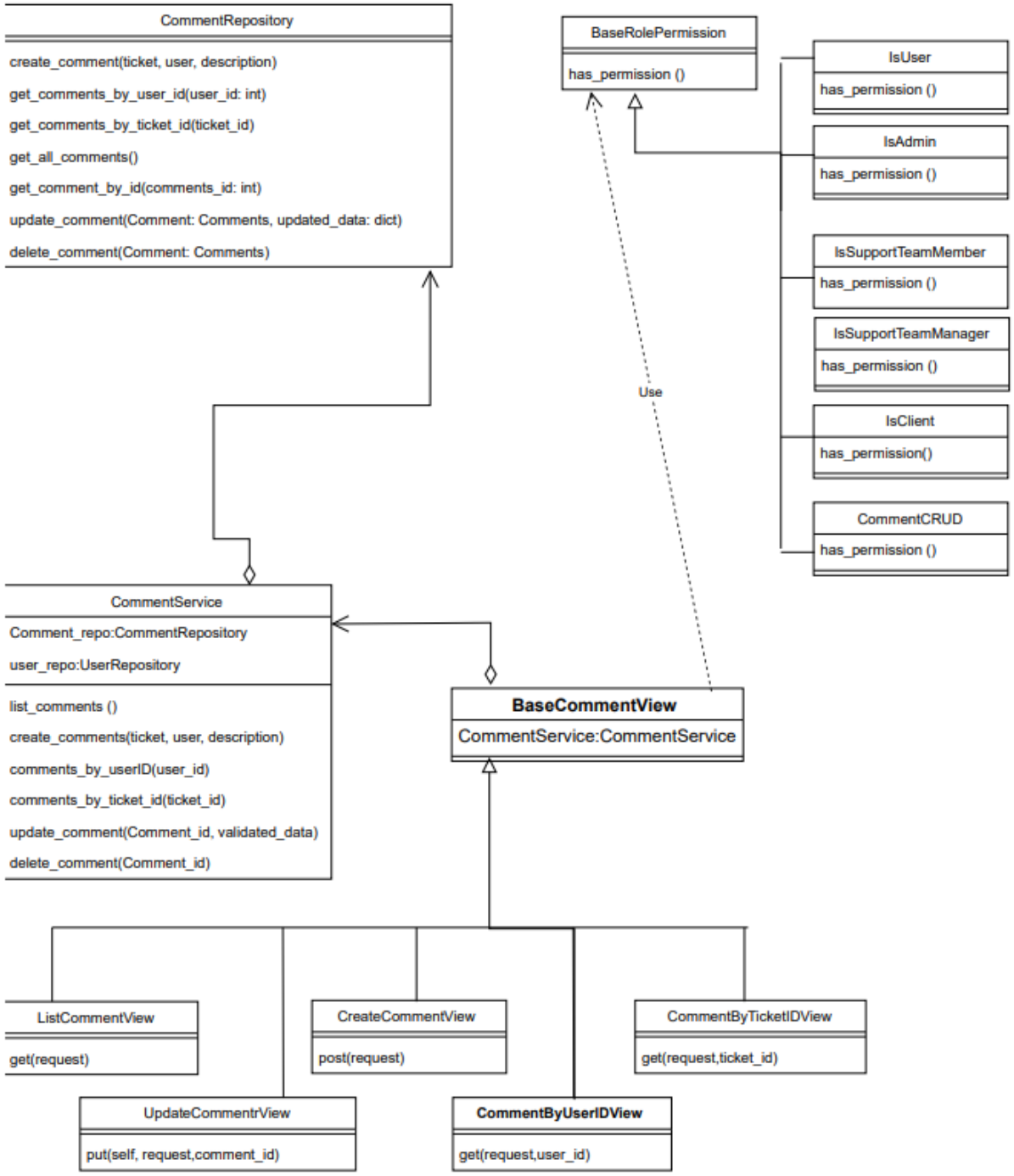
  This class diagram illustrates the architecture of a comment management system designed to handle user interactions through comments associated with tickets. The system follows a layered approach with components for role-based access control (RBAC), data access, business logic, and presentation, enabling scalable and secure comment operations.

Relationships

1. **Dependency**
   - **CommentService** relies on **CommentRepository** for database operations related to comments.
   - **BaseCommentView** depends on **CommentService** to process and deliver comment-related data to endpoints.
   - **Role-Based Permissions** extend **BaseRolePermission** to enforce access rules.
   - **Views** such as ListCommentView and CreateCommentView depend on **BaseCommentView** for shared logic.

2. **Inheritance**
   - **BaseCommentView** is the parent class for comment-related views like ListCommentView, and CreateCommentView.
   - Permission classes (e.g., IsAdmin, IsClient) inherit from **BaseRolePermission** for implementing specific access controls.

Figure34 Comment management - class diagram



Chapter 6 Artificial Intelligence Design and Techniques

1. Introduction

This chapter provides a comprehensive overview of the artificial intelligence design and techniques adopted in the AI-Powered Ticketing System (APTS). It presents the key AI components integrated into the system, including text-based analysis, automated decision support, and intelligent user assistance mechanisms.

The chapter focuses on explaining the design rationale, selected techniques, and integration approach of artificial intelligence within the ticketing system. It highlights how natural language processing and machine learning techniques are utilized to analyze ticket content, support priority estimation, categorize user requests, automatically assign tickets to appropriate support teams, and assist users through an intelligent chatbot.

Furthermore, this chapter emphasizes the role of artificial intelligence as a supportive layer that enhances system efficiency and decision-making while maintaining human control over critical operations. By clearly defining the responsibilities and limitations of each AI component, the chapter demonstrates how the proposed design contributes to building a scalable, modular, and reliable intelligent ticketing system.

2. AI Components Separation and Design Rationale

The artificial intelligence layer in the AI-Powered Ticketing System (APTS) is designed following a modular and component-based approach. Rather than relying on a single unified AI model, the system integrates multiple independent AI components, each responsible for a specific task within the ticket lifecycle. This design decision aims to improve clarity, maintainability, and reliability while ensuring that each AI function can be developed, evaluated, and enhanced independently.

The separation of AI components allows the system to address different operational needs without introducing unnecessary dependencies between outputs. Sentiment analysis is used exclusively to support priority estimation, focusing on identifying the emotional tone and urgency expressed in ticket content. Ticket categorization is implemented to classify tickets into predefined types for organizational and analytical purposes, without influencing routing or prioritization decisions. Automatic team assignment is handled by a dedicated machine learning model trained specifically for

routing tickets to the appropriate support teams. Additionally, the Retrieval-Augmented Generation (RAG) chatbot operates as an independent user assistance module, providing contextual support without directly affecting ticket processing decisions.

This clear separation of responsibilities prevents the propagation of errors across components and reduces the risk of cascading failures within the system. For example, an incorrect sentiment prediction does not impact team assignment, and category classification does not override priority decisions. Such isolation ensures that each AI component contributes to the system in a controlled and predictable manner.

From a design perspective, this approach aligns with best practices in intelligent system engineering, where AI modules are treated as assistive services rather than tightly coupled decision-makers. It also enhances the system's extensibility, allowing new AI features or improved models to be integrated in future iterations without requiring major architectural changes. By adopting this modular AI design, our APTS achieves a balanced combination of automation, transparency, and human oversight.

3. Sentiment Analysis for Priority Estimation

In the proposed system, sentiment analysis is implemented using the **spaCy** natural language processing library, extended with the **SpacyTextBlob** component to enable sentiment polarity detection. This approach relies on a pre-trained linguistic pipeline and does not involve any custom model training or dataset preparation, ensuring simplicity and fast inference during runtime.

The sentiment analysis process begins by loading a lightweight English language model provided by **spaCy**. This model is responsible for core NLP tasks such as tokenization and text normalization. To support sentiment detection, an additional sentiment component is integrated into the **spaCy** processing pipeline. This component enriches the processed document with sentiment-related attributes without modifying the underlying language model.

When a ticket description is submitted, the text is passed through the **spaCy** pipeline, where it is first tokenized into a structured document representation. The sentiment component then analyzes the processed text and produces a sentiment polarity score. This polarity value is a continuous numerical measure that reflects the emotional tone of the ticket content, ranging from negative to positive sentiment.

Based on the extracted polarity score, the system applies a threshold-based decision logic to estimate the ticket priority. Strongly negative sentiment values indicate potential urgency or user dissatisfaction and are therefore mapped to a high-priority level. Neutral sentiment values correspond to medium priority, while positive sentiment values are mapped to low priority. This mapping strategy allows the system to translate qualitative emotional signals into actionable priority indicators.

The sentiment-based priority estimation mechanism operates independently from other AI components in the system. The resulting priority level is used solely as a supporting signal during ticket handling and does not influence ticket categorization or automatic team assignment. This design ensures modularity, prevents unintended dependencies between AI outputs, and preserves human oversight in critical decision-making processes.

3.1 Technical Workflow of Sentiment Analysis

The sentiment analysis process in the proposed system follows a clearly defined execution workflow. Each step is executed sequentially to ensure accurate sentiment extraction and reliable priority estimation.

Step 1: Language Model Initialization

The process begins by loading a pre-trained English language model provided by spaCy. This model initializes the core NLP pipeline required for processing textual input. The loaded pipeline is responsible for handling fundamental tasks such as text segmentation and normalization.

Step 2: Pipeline Extension with Sentiment Component

After initializing the base language model, the sentiment analysis capability is added to the pipeline by integrating the **SpacyTextBlob** component. This component is appended at the end of the pipeline to ensure that sentiment analysis is performed after the text has been fully processed by the core NLP components.

This step transforms the pipeline from a purely linguistic processor into a sentiment-aware processing chain.

3.2 Text Processing and Document Representation

Step 3: Ticket Text Processing

When a ticket description is submitted, the text is passed to the spaCy pipeline. The pipeline converts the raw text into a structured document object. This document object serves as the central data structure through which all subsequent NLP operations are performed.

Step 4: Tokenization

The pipeline tokenizes the input text into individual tokens. Each token represents a meaningful unit of text and is enriched with linguistic metadata. Tokenization enables the sentiment component to analyze the text in a structured and consistent manner rather than as raw text.

3.3 Sentiment Polarity Extraction

Step 5: Sentiment Analysis Execution

Once tokenization and preprocessing are completed, the sentiment component analyzes the processed document. The sentiment analyzer computes a polarity score that reflects the overall emotional orientation of the ticket description.

The polarity score is a continuous numerical value, where negative values indicate negative sentiment, values close to zero indicate neutral sentiment, and positive values indicate positive sentiment.

Step 6: Polarity Validation

Before using the sentiment output, the system verifies that the sentiment component has successfully enriched the document with sentiment metadata. This validation step ensures robustness and prevents unexpected runtime behavior.

3.4 Threshold-Based Priority Determination

Step 7: Priority Mapping Logic

The extracted polarity score is interpreted using a threshold-based decision mechanism. The system applies predefined thresholds to map sentiment polarity into discrete priority levels.

- Polarity values below a negative threshold are mapped to High priority
- Polarity values within a neutral range are mapped to Medium priority
- Polarity values above a positive threshold are mapped to Low priority

This rule-based approach provides a transparent and explainable method for converting sentiment signals into operational priorities.

3.5 Output Handling and System Integration

Step 8: Priority Output Generation

The resulting priority level is returned as the final output of the sentiment analysis component. This output is stored and used as a supporting signal during ticket handling.

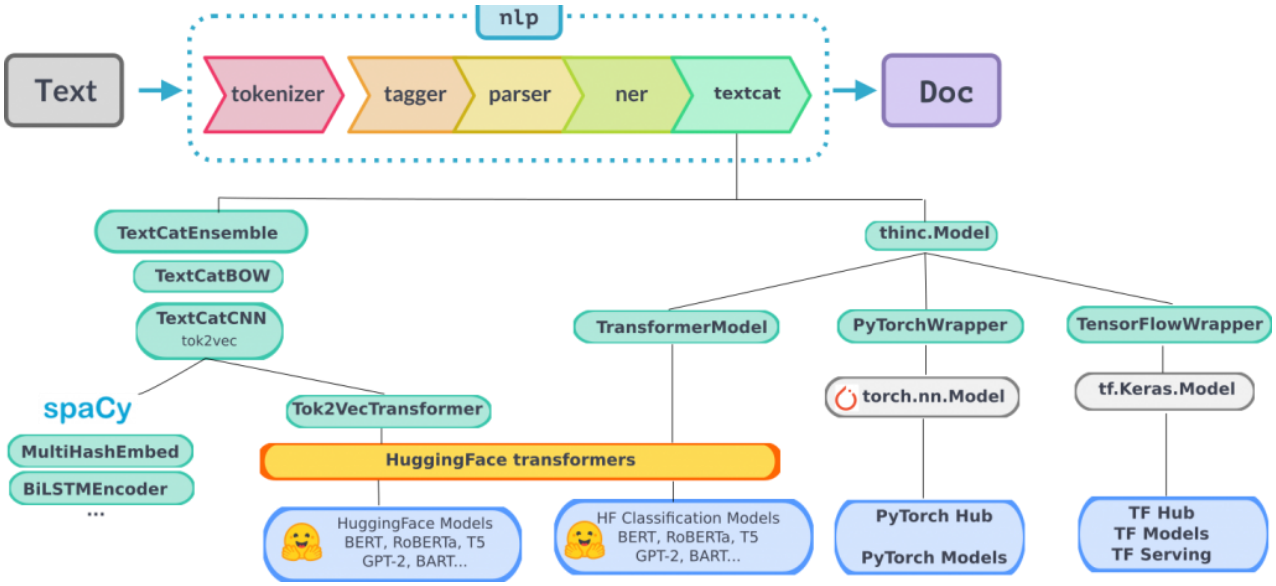
Step 9: Component Isolation

The priority output generated by sentiment analysis is not shared with other AI components. It does not influence ticket categorization or automatic team assignment. This strict isolation ensures that sentiment analysis remains a supportive module rather than a controlling one.

3.6 Design Advantages of the Step-Based Approach

The step-based implementation of sentiment analysis offers several advantages. It enables clear traceability from input text to final priority output, simplifies debugging and validation, and enhances explainability for human operators. Additionally, this design supports system extensibility by allowing individual steps to be modified or replaced without affecting the overall AI architecture.

Figure 35 General spaCy NLP Pipeline Architecture



This Figure illustrates the general pipeline architecture adopted by spaCy for text processing. In the proposed system, only a subset of these components is utilized, focusing primarily on tokenization and sentiment analysis through an integrated sentiment component.

4. Ticket Category Classification Using BERT

Ticket category classification is implemented in the proposed system to identify the general nature of user requests and organize tickets into predefined categories. Unlike the automatic team assignment component, this classification task does not involve model training or dataset preparation. Instead, it relies on the contextual language understanding capabilities of a pre-trained BERT model to perform inference-only classification.

The primary objective of this component is to enhance ticket organization and reporting by assigning each ticket to one of the predefined categories: **Request**, **Feedback**, or **Complaint**. The resulting category label is used for informational and analytical purposes and does not influence priority estimation or team assignment.

4.1 Category Definition and Scope

The system defines a fixed set of ticket categories prior to runtime. These categories represent high-level ticket intent types commonly observed in service desk systems.

- Request: Tickets requesting a service, action, or information
- Feedback: Tickets providing opinions or suggestions
- Complaint: Tickets expressing dissatisfaction or reporting issues

By limiting the classification to a small and clearly defined label set, the system ensures consistent categorization and avoids ambiguity during inference.

4.2 Technical Workflow of BERT-Based Category Classification

The category classification process follows a structured execution workflow that transforms raw ticket text into a semantic category label.

Step 1: Ticket Text Input

The textual content of the ticket, including the title and description, is collected as the input for classification. This text represents unstructured natural language data that requires semantic interpretation.

Step 2: Text Tokenization Using BERT Tokenizer

Before classification, the input text is processed using the tokenizer associated with the selected BERT model. The tokenizer splits the text into subword tokens and maps them into numerical representations required by the model.

This step preserves contextual meaning and allows the model to handle variations in vocabulary and sentence structure.

Step 3: Contextual Encoding

The tokenized text is passed through the pre-trained BERT model. During this step, BERT generates contextual embeddings that capture semantic relationships between words based on their surrounding context.

Since the model is pre-trained on large-scale corpora, it can effectively understand the intent of the ticket without requiring additional training for this specific task.

Step 4: Category Inference

The contextual representation produced by BERT is used to infer the most suitable category among the predefined labels. The classification decision is generated during inference time only, without updating model parameters or performing backpropagation.

This inference-only approach enables efficient execution while maintaining acceptable classification accuracy for high-level categorization tasks.

4.3 Output Handling and Usage

The output of the category classification component is a single category label corresponding to the ticket's general intent. This label is stored as part of the ticket metadata and used for:

- Ticket organization and filtering
- Statistical analysis and reporting
- High-level insight into user interaction patterns

The category output does not affect priority estimation or automatic team assignment. Each AI component operates independently to prevent unintended coupling between system decisions.

4.4 Design Rationale for Using BERT Without Training

The decision to use a pre-trained BERT model without additional training is driven by several design considerations:

- The classification task targets broad intent categories rather than fine-grained domain-specific labels
- Pre-trained BERT models provide strong contextual understanding without requiring labeled datasets
- Avoiding training reduces system complexity and computational cost
- Inference-only usage ensures fast response times during ticket creation

This design choice balances performance and practicality while remaining suitable for the scope of the proposed system.

4.5 Component Independence and System Integrity

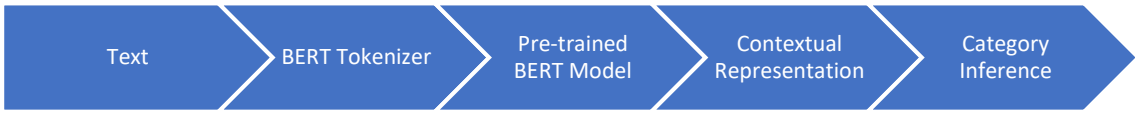


Figure 36 BERT-Based Inference Workflow for Ticket Category Classification

The category classification component is intentionally isolated from other AI modules. Its output is not used to override or influence sentiment-based priority estimation or machine learning–based team assignment. This separation ensures that errors in category prediction do not propagate to operational decisions, thereby improving system reliability and transparency.

5. AI-Based Automatic Team Assignment

Automatic team assignment is implemented to route incoming tickets to the most appropriate support team based on their textual content. Unlike sentiment analysis and category classification components, this module relies on supervised machine learning and involves explicit model training and evaluation. The objective of this component is to reduce manual routing effort and improve response efficiency by leveraging historical ticket data.

5.1 Dataset Description

The automatic team assignment component is trained using the IT Service Ticket Classification Dataset, which is publicly available on Kaggle. The dataset consists of real-world IT service desk tickets and is designed for supervised text classification tasks related to support team routing.

- The dataset contains 47,837 ticket records, where each record represents an individual IT service support ticket. Each ticket includes a textual description of the reported issue or request, along with a corresponding label that identifies the support team responsible for handling the ticket. These labels serve as the target classes for the supervised team assignment task.

- The dataset covers eight distinct support teams, namely: Hardware, HR Support, Access, Miscellaneous, Storage, Purchase, Internal Project, and Administrative Rights. Each support team represents a specific IT service domain, such as infrastructure-related issues, access management, internal project coordination, or administrative support. The presence of multiple well-defined classes makes the dataset suitable for evaluating multi-class text classification models in a realistic service desk environment.

Due to its size, diversity of ticket content, and well-defined team labels, this dataset provides a reliable foundation for training and evaluating machine learning models for automatic team assignment.

5.2 Data Preparation and Splitting

Before training the machine learning models for automatic team assignment, the dataset undergoes a data preparation phase to ensure that the textual input and corresponding labels are suitable for supervised learning. This phase focuses on cleaning the data, preparing labels, and dividing the dataset into training and testing subsets.

Step 1: Text Cleaning and Normalization

The textual content of each ticket is first processed to remove unnecessary noise that may negatively affect model performance. This step includes basic text cleaning operations such as removing redundant whitespace and non-informative characters, while preserving the core semantic content of the ticket descriptions. The goal of this step is to provide consistent and clean textual input for feature extraction and model training.

Step 2: Label Preparation

Each ticket in the dataset is associated with a predefined support team label. These labels represent the target classes for the team assignment task. During this step, the categorical team labels are prepared in a format suitable for machine learning models, enabling the models to learn the mapping between ticket text and the corresponding support team.

Step 3: Dataset Splitting

After preprocessing, the dataset is divided into two mutually exclusive subsets to support model training and evaluation:

- Training Set (70%): Used to train the machine learning models and allow them to learn patterns from the ticket data.
- Testing Set (30%): Used to evaluate the performance of the trained models on unseen data.

This split ensures that model evaluation reflects the generalization capability of the models rather than their ability to memorize the training data.

Step 4: Preparation for Multiple Models

The prepared training and testing datasets are reused consistently across different models to ensure a fair comparison. Both the baseline SVM model and the BERT-based model are trained and evaluated using the same data split, allowing performance differences to be attributed to the modeling approach rather than data variation.

5.3 Baseline Model: Support Vector Machine (SVM)

5.3.1 Model Selection Rationale

The selection of the Support Vector Machine (SVM) model in this work is not arbitrary, but rather motivated by findings reported in the reference study associated with the selected dataset. In that study, multiple machine learning and deep learning models were evaluated for the IT service ticket classification task, and SVM was identified as one of the highest-performing traditional models. Based on these empirical results, SVM was chosen in the proposed system as a baseline model to provide a reliable and empirically justified point of comparison with more advanced deep learning approaches.

Support Vector Machine is widely used in text mining and document classification tasks due to its effectiveness in handling high-dimensional feature spaces. Its proven performance in prior studies makes it a suitable baseline for evaluating the effectiveness of contextual transformer-based models in automatic team assignment.

5.3.2 Training and Evaluation Setup

In the proposed system, ticket text is transformed into numerical feature representations using a traditional text vectorization approach, enabling the SVM model to process unstructured textual data in a supervised learning setting. The dataset is divided into 70% for training and 30% for testing to ensure a fair assessment of the model's generalization capability.

The SVM model is trained using the training subset, where it learns decision boundaries that separate ticket descriptions according to their corresponding support teams. After training, the model is evaluated on the testing subset to measure its classification performance on previously unseen data. The evaluation is conducted using standard multi-class classification metrics, including accuracy, precision, recall, and F1-score.

5.3.3 Experimental Results

After training the SVM model on 70% of the dataset and evaluating it on the remaining 30% testing subset, which consists of 14,352 ticket records, the model demonstrates strong baseline performance. The experimental results indicate that the SVM model achieves an overall classification accuracy of 86.02%.
In addition, the macro-averaged F1-score reaches 86.08%, while the weighted F1-score is 86.02%, reflecting consistent performance across both balanced and imbalanced class distributions. These results confirm that SVM is capable of effectively capturing relevant patterns in ticket descriptions using traditional text representations.

5.3.4 Per-Class Performance Analysis

A detailed analysis of per-class performance reveals that the SVM model performs particularly well for several support teams. Categories such as Purchase and Storage achieve high F1-score values, indicating that tickets related to these domains are classified with high reliability. Other classes, including Access, HR Support, and Internal Project, also show stable precision and recall values, demonstrating the model's ability to distinguish between multiple IT service domains.

Some variation in performance is observed across classes, which can be attributed to differences in class distribution and textual complexity. For instance, categories with fewer samples, such as Administrative Rights, exhibit lower recall compared to more frequent classes. This behavior is expected in real-world multi-class classification tasks and highlights the challenges associated with class imbalance.

5.3.5 SVM Performance Summary

Table 37: SVM Performance summary

Metric	Value
Accuracy	**86.02%**
F1-score (Macro Average)	**86.08%**
F1-score (Weighted Average)	**86.02%**
Test Set Size	**14,352 tickets**

Table 38: SVM Classification Performance per Class

Support Team	Precision	Recall	F1-score	Support
Access	0.896	0.877	0.887	2138
Administrative Rights	0.850	0.710	0.774	528
HR Support	0.864	0.875	0.870	3275
Hardware	0.822	0.871	0.846	4085
Internal Project	0.892	0.835	0.863	636
Miscellaneous	0.834	0.832	0.833	2118
Purchase	0.962	0.892	0.926	739
Storage	0.919	0.861	0.889	833

5.3.6 Discussion

The obtained results confirm that SVM provides a strong and reliable baseline for the automatic team assignment task. Its ability to achieve high classification accuracy using traditional text representations validates its selection based on the reference study associated with the dataset. However,

despite its effectiveness, the SVM model relies on surface-level text features and lacks deep contextual understanding of language.
These limitations motivate the adoption of transformer-based models, such as BERT, which are capable of capturing richer semantic relationships within ticket descriptions. The following section presents the BERT-based team assignment model and compares its performance against the SVM baseline.

5.4 BERT-Based Team Assignment Model

5.4.1 Model Selection Rationale

The selection of the BERT-based model for automatic team assignment is motivated by findings reported in the reference study associated with the selected dataset, where transformer-based models achieved superior performance compared to traditional machine learning approaches. In particular, BERT demonstrated strong results due to its ability to capture contextual relationships within text, making it well suited for complex natural language classification tasks such as IT service ticket routing.

Based on these findings, BERT is selected in the proposed system as the primary deep learning model for automatic team assignment, with the objective of improving classification accuracy beyond the traditional SVM baseline.

5.4.2 Training and Evaluation Setup

In this component, ticket descriptions are processed using the tokenizer associated with the pre-trained BERT model. The tokenizer converts raw text into subword tokens and numerical representations required by the transformer architecture. Unlike traditional vectorization techniques, this approach preserves contextual meaning within ticket descriptions.

The dataset is split into 70% for training and 30% for testing, consistent with the experimental setup used for the SVM model. This consistency ensures a fair comparison between models. The BERT model is fine-tuned on the training subset,

allowing it to adapt its parameters to the specific task of support team classification.

After training, the model is evaluated on the testing subset using the same multi-class classification metrics applied to the baseline model.

5.4.3 Experimental Results

Following training and evaluation, the BERT-based model demonstrates improved performance over the traditional baseline. The experimental results indicate that the model achieves an overall classification accuracy of 87.35% on the testing subset, which consists of 14,352 ticket records.

The macro-averaged F1-score reaches 87.12%, while the weighted F1-score is 87.36%, reflecting robust performance across both frequent and less frequent support team categories.

5.4.4 Per-Class Performance Analysis

An analysis of per-class performance shows that the BERT model consistently improves classification results across most support teams. Categories such as Purchase, Storage, and Access achieve particularly high F1-score values, indicating that contextual language representations enable more accurate differentiation between ticket types.

Compared to the SVM baseline, BERT demonstrates improved recall for classes with higher textual variability, such as Hardware and HR Support. While some variation in performance remains for less frequent classes, such as Administrative Rights, the overall classification behavior is more balanced due to BERT's ability to capture semantic context.

5.4.5 BERT Performance Summary

Table 39: BERT Performance Summary

Metric	Value
Accuracy	**87.35%**
F1-score (Macro Average)	**87.12%**
F1-score (Weighted Average)	**87.36%**
Test Set Size	**14,352 tickets**

Table 40 :BERT Classification Performance per Class

Support Team	Precision	Recall	F1-score	Support
Access	0.935	0.879	0.906	2138
Administrative Rights	0.835	0.756	0.793	528
HR Support	0.873	0.893	0.883	3275
Hardware	0.828	0.903	0.864	4085
Internal Project	0.903	0.836	0.869	636
Miscellaneous	0.866	0.821	0.843	2118
Purchase	0.940	0.896	0.918	739
Storage	0.937	0.856	0.895	833

5.4.6 Discussion

The obtained results confirm that the BERT-based model outperforms the traditional SVM baseline for the automatic team assignment task. The observed improvement can be attributed to BERT's contextual language understanding, which enables more accurate interpretation of complex ticket descriptions.

Despite the increased computational cost associated with transformer-based models, the performance gains justify the selection of BERT as the final model for deployment in the proposed system. The following section presents a direct

comparison between the SVM and BERT models to further highlight the performance differences and support the final model selection.

5.5 Model Comparison: SVM vs BERT

To determine the most suitable model for automatic team assignment, a comparative evaluation is conducted between the traditional Support Vector Machine (SVM) baseline model and the transformer-based BERT model. Both models are trained and evaluated using the same dataset split (70% training and 30% testing) and assessed using identical multi-class classification metrics to ensure a fair and consistent comparison.

5.5.1 Quantitative Performance Comparison

The Table presents a summary comparison of the overall performance metrics achieved by the SVM and BERT models on the test set.

Table 41: Performance Comparison Between SVM and BERT

Metric	SVM	BERT
Accuracy	86.02%	**87.35%**
F1-score (Macro Average)	86.08%	**87.12%**
F1-score (Weighted Average)	86.02%	**87.36%**
Test Set Size	14,352 tickets	14,352 tickets

The results indicate that the BERT model consistently outperforms the SVM baseline across all evaluation metrics. Although the numerical improvement may appear moderate, it is significant in the context of a multi-class classification task involving real-world service desk data.

5.5.2 Qualitative and Per-Class Comparison

Beyond overall performance metrics, a per-class analysis reveals that BERT provides more balanced classification behavior across different support teams. The BERT model demonstrates higher recall and F1-scores for several categories, particularly those with more diverse and context-dependent ticket descriptions, such as Hardware, HR Support, and Miscellaneous.

While the SVM model performs strongly for well-defined and frequent categories, its reliance on surface-level text features limits its ability to fully capture contextual relationships. In contrast, BERT's contextual embeddings enable improved semantic understanding, resulting in better handling of ambiguous or complex ticket descriptions.

5.5.3 Discussion and Model Selection

The comparative analysis confirms that SVM serves as a strong and reliable baseline model, validating its selection based on the reference study associated with the dataset. However, the BERT model demonstrates superior performance due to its ability to capture deeper semantic and contextual information within ticket text.

Based on both quantitative metrics and qualitative behavior across classes, the BERT model is selected as the final model for automatic team assignment in the proposed system. Despite its higher computational cost, the performance gains justify its deployment, particularly in scenarios where accurate ticket routing can significantly improve support efficiency and response times.

6. Retrieval-Augmented Generation (RAG) Chatbot Design and Integration

6.1 Purpose and Role of the RAG Chatbot

The Retrieval-Augmented Generation (RAG) chatbot is integrated into the system to provide intelligent, context-aware responses to user inquiries related to tickets and system procedures. Unlike traditional chatbots that rely solely on a language model's internal knowledge, the proposed RAG chatbot retrieves relevant information from a dedicated knowledge base before generating a response.

The primary objectives of the RAG chatbot are:

- To assist users by providing accurate and policy-compliant answers
- To reduce repetitive or unnecessary ticket creation
- To support ticket resolution by suggesting verified solutions
- To minimize hallucinated responses through controlled knowledge grounding

The chatbot operates as an assistive component and does not autonomously modify tickets or system state.

6.2RAG System Architecture Overview

The RAG chatbot is implemented as an independent service that interacts with the main ticketing system through a well-defined API layer. As illustrated in the system architecture, the chatbot service consists of two main layers:

API Layer: Handles incoming chat requests and response delivery

Core Layer: Implements the internal RAG pipeline and decision logic

The RAG service communicates with:

- The client interface for user queries
- A dedicated RAG database for knowledge retrieval
- The mini-AI service for contextual signals (priority, category, team)

This separation ensures modularity and scalability of the chatbot component.

6.3 Chat Engine Processing Pipeline (Step-by-Step Workflow)

The Chat Engine follows a structured, state-driven processing pipeline designed to manage conversational input, control execution flow, and determine whether knowledge retrieval and response generation are required. The pipeline is implemented as a state graph, where each stage performs a specific operation and conditionally routes the execution to subsequent stages based on the nature of the user query.

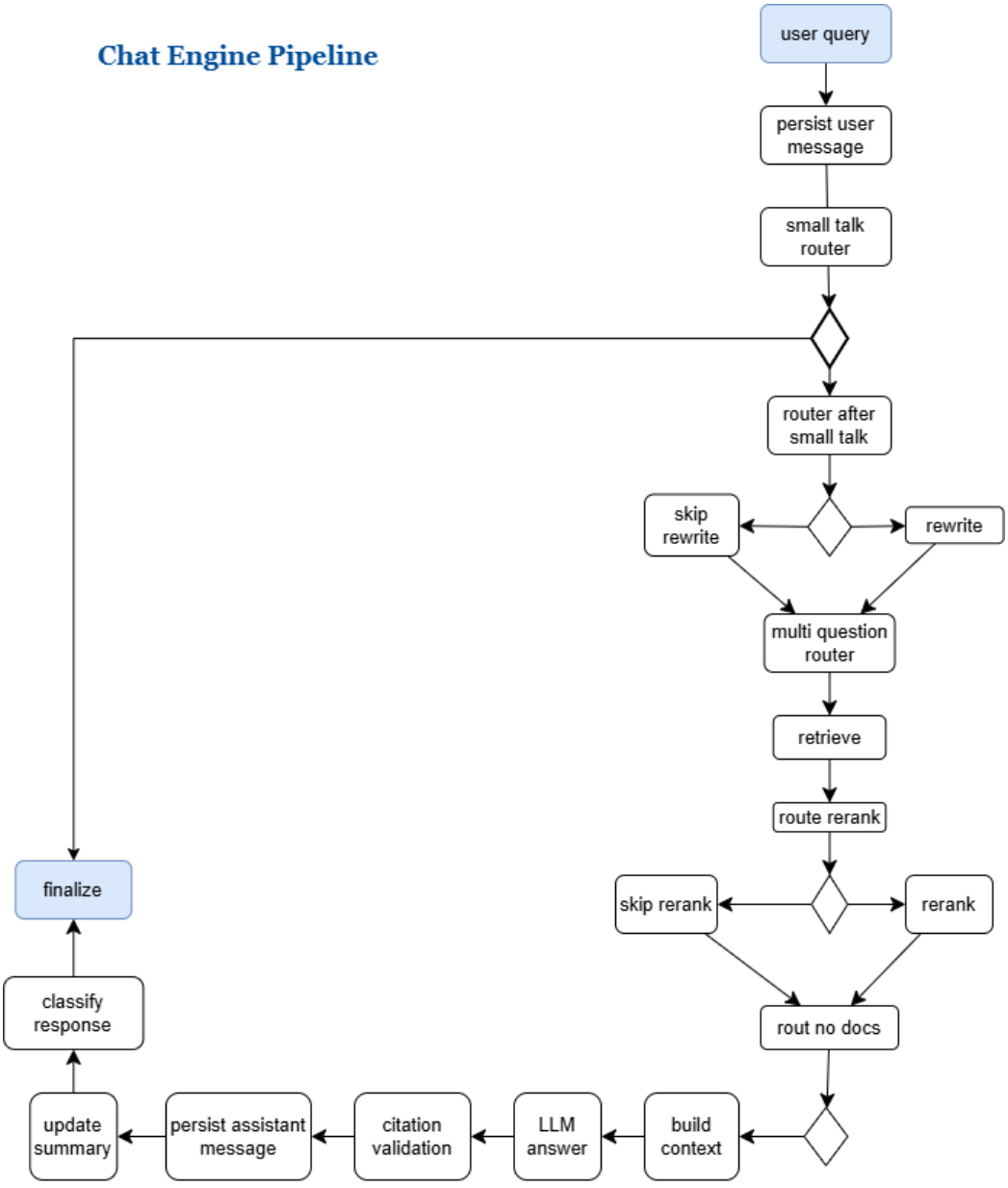


Figure 37 chat engine pipeline

Step 1: Persist User Message

Upon receiving a user query, the message is first stored to ensure conversation continuity, traceability, and support for auditing and interaction analysis.

Step 2: Small Talk Routing

The stored message is passed to the small talk router, which determines whether the input represents general conversational content or a knowledge-seeking request. This step enables early routing decisions and prevents unnecessary processing.

- If the input is classified as small talk, the pipeline bypasses all subsequent processing stages and proceeds directly to the finalize state, returning an immediate response.

Step 3: Routing After Small Talk and Query Preparation

For non-small-talk inputs, the pipeline continues to a routing stage that determines whether query rewriting is required.

- **Rewrite:** The user input is transformed into a standalone query to remove conversational dependencies.
- **Skip Rewrite:** The original input is retained when rewriting is unnecessary.

Both paths converge at the **multi question router**, which detects compound or multi-part queries and prepares them for downstream processing.

Step 4: Document Retrieval

The prepared query is forwarded to the retrieve stage, where relevant documents are fetched from the RAG database using vector similarity search. Retrieval parameters may include:

- Top-K selection
- Similarity score thresholds
- Optional metadata filtering

Step 5: Re-ranking Decision

Following retrieval, the pipeline evaluates whether re-ranking is required:

- **Re-rank:** Retrieved documents are reordered to improve relevance.
- **Skip Re-rank:** The original retrieval order is preserved.

Step 6: No-Document Routing

A routing decision evaluates whether sufficient relevant documents are available:

- **Fallback No Docs:** When no suitable documents are found, the pipeline avoids context construction and proceeds directly toward response finalization.
- **Build Context:** When relevant documents are available, they are aggregated into a structured context.

Step 7: LLM-Based Answer Generation

If a context is successfully built, the LLM answer stage generates a response grounded in the retrieved knowledge. The language model is constrained to rely on the constructed context, reducing the risk of hallucinated output.

Step 8: Citation Validation

The generated response undergoes citation validation to ensure that it is supported by retrieved documents and complies with grounding requirements.

Step 9: Persist Assistant Message and Conversation Update

The validated response is stored using persist assistant message, after which the update summary stage updates the conversation summary to support long-running interactions and efficient context management.

Step 10: Response Classification and Finalization

The response is analyzed in the classify response stage and categorized as:

- Normal
- Soft Refusal
- Hard Refusal
- Needs Human

Based on this classification, the pipeline terminates at the finalize state, returning a controlled and policy-compliant response to the user.

6.4 Solution Engine Processing Pipeline (Step-by-Step Workflow)

The Solution Engine follows a structured, decision-driven processing pipeline designed to generate validated and actionable solutions based on retrieved knowledge. Unlike the Chat Engine, the Solution Engine does not handle conversational routing or small talk, but instead focuses on refining retrieved information and producing reliable solution outputs.

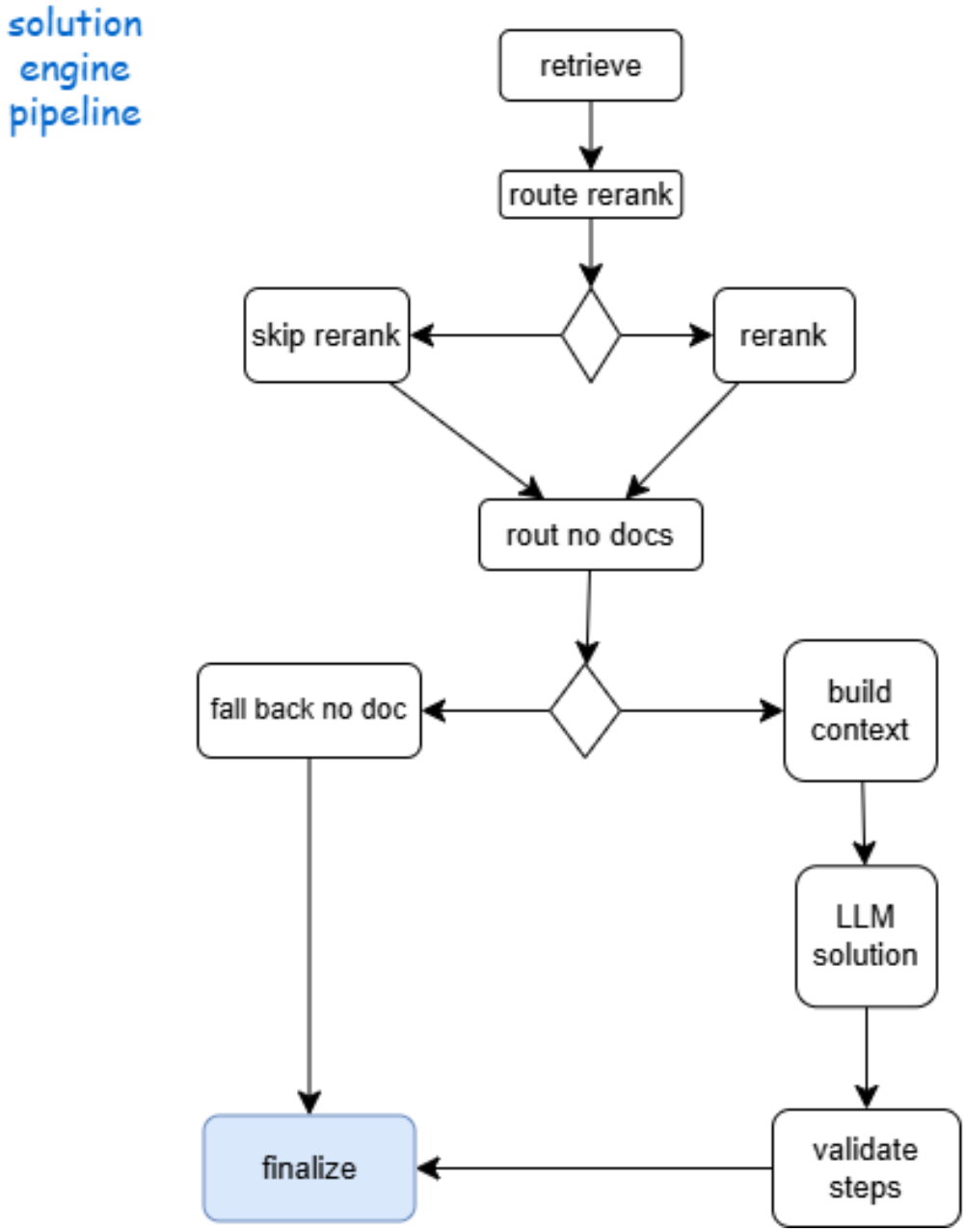


Figure 38 Solution Engine Processing Pipeline

Step 1: Document Retrieval

The pipeline begins with the retrieve stage, where candidate solution documents are retrieved from the knowledge base using vector similarity search. This step assumes that the input query has already been identified as a knowledge-oriented request and forwarded to the Solution Engine.

Step 2: Re-ranking Routing

After retrieval, the pipeline evaluates whether re-ranking is required through a routing decision:

- Re-rank: Retrieved documents are reordered to improve relevance and alignment with the query.
- Skip Re-rank: The original retrieval order is preserved when re-ranking is not necessary.

This conditional routing optimizes performance while maintaining solution quality.

Step 3: No-Document Routing

Following re-ranking (or skipping it), the pipeline evaluates the availability of relevant documents:

- **Fallback No Docs:**
- When no sufficiently relevant documents are found, the pipeline bypasses solution generation and proceeds directly to finalization. This prevents the system from generating unsupported or hallucinated solutions.
- **Build Context:**

When relevant documents are available, they are aggregated and structured into a unified context suitable for solution generation.

Step 4: LLM-Based Solution Generation

If a context is successfully constructed, the LLM solution stage generates a solution grounded in the retrieved and structured information. The language model is constrained to rely on the constructed context, ensuring that generated solutions remain accurate and actionable.

Step 5: Solution Validation

The generated solution undergoes a validate steps phase, where the output is checked for logical consistency, procedural correctness, and alignment with known policies or best practices. This validation step acts as an additional safeguard before delivering the solution.

Step 6: Finalization

The pipeline terminates at the finalize state, producing the final solution output. Depending on the execution path, the finalized output may represent either a validated solution or a controlled fallback response in cases where no reliable knowledge was available.

Design Characteristics of the Solution Engine Pipeline

The Solution Engine pipeline is intentionally designed with explicit routing and fallback mechanisms. By separating retrieval, re-ranking, validation, and finalization into clearly defined stages, the system ensures that solutions are generated only when sufficient and reliable information is available.

This design minimizes the risk of incorrect or misleading solutions and supports the operational reliability of the ticketing system.

6.5 Response Classification and Safety Control

Response classification plays a critical role in ensuring safe and reliable chatbot behavior. By categorizing responses and identifying refusal reasons, the system enforces policy compliance and prevents the chatbot from providing harmful or unauthorized guidance.

In cases where the response is classified as Needs Human, the system flags the interaction for manual review and potential administrator intervention.

6.6 Integration with the Ticketing System

The RAG chatbot is integrated with the ticketing system in a non-intrusive manner. It can:

- Answer user questions before ticket creation
- Provide guidance during ticket resolution
- Reference previous solved tickets and policies

However, the chatbot does not directly modify tickets or system data, preserving human oversight and system integrity.

6.7 Design Advantages of the RAG-Based Approach

The proposed RAG-based chatbot offers several advantages:

- Reduced hallucination through knowledge grounding
- Transparent and explainable answers via citations
- Modular architecture with clear responsibility separation
- Scalability through independent service deployment
- Enhanced user experience without compromising control

6.8 Response Classification and Safety Control

Response classification and safety control constitute a critical component of the RAG chatbot design. While retrieval-augmented generation significantly reduces hallucination by grounding responses in verified knowledge sources, an additional control layer is required to ensure that generated answers are safe, appropriate, and aligned with system policies.

In the proposed system, every generated response produced by the language model undergoes a dedicated classification step before being returned to the user. This step enables the system to assess the nature of the response, detect potential risks, and determine whether human intervention is required.

6.8.1 Response Classification Categories

The chatbot classifies each generated response into one of the following four categories:

- **Normal:**

  The response is valid, grounded in retrieved knowledge, and safe to be delivered directly to the user.

- **Soft Refusal:**

  The response partially refuses to answer the user's request while providing an explanation or alternative guidance. This category is used when the request cannot be fully satisfied but does not violate critical policies.

- **Hard Refusal:**

  The response is completely blocked due to policy constraints, safety concerns, or insufficient reliable information. In this case, the system avoids providing any potentially harmful or misleading content.

- **Needs Human:**

  The response requires escalation to a human operator or administrator. This classification is applied when the system determines that automated handling is insufficient or inappropriate, such as in sensitive, ambiguous, or high-risk scenarios.

This categorization framework allows the system to handle a wide range of interaction outcomes while maintaining strict safety boundaries.

6.8.2 Refusal Reason Identification

In addition to classifying the response, the system identifies the underlying reason for refusal when a response is categorized as either Soft Refusal or Hard Refusal. These reasons may include policy limitations, lack of sufficient contextual evidence, or the sensitive nature of the user request.

By explicitly identifying refusal reasons, the system improves transparency and provides valuable feedback for both users and administrators. This mechanism also supports future system refinement by highlighting recurring refusal patterns.

6.8.3 Human Intervention Decision

For responses classified as Needs Human, the system determines that automated resolution is not suitable. In such cases, the interaction is flagged for human review, and administrative intervention may be required. This ensures that complex or sensitive cases are handled by qualified personnel rather than relying solely on automated decision-making.

This design choice preserves human oversight in critical situations and aligns with best practices for deploying AI systems in operational environments.

6.9.4 Role of Safety Control in System Reliability

The response classification and safety control mechanism plays a key role in enhancing overall system reliability. By enforcing an additional validation layer after response generation, the system prevents unsafe outputs, reduces the risk of misinformation, and ensures compliance with internal policies.

Furthermore, this mechanism complements citation validation by ensuring that responses are not only grounded in retrieved knowledge but also appropriate for

delivery. Together, these safeguards contribute to a controlled, transparent, and trustworthy AI-assisted ticketing environment.

6.9.5 Design Benefits

The integration of response classification and safety control provides several advantages:

- Clear separation between response generation and response approval
- Reduced risk of unsafe or misleading outputs
- Explicit escalation path for complex cases
- Improved system explainability and auditability

These benefits reinforce the role of the RAG chatbot as a supportive and controlled assistant rather than an autonomous decision-maker.

6.9 Integration of the RAG Chatbot with the Ticketing System

The RAG chatbot is integrated into the ticketing system as an independent and supportive service, designed to enhance user interaction and assist in ticket resolution without directly modifying system data or overriding human decisions. This integration ensures that the chatbot operates within clearly defined boundaries while complementing existing ticket management workflows.

The chatbot communicates with the ticketing system through a dedicated API layer, allowing client applications to submit user queries and receive responses in real time. This service-oriented integration approach preserves loose coupling between the chatbot and core ticketing components, thereby improving system maintainability and scalability.

6.9.1 Interaction Flow with the Ticketing System

The integration follows a controlled interaction flow. When a user submits a question related to a ticket, the request is forwarded to the RAG chatbot service along with contextual metadata such as ticket description, category, and priority.

This contextual information enables the chatbot to generate more accurate and relevant responses.

The chatbot processes the request independently and returns a response that may include suggested solutions, procedural guidance, or references to previously resolved tickets and policies. The returned response is then displayed to the user without automatically altering the ticket's state or attributes.

6.9.2 Relationship with AI Decision Modules

The RAG chatbot operates alongside other AI components in the system, including sentiment-based priority estimation, ticket categorization, and automatic team assignment. However, it remains functionally independent from these modules.

The chatbot may consume outputs from these AI components as contextual signals, but it does not influence or override their decisions. This separation ensures that conversational assistance does not interfere with operational decision-making processes such as ticket routing or escalation.

6.9.3 Human Oversight and Escalation Support

In cases where the chatbot response is classified as Needs Human, the integration allows the ticketing system to flag the interaction for manual review. This enables administrators or support agents to intervene when automated assistance is insufficient or inappropriate.

By supporting controlled escalation rather than automated action, the integration maintains human oversight and aligns with best practices for deploying AI systems in customer support environments.

6.9.4 Integration Benefits

The integration of the RAG chatbot with the ticketing system provides several practical benefits:

- Improved user experience through immediate, context-aware assistance

- Reduced number of repetitive or low-value tickets
- Support for faster ticket resolution through knowledge reuse
- Preservation of system integrity through non-intrusive integration

This design ensures that the chatbot functions as an intelligent assistant rather than an autonomous system component.

6.9.5 Architectural Consistency

The integration aligns with the overall system architecture, where each service has a clearly defined responsibility. By isolating the RAG chatbot as a separate service, the system remains extensible and adaptable to future enhancements, such as expanding the knowledge base or introducing additional safety mechanisms.

7. Summary

This chapter presented the design and implementation of artificial intelligence techniques integrated into the proposed ticketing system. The focus was on structuring AI components as independent yet complementary modules that enhance automation, decision support, and user assistance while preserving system reliability and human oversight.

The chapter first introduced natural language processing techniques applied to sentiment analysis and ticket categorization. Sentiment analysis was utilized to support priority estimation using a lightweight, pre-trained approach, while ticket categorization employed a pre-trained BERT model to classify tickets into predefined categories without additional fine-tuning. Automatic team assignment was then addressed as a supervised learning task, where both SVM and BERT models were trained and evaluated on a real-world dataset. The comparative analysis demonstrated the superior ability of BERT to capture contextual information, thereby justifying its selection as the final model for ticket routing.

In addition, the chapter detailed the design of a Retrieval-Augmented Generation (RAG) chatbot, emphasizing its role in delivering context-aware and knowledge-grounded responses. The chatbot architecture was explained through the RAG

processing pipeline and state graph workflow, which illustrated the execution logic and conditional routing mechanisms. A dedicated response classification and safety control layer was also presented to regulate output quality and support human intervention when required. Furthermore, the integration of the chatbot into the ticketing system was discussed, highlighting a non-intrusive, service-oriented design that preserves overall system integrity.

Overall, the AI design presented in this chapter demonstrates a modular, interpretable, and scalable approach to integrating artificial intelligence into a ticketing system. By combining automated decision-making with explicit control mechanisms and human-in-the-loop principles, the proposed system achieves a balanced and reliable AI-assisted support environment.

Chapter 7 Practical implementation

1. Introduction

In this chapter, we will present the practical implementation of the system, detailing the technologies and tools employed. Additionally, we will showcase the system interfaces and conclude by executing test cases to verify the system's functionalities across different scenarios.

2. Used tools

1. Django (Backend Framework):

Django is a high-level Python web framework designed to support rapid development and clean, maintainable system design. It provides a structured approach to web application development and integrates core components such as models, views, and URL routing.

In this project, the Django REST Framework (DRF) is used to implement RESTful APIs. DRF extends Django's capabilities by providing tools for API serialization, authentication, and request handling, enabling efficient communication between the backend services and client applications.

2. React (Frontend Framework):

React is a JavaScript library used for building modern and interactive user interfaces. It follows a component-based architecture that promotes reusability and efficient UI updates. React enables dynamic rendering of user interfaces based on changes in application state, making it suitable for developing responsive and scalable web applications.

In the proposed system, React is used to implement the frontend interface, allowing seamless interaction between users and the ticketing platform.

3. My SQL Database:

MySQL is an open-source relational database management system (RDBMS) used for storing and managing structured data. It is known for its reliability, scalability, and support for complex queries.

In this system, MySQL is used to persist application data, including user information, tickets, and system logs, ensuring data consistency and integrity.

4. Visual studio code (VS code):

Visual Studio Code is a lightweight yet powerful source code editor that supports multiple programming languages and development tools. It provides features such as syntax highlighting, debugging support, and integrated version control.

Visual Studio Code is used as the primary development environment for implementing both frontend and backend components of the system.

5. Bitbucket:

Bitbucket is a Git-based version control platform that supports collaborative software development. It provides repositories for source code management and integrates with continuous integration tools to track changes and manage development workflows.

In this project, Bitbucket is used to manage source code versions and facilitate team collaboration.

3. AI Used Technologies and Tools

- **SpaCY**

  SpaCy is a robust Natural Language Processing (NLP) library designed for efficient and accurate text processing ,making it ideal for production use. It provides advanced capabilities such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. By integrating SpaCy with SpacyTextBlob, we leverage SpaCy's efficient text processing pipeline and TextBlob's sentiment analysis to classify review sentiments. This combination allows us to process and analyze large volumes of text data quickly and accurately, making SpaCy an essential tool for our NLP tasks.

- **BERT and Transformers**

  A BERT-based model from the Hugging Face Transformers library is utilized for text classification tasks within the system. BERT enables deep contextual

understanding of textual data, making it suitable for ticket categorization and automatic team assignment.

Pre-trained BERT models are employed to leverage contextual language representations, while fine-tuning is applied selectively where supervised learning is required.

- **Python**

  Python serves as the primary programming language for implementing artificial intelligence components and integrating machine learning models into the system. Its extensive ecosystem of libraries and frameworks supports efficient development of data processing pipelines and AI services.

4. System interfaces

- **Register**

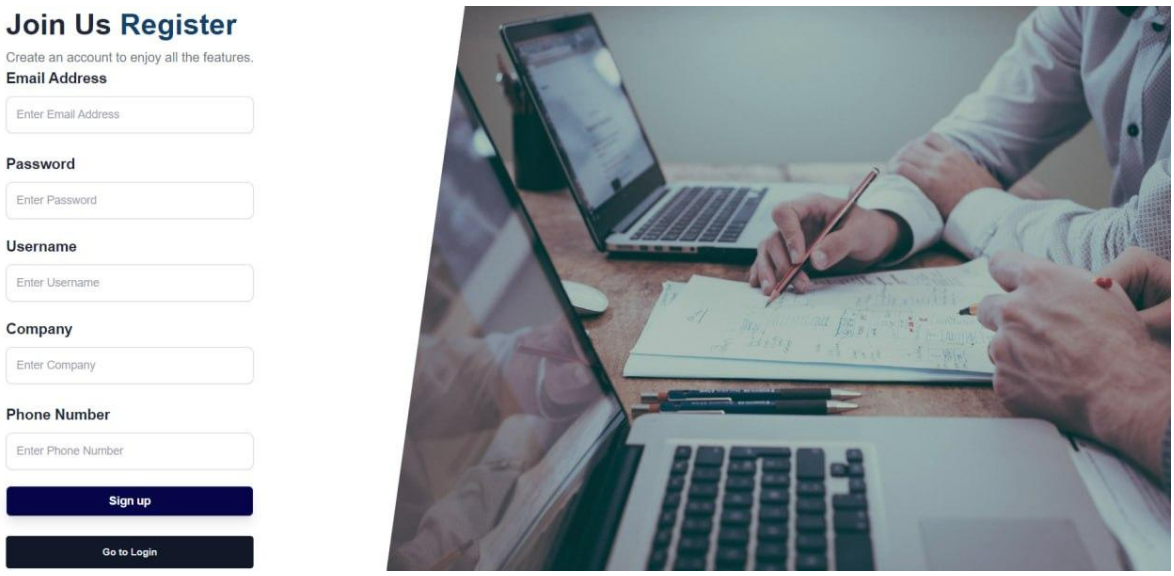


Figure 39 Sign up interface

- **Log in**

Figure 40 Log in interface

- **Profile management**

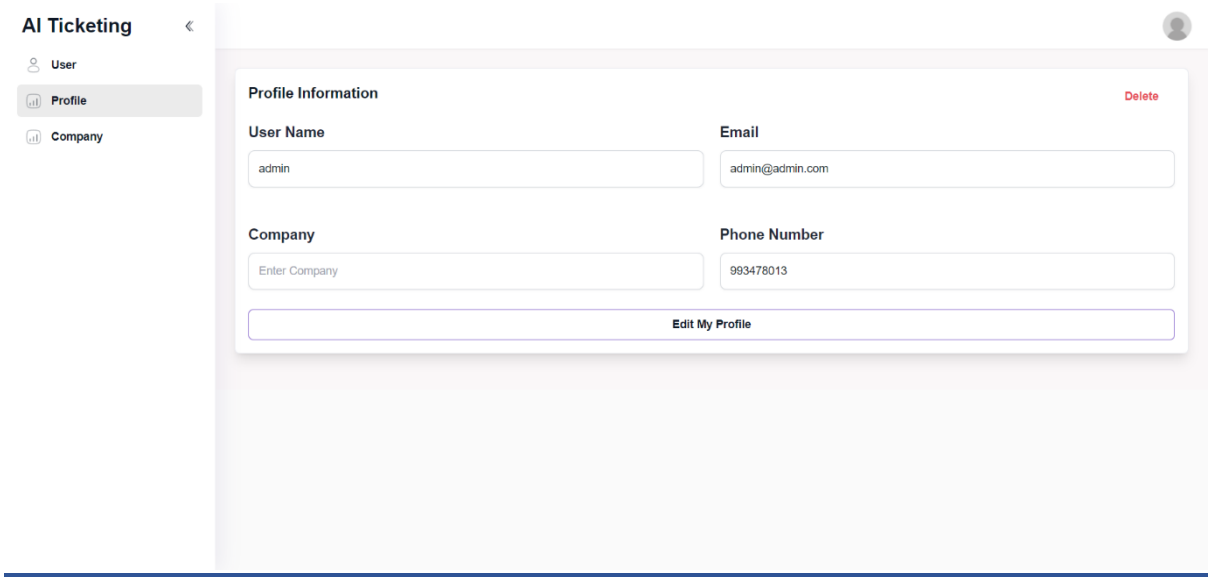


Figure 41 Profile management interface

i. Admin's interfaces

- **Users' management**

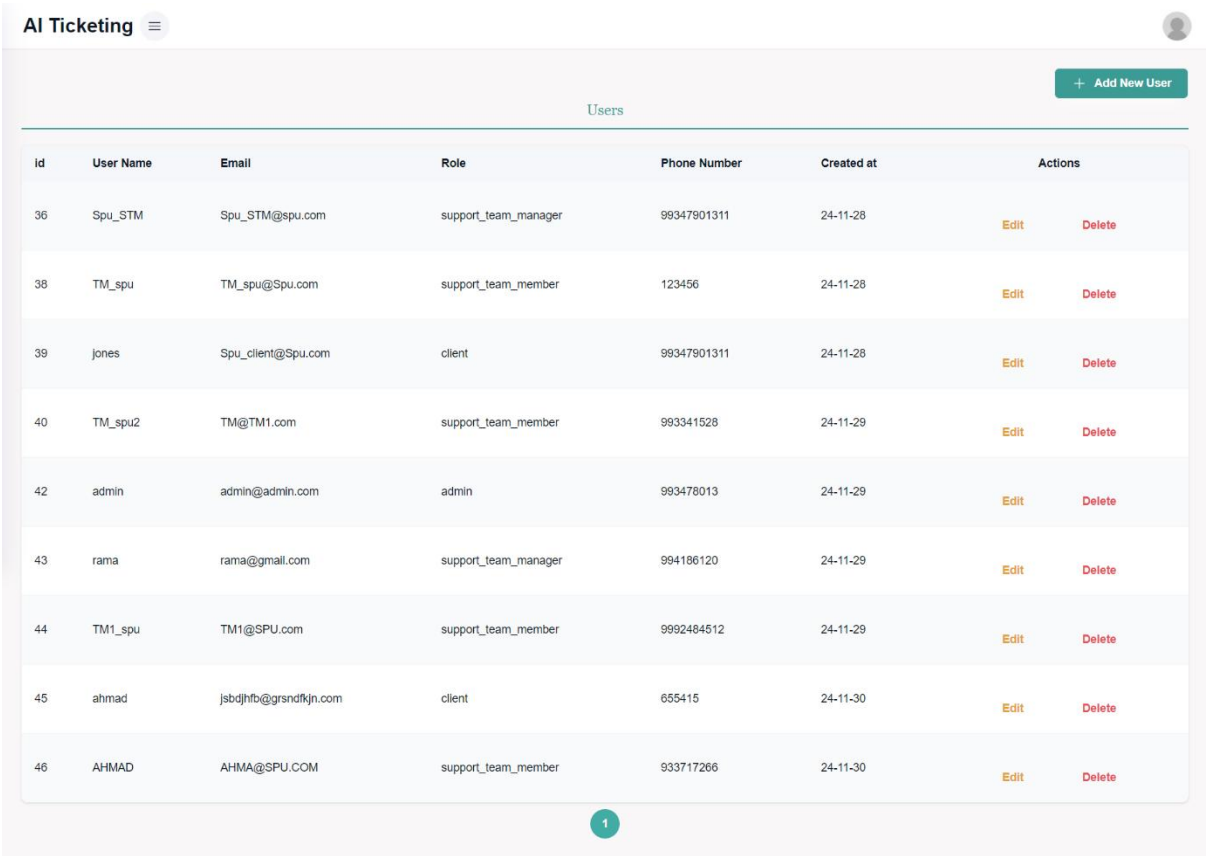


Figure 42 Users management intareface

- **add new user**

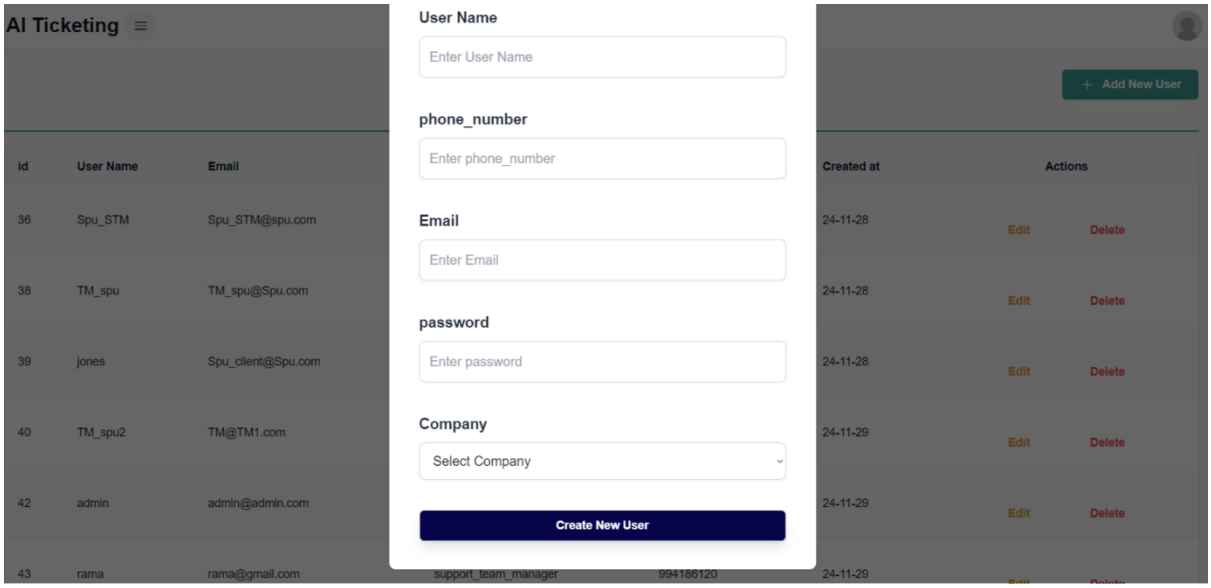


Figure 43 add new user interface

- **Company management**

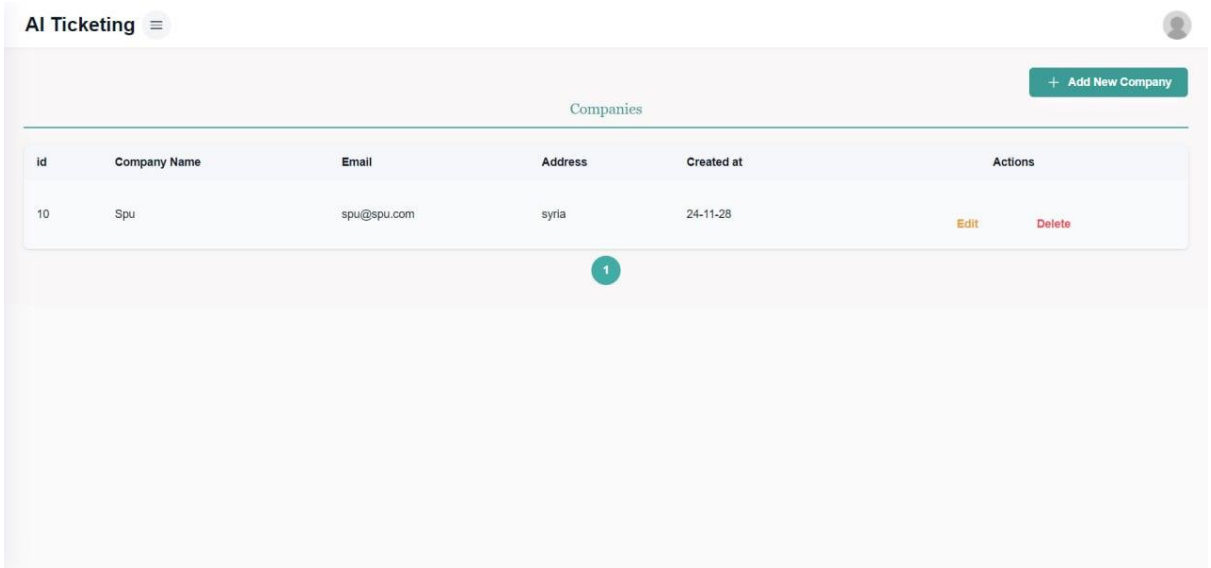


Figure 44 Company management interface

- **add new company**

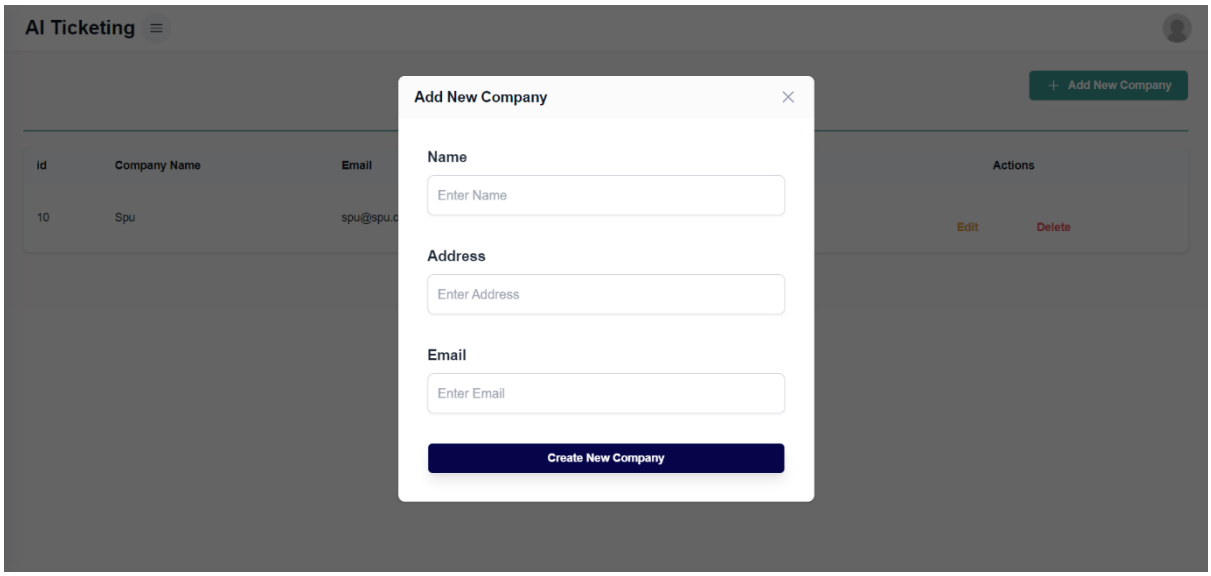
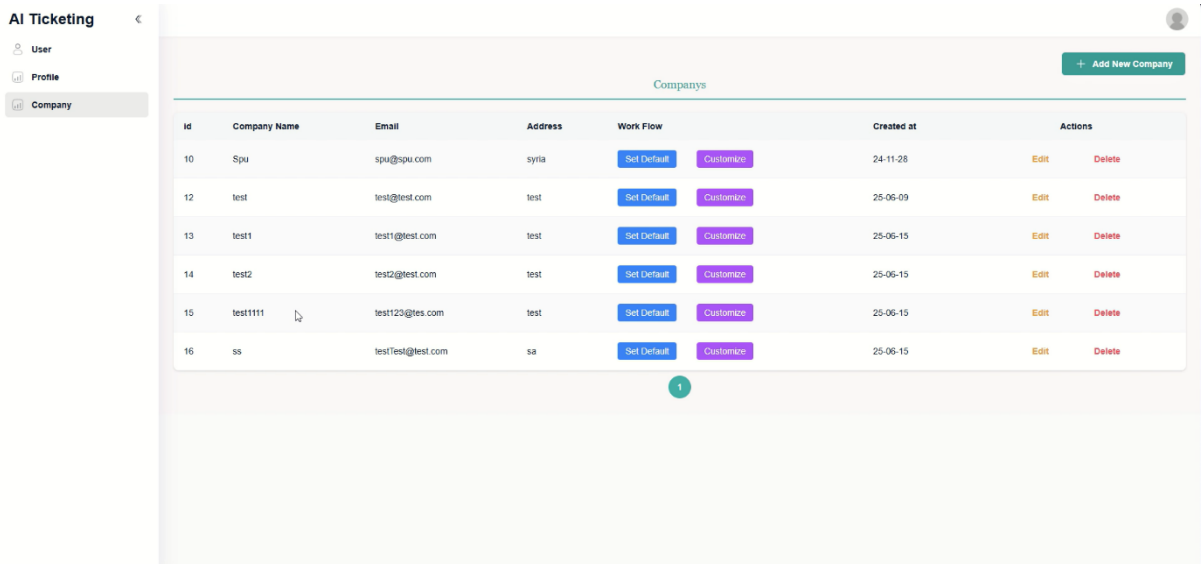


Figure 45 add company interface

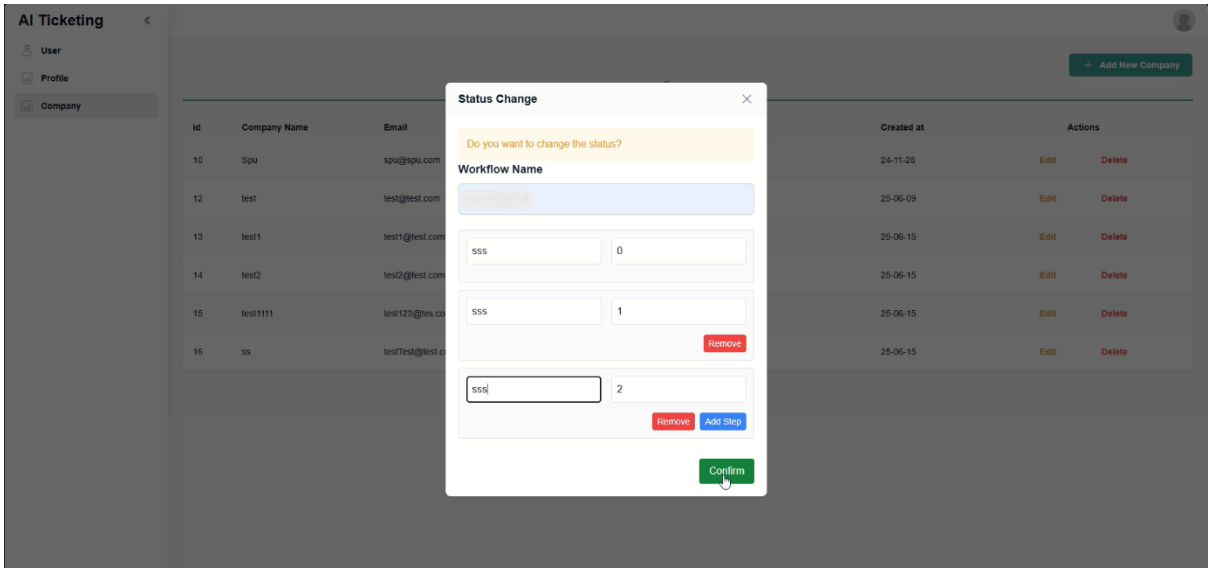
- **Workflow management**

Figure46 workflow management interface



This interface represents the Company Management section of the AI Ticketing System (Admin view). It allows admins to manage company profiles and configure their workflow behavior.

Figure47 create new workflow interface



This pop-up interface appears when the Admin clicks "Customize" under the Work Flow column for a specific company in the Company Management table.

ii. Support team manager interfaces

- **Ticket management**

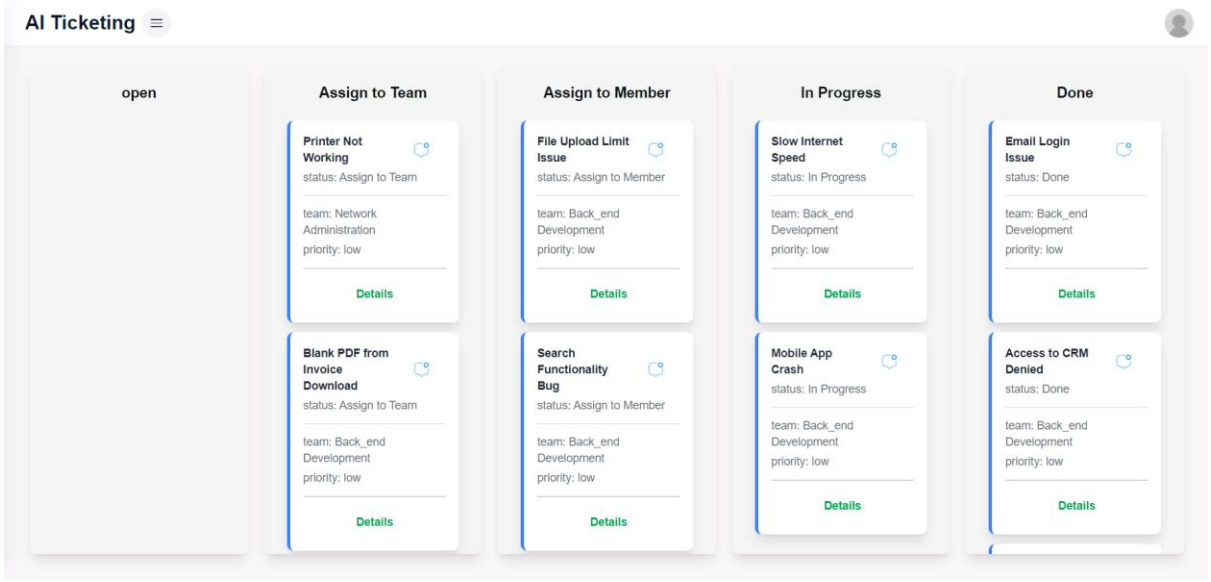


Figure 48 Ticket management interface

- **teams' management**

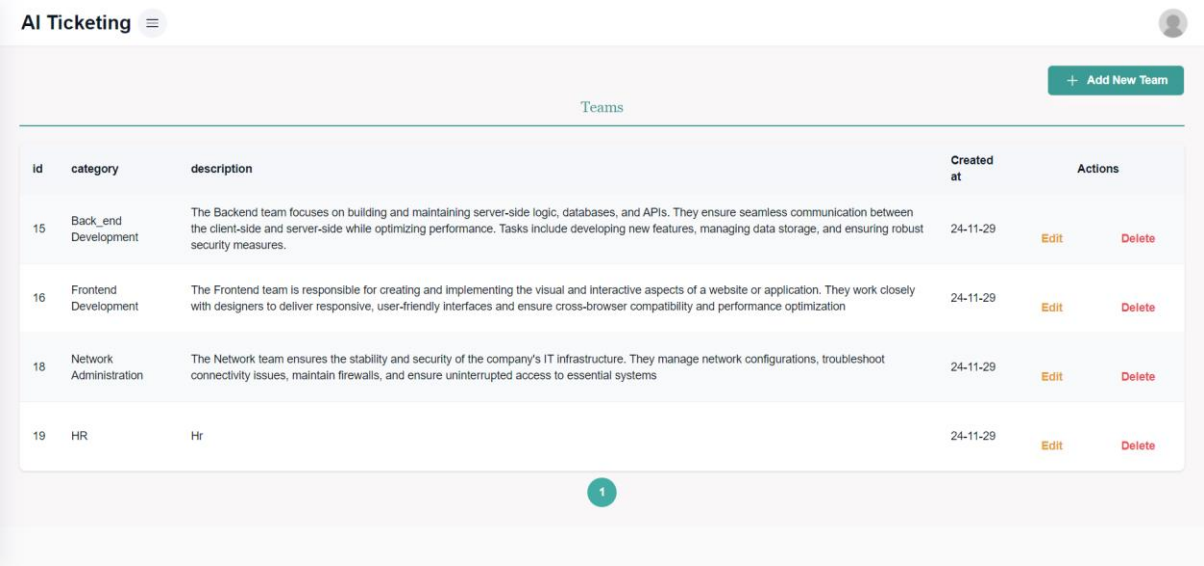


Figure 49 teams management interface

- **add new team**

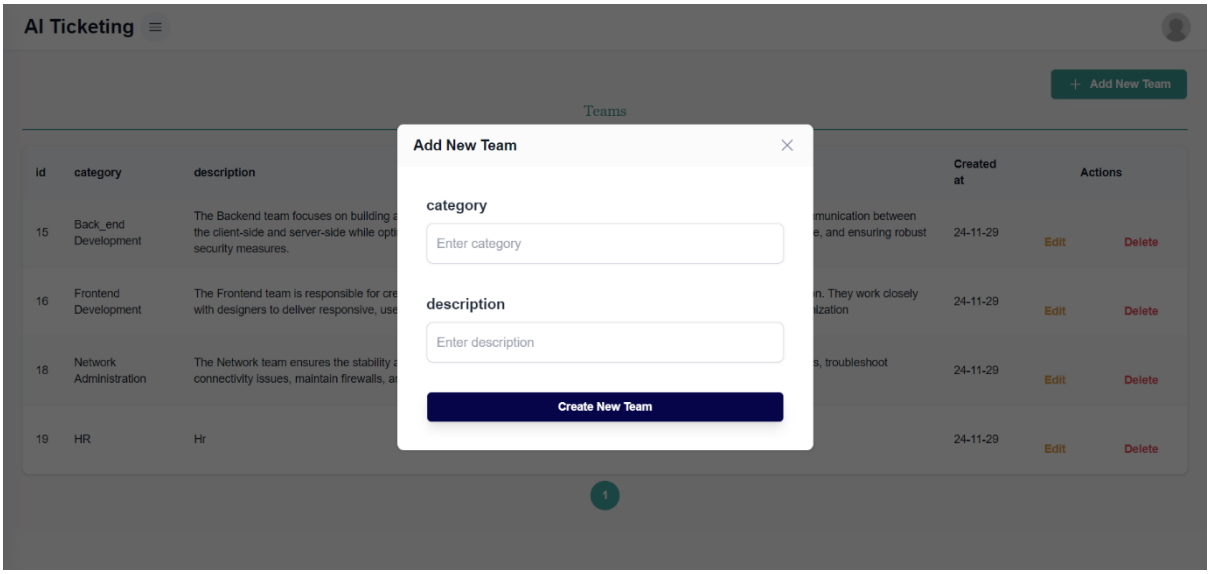


Figure 50 add new team interface

- **edit team**

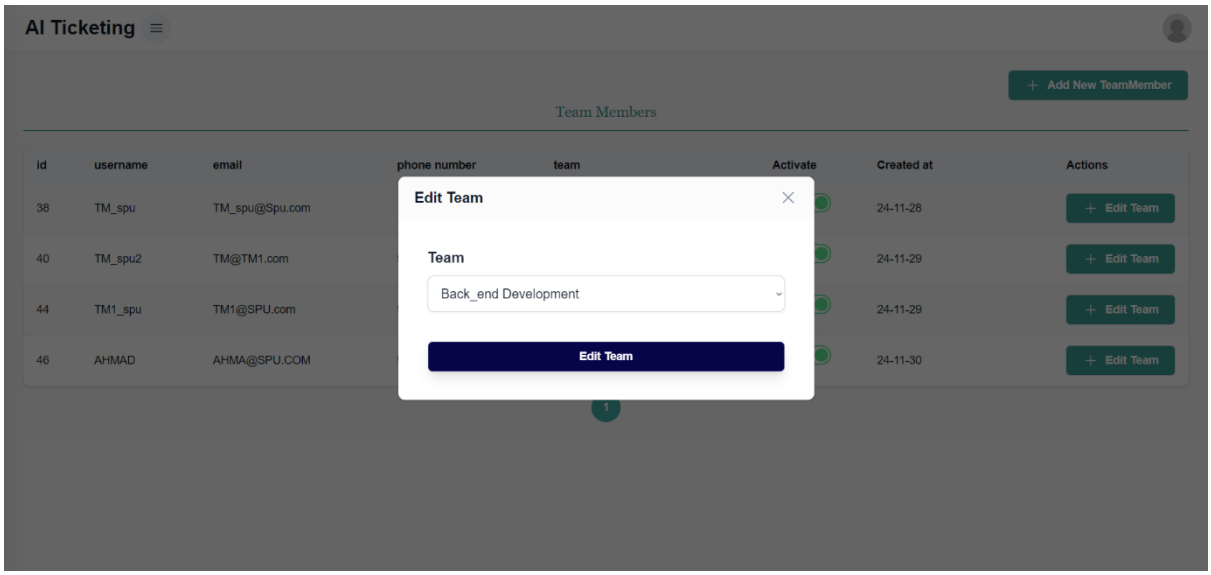


Figure51 edit team interface

- **member management**

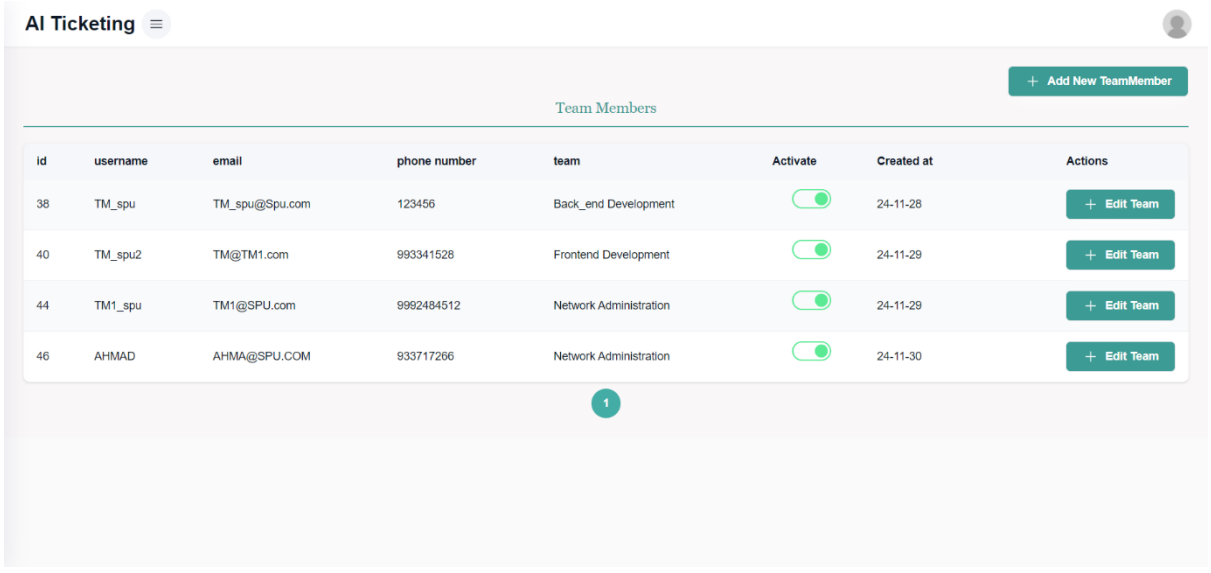


Figure 52 member management interface

- **add new member**

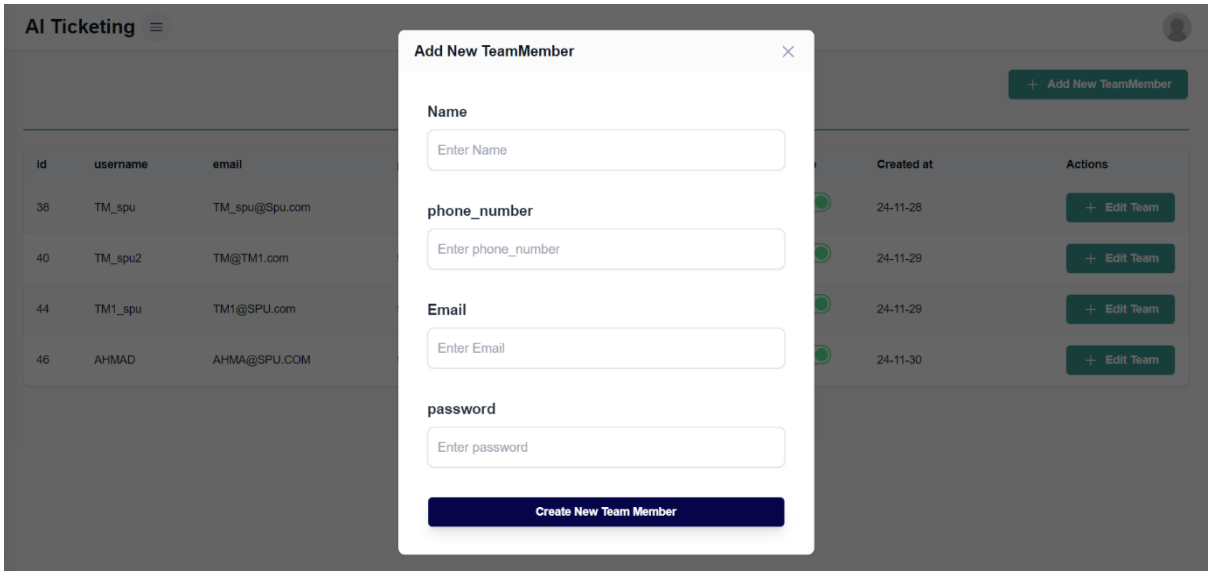


Figure 53 add new member interface

- **Tickets log**

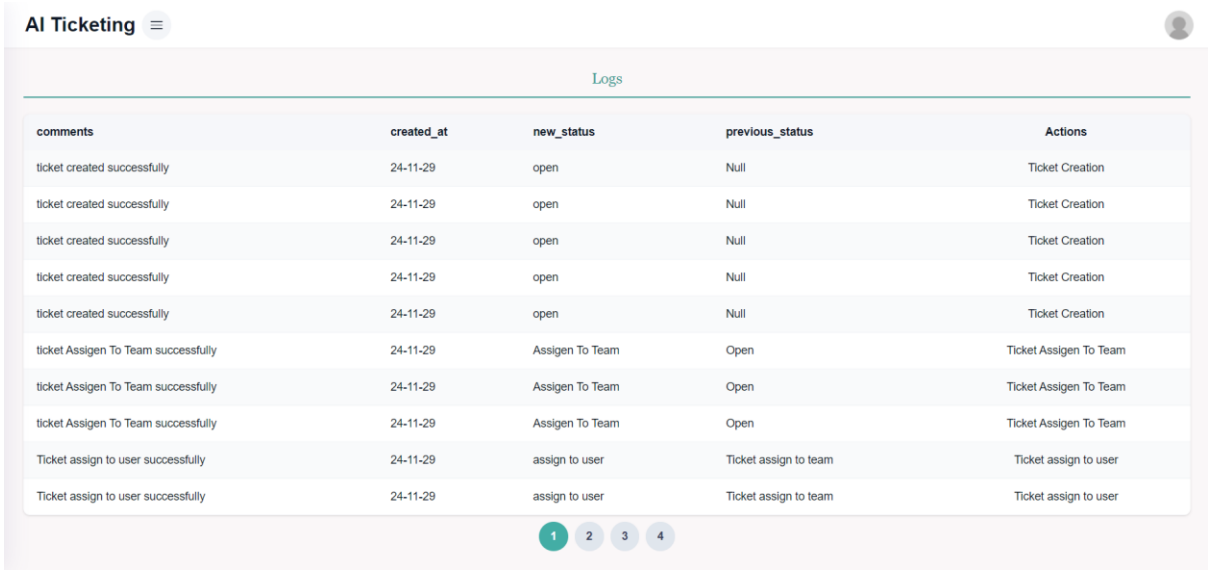


Figure 54 tickets log interface

iii. Support team member interfaces

- **Tickets assign to member**

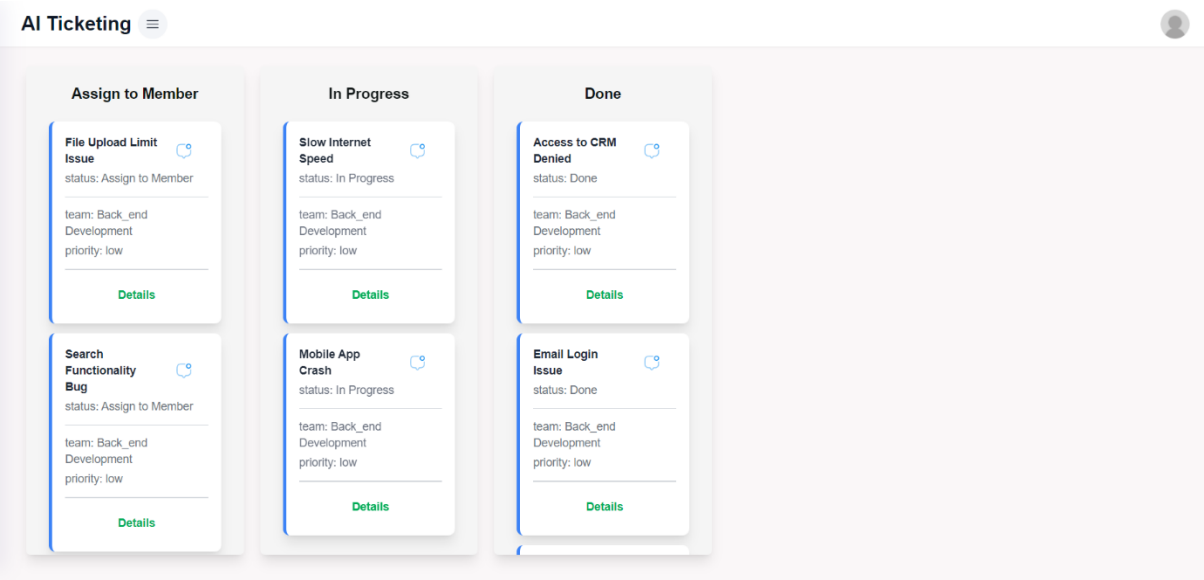


Figure 55 tickets assign to member interface

- **confirmation prompt before editing the ticket status**

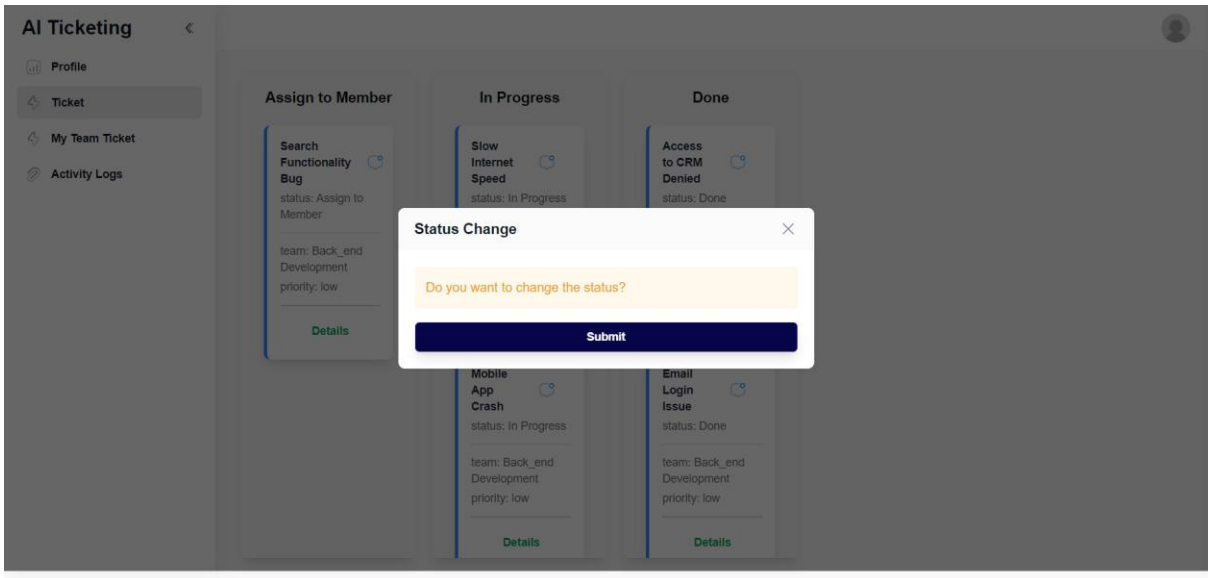


Figure 56 confirmation prompt interface

- **Tickets assign to team**

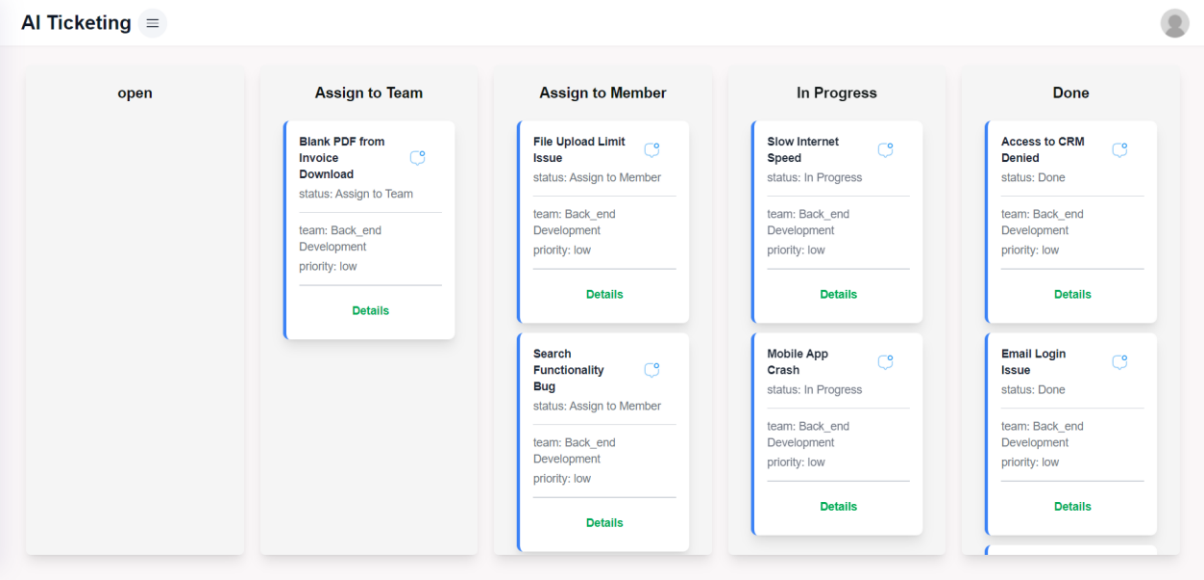


Figure 57 Tickets assign to team interface

- **Tickets log**

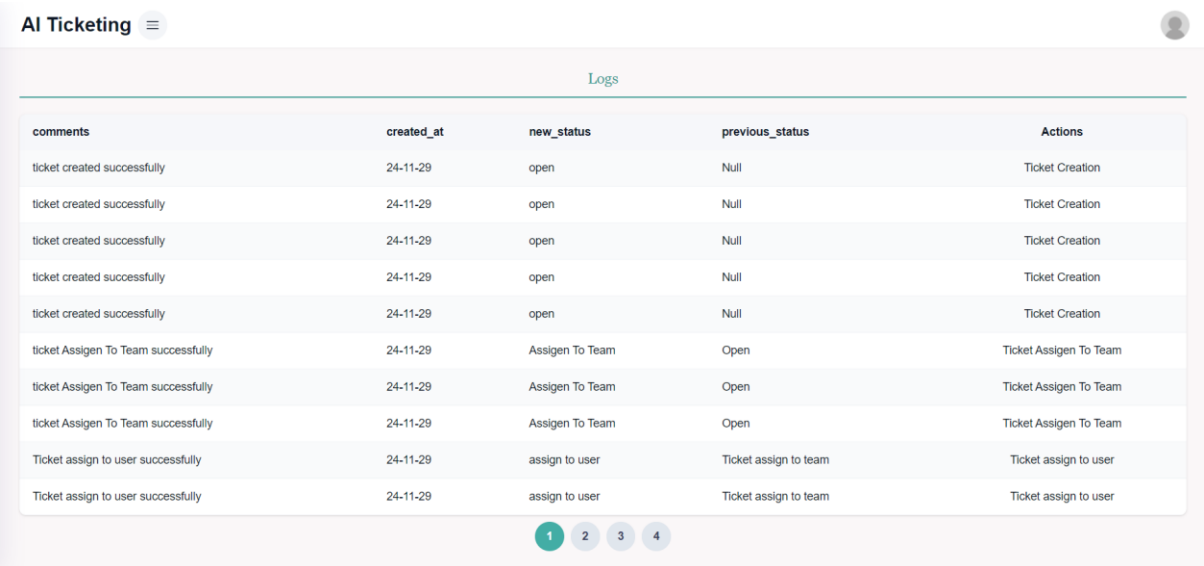


Figure 58 tickets log interface

iv. Client interfaces

- **List of tickets**

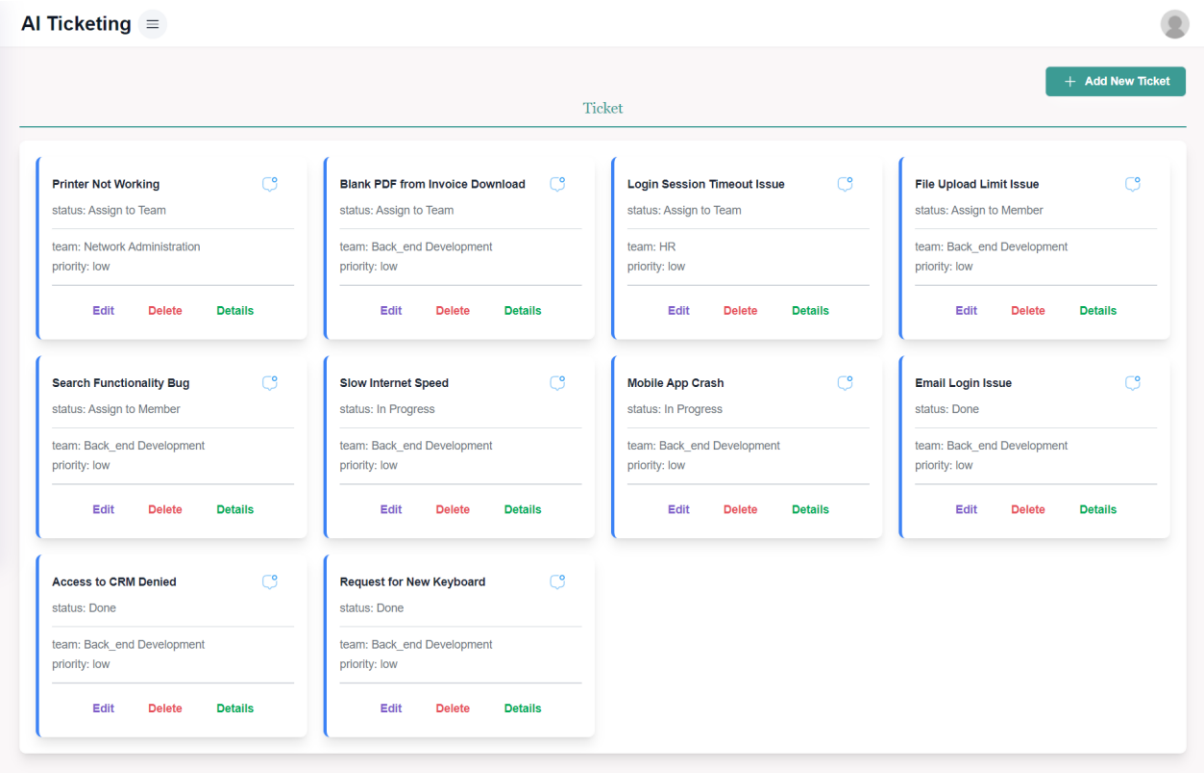


Figure59 List of tickets interface

- **Add new ticket**

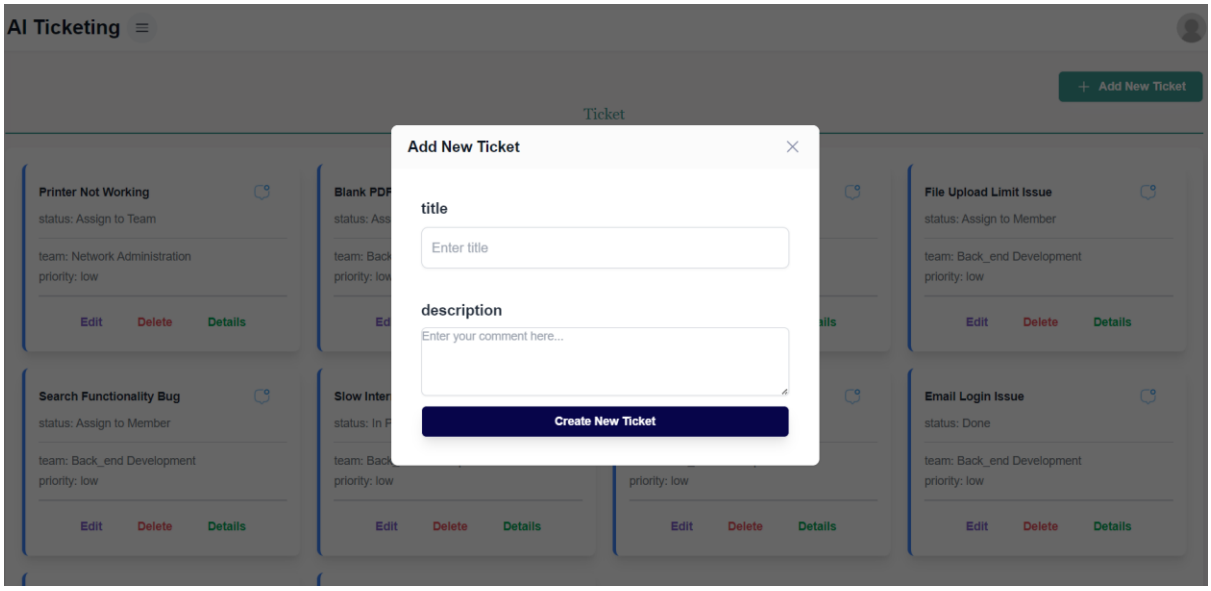


Figure 60 Add new ticket interface

There are shared interfaces among multiple users (STM, STMe, and clients)

- **Ticket details**

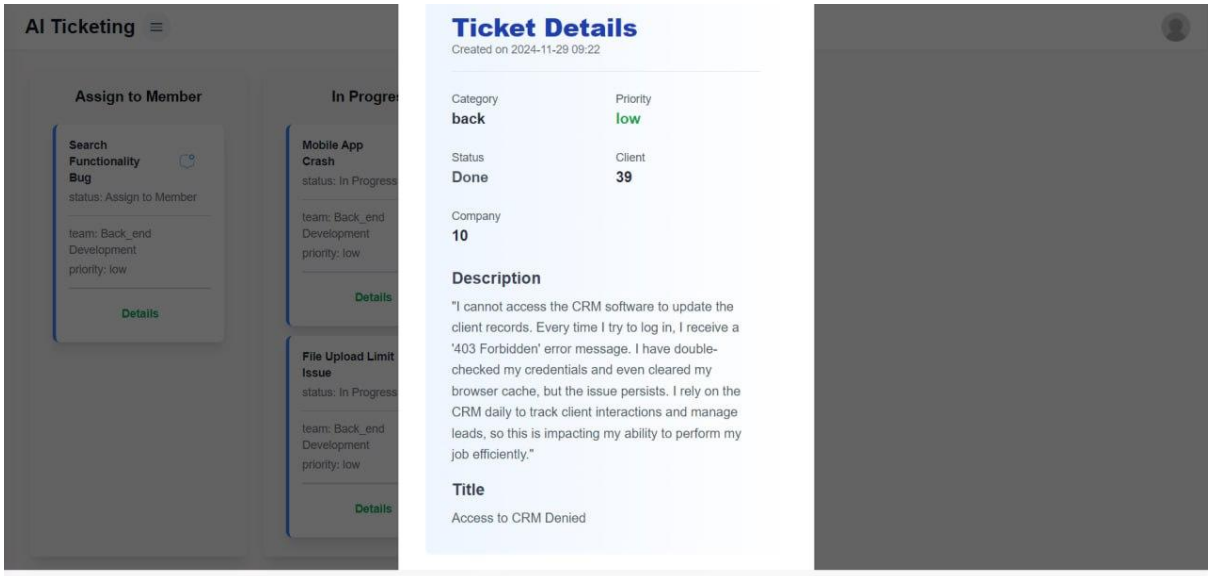


Figure 61 ticket details

- **Add comment to ticket**

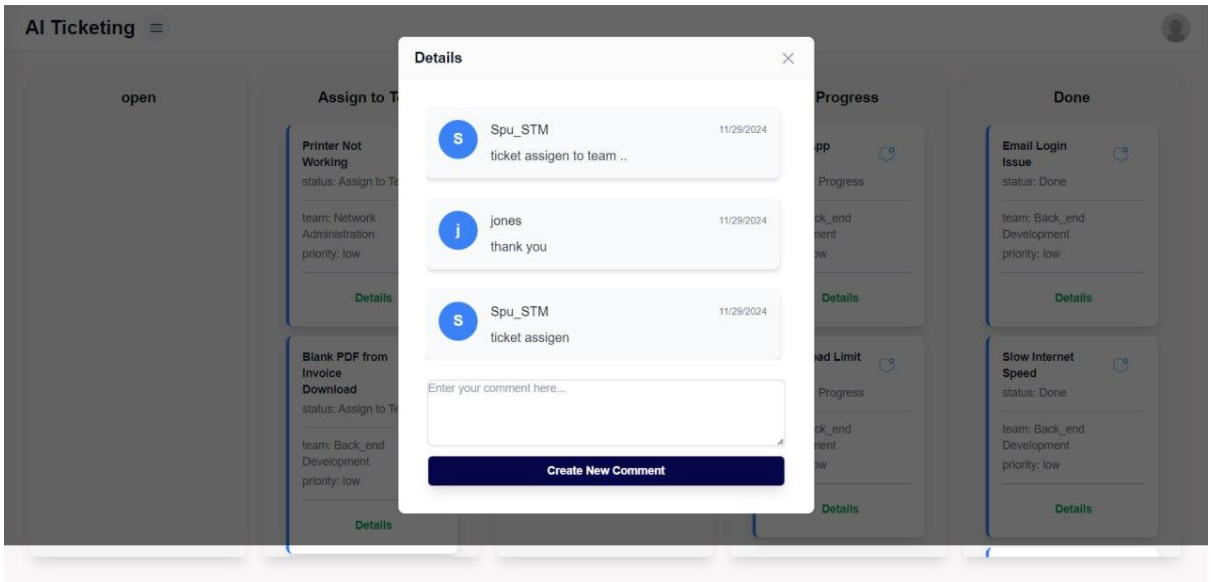
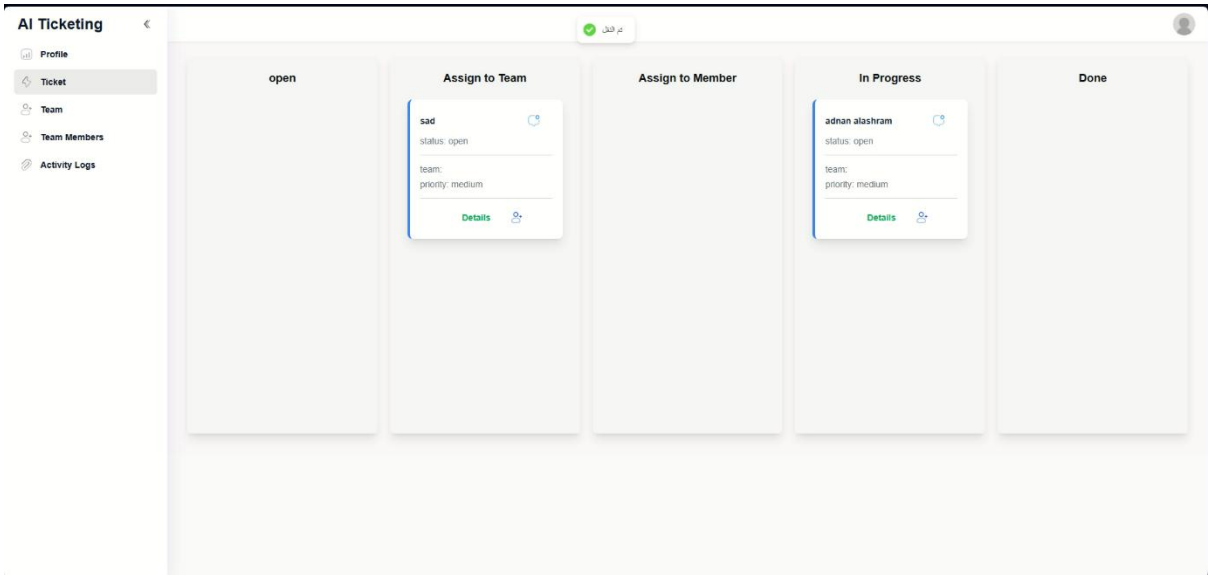


Figure 62 add comment to ticket interface

- **Default workflow**

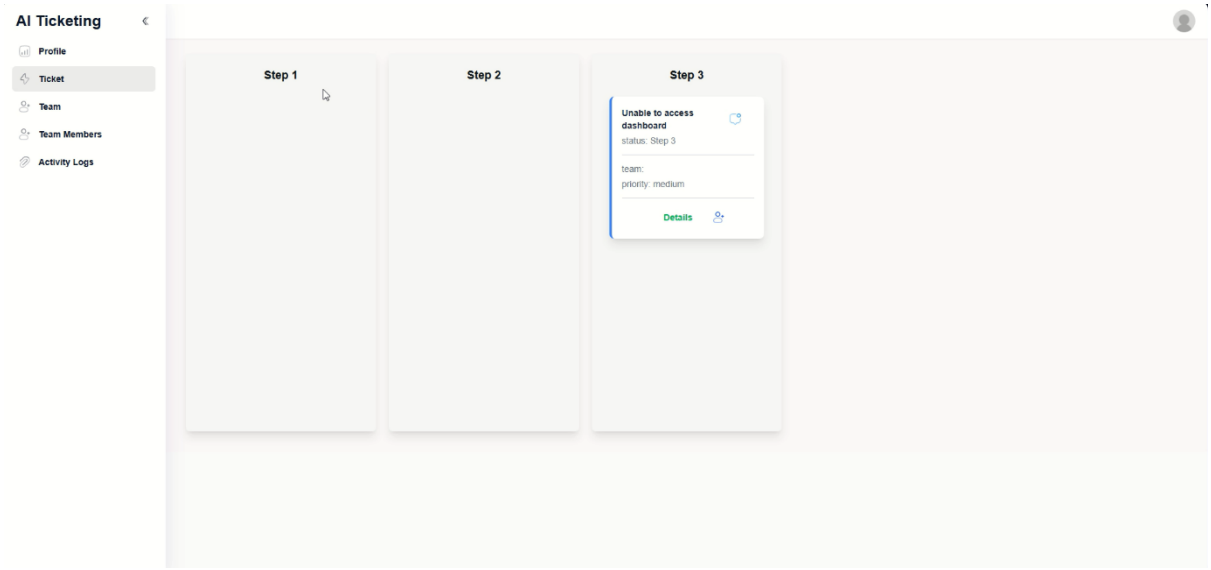
Figure63 Default workflow interface



This interface showcases the Default Workflow used in the AI Ticketing System. It represents the standard ticket lifecycle used across companies that do not use a custom workflow.

- **Customize workflow**

Figure64 Customize workflow interface



This interface represents an active custom workflow view for managing tickets in the AI Ticketing System. It replaces the default workflow with admin-defined steps.

- **Reporting and analysis**

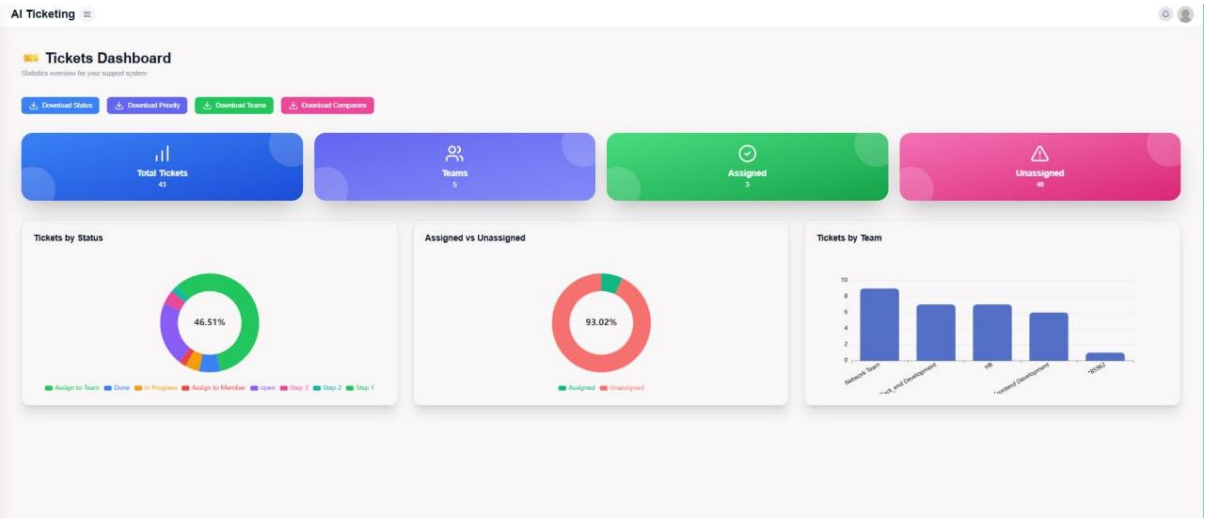


Figure 65 Reporting and analysis interface

- **Company AI Configuration Dashboard**

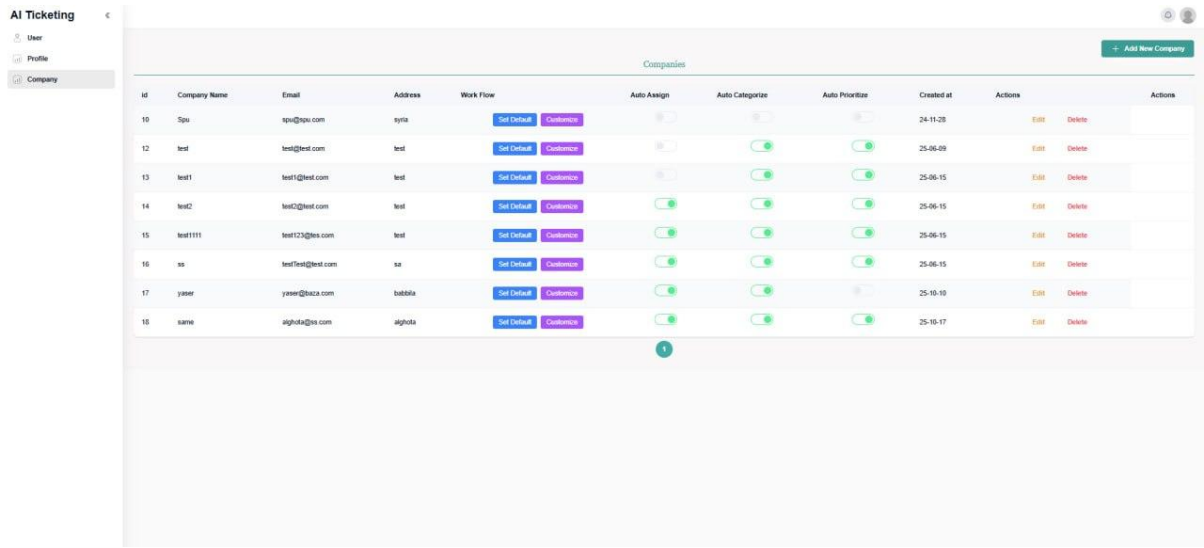


Figure 66 Company AI Configuration Dashboard

This interface represents the company management and configuration dashboard within the AI-Powered Ticketing System. It allows the **Administrator** to manage company information and configure system behavior on a per-company basis. Through this interface, the Admin can assign either a default or customized workflow and enable or disable AI features such **as Auto Team Assignment, Auto Ticket Categorization, and Auto Ticket Prioritization**

This dashboard provides flexibility and centralized control, allowing each company to use the system according to its operational needs while maintaining consistent system governance.

- **RAG-Based AI Chatbot Interface**

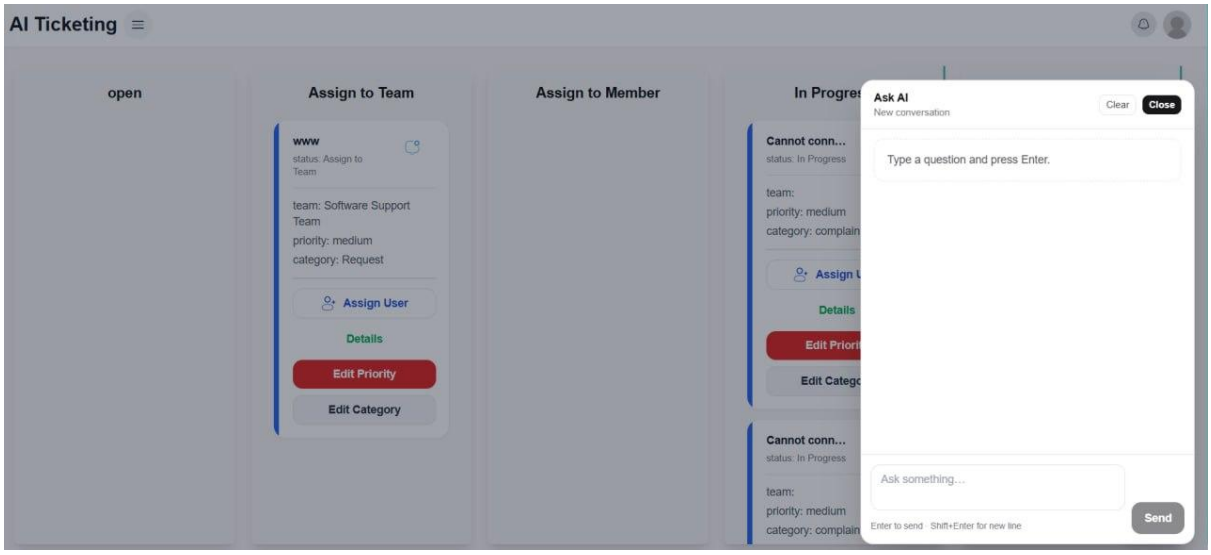


Figure 67 RAG-Based AI Chatbot Interface

This interface enables support team members to interact with the RAG-based AI chatbot while handling tickets. It supports conversational queries and solution suggestions by retrieving relevant knowledge base content to assist in ticket resolution without performing actions automatically.

- **Knowledge Base Upload Interface**

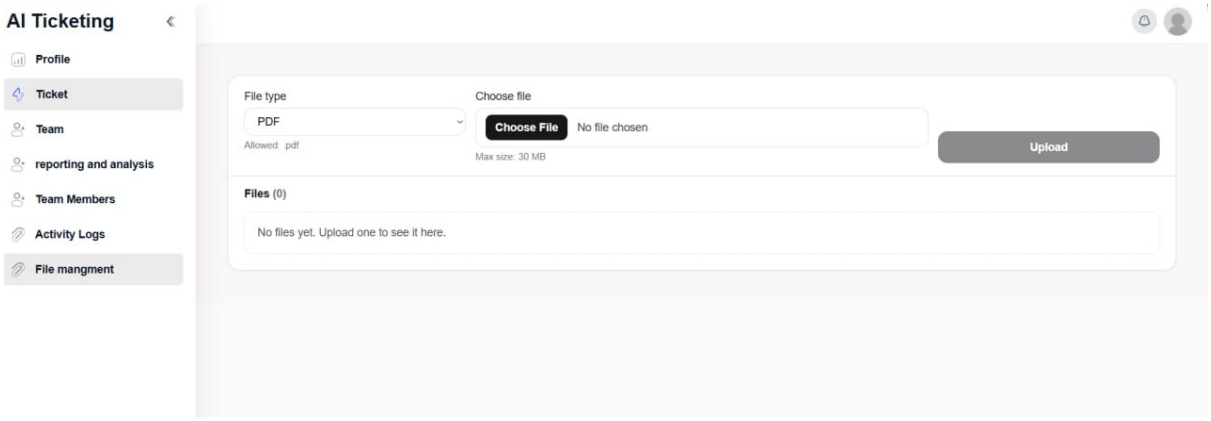


Figure 68 Knowledge Base Upload Interface

This screen provides a simple upload mechanism for adding new documents to the Knowledge Base. Users can select the file type, upload files, and manage stored documents, which are later used by the AI system during information retrieval.

- **Knowledge Base / File Management Interface**

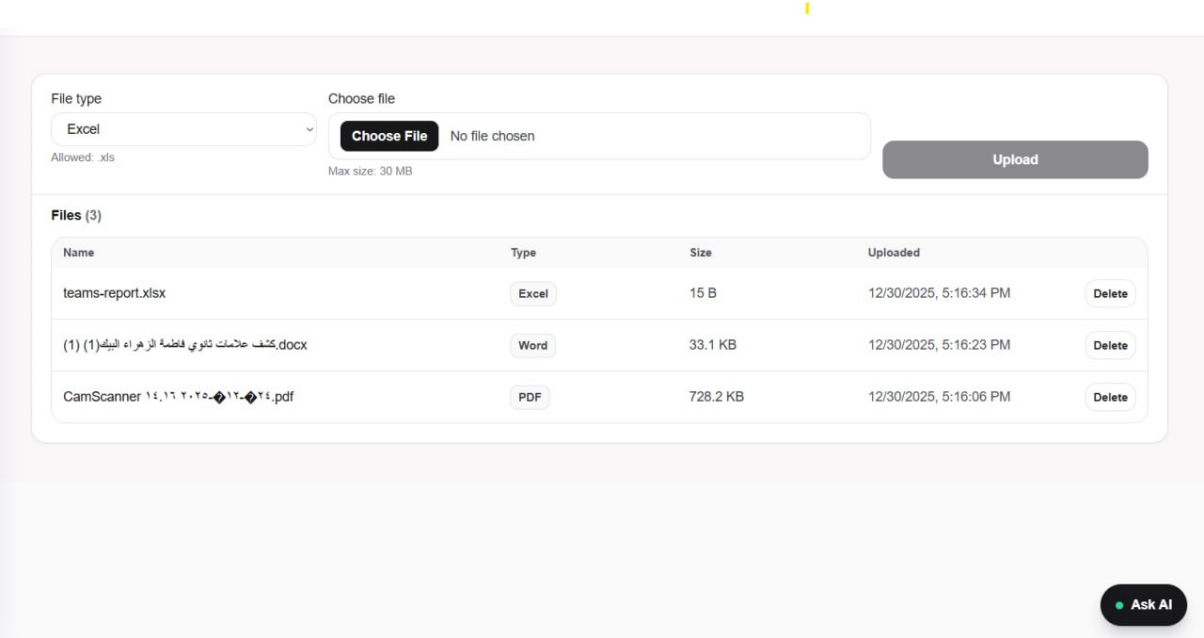


Figure 69 Knowledge Base / File Management Interface

This interface allows authorized users to upload, view, and delete files used as a Knowledge Base for the RAG-based AI chatbot. Supported file types (e.g., PDF, Word, Excel) can be managed to ensure the chatbot retrieves up-to-date and relevant information.

5. Test cases execution

Table 42 Test cases execution

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-01	Check results on display the user's dashboard.		The system should display the user info.	The system should display the user info.	Pass
Tc-02	Check results on add new user (STM)	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	“New user (STM) created successfully”	“New user (STM) created successfully”	Pass
Tc-03	Check results on add new user (STM) but user already exist	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	Error message “user already exist with this email”	Error message “user already exist with this email”	Pass
Tc-04	Check results on add new user with Missing Fields	User Name: phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	Error message “fields required “	Error message “fields required “	Pass
Tc-05	Check results on add new user with invalid information	User Name: “yaser” phone_number:”0993479013” Email: yasergmail.com Password: 1578 Company: “spu”	Error message “invalid information”	Error message “invalid information”	pass
Tc-06	Check results on update user info	User Name: “nuha” phone_number :”0993479013” Email : yaser@gmail.com Password : 1234578	user updated successfully	user updated successfully	pass
Tc-07	Check results on update user info with invalid information	User Name: “nuha” phone_number:”0993479013” Email: yasergmail.com Password: 1234	Error message “invalid information”	Error message “invalid information”	pass
Tc-08	Check results on delete user		User deleted successfully	User deleted Successfully	Pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-09	Check results on update account info.	User Name: “nuha” phone_number :”0993479013” Email: yaser@gmail.com Password: 1234578	Account updated successfully	Account updated successfully	pass
Tc-10	Check results update account info with invalid information	User Name: “nuha” phone_number:”0993479013” Email: yasergmail.com Password: 1578	Error message “invalid information”	Error message “invalid information”	pass
Tc-11	Check results on update account info without any changes	User Name: “nuha” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	Error message “No changes made “	Error message “No changes made “	Pass
Tc-12	Check results on display the Companies dashboard of the admin.		The system should display the Companies info.	The system should display the Companies info.	Pass
Tc-13	Check results on add new Company	Name: “SPU” Address: “Damascus” Email: spu@spu.com	“New Company created successfully”	“New Company created successfully”	Pass
Tc-14	Check results on add new Company but Company already exist	Name: “SPU” Address: “Damascus” Email: spu@spu.com	Error message “Company already exists with this email”	Error message “Company already exists with this email”	Pass
Tc-15	Check results on add new Company with Missing Fields	Name: “SPU” Address: “Damascus” Email:	Error message “fields required”	Error message “fields required”	pass
Tc-16	Check results on add new Company with invalid information	Name: “SPU” Address: “Damascus” Email: spu/spu.com	Error message “invalid information”	Error message “invalid information”	pass
Tc-17	Check results on update Company info	Name: “SPU2” Address: “Damascus” Email: spu@spu.com	“Company updated successfully”	“Company updated successfully”	Pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-18	Check results on update Company info with invalid information	Name: “SPU2” Address: “Damascus” Email: spu/spu.com	Error message “invalid information”	Error message “invalid information”	pass
Tc-19	Check results on delete Company		Company deleted successfully	Company deleted Successfully	Pass
Tc-20	Check results on add new team.	Category: “Back_end Development “ Description: “ The Backend team focuses on building and maintaining server-side logic”	“New Team created successfully”	“New Team created successfully”	Pass
Tc-21	Check results on add new Team with Missing Fields	Category: “Back_end Development “ Description: “”	Error message “fields required”	Error message “fields required”	Pass
Tc-22	Check results on update Team info	Category: “Back_end Development “ Description: “ The Backend team”	Team updated successfully	Team updated successfully	pass
TC-23	Check results on delete Team		Team deleted successfully	Team deleted Successfully	Pass
Tc-24	Check results on add new STMe	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578	“New STMe created successfully”	“New STMe created successfully”	Pass
Tc-25	Check results on add new STMe but STMe already exist	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578	Error message “STMe already exist with this email”	Error message “STMe already exist with this email”	Pass
Tc-26	Check results on add new STMe with Missing Fields	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password:	Error message “fields required”	Error message “fields required”	Pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-27	Check results on add new STMe with invalid information	User Name: “yaser” phone_number:”0993479013” Email: yaser/gmailcom Password: 1231	Error message “invalid information”	Error message “invalid information”	pass
Tc-28	Check results on add STMe to team	Team:” back_end”	The STMe added Team successfully	The STMe added Team successfully	pass
Tc-29	Check results on update team for STMe	Team:” back_end”	The STMe Team updated successfully	The STMe Team updated successfully	pass
Tc-30	Check results on Change status for STMe		The system should Change STMe status	The system should Change STMe status	pass
Tc-31	Check results on show Ticket Details.		The system should display the Ticket Details.	The system should display the Ticket Details.	Pass
Tc-32	Check results on add comment on Ticket	Comment “this ticket passed”	Comment added successfully	Comment added successfully	Comment added successfully
Tc-33	Check results on see all comments on Ticket		The system should display the comments.	The system should display the comments.	Pass
Tc-34	Check results on Assign Ticket to Team Transaction by STM		The system should allow the user to drag and drop ticket	The system should allow the user to drag and drop ticket	pass
Tc-35	Check results on Assign to members or in progress or done Transaction by STM		The system should not allow the user to drag and drop ticket	The system should not allow the user to drag and drop ticket	pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-36	Check results on Assign Ticket to me Transaction by STMe in My Team Ticket interface		The system should allow the user to drag and drop ticket	The system should allow the user to drag and drop ticket	pass
Tc-37	Check results on from assign to member to In Progress or Done Transaction by STMe in My Team Ticket interface		The system should not allow the user to drag and drop ticket	The system should not allow the user to drag and drop ticket	pass
Tc-38	Check results on from assign to team to In Progress Transaction by STMe in Ticket interface		The system should not allow the user to drag and drop ticket	The system should not allow the user to drag and drop ticket	Pass
Tc-39	Check results on from assign to team to Done Transaction by STMe in Ticket interface		The system should not allow the user to drag and drop ticket	The system should not allow the user to drag and drop ticket	Pass
Tc-40	Check results on from In Progress to Done Transaction by STMe in Ticket interface		The system should allow the user to drag and drop ticket	The system should allow the user to drag and drop ticket	Pass
Tc-41	Check results on add new Ticket by Client	Title: “Printer Not Working” Description: “The office printer located on the third floor Not Working”	Ticket created successfully	Ticket created successfully	Pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-42	Check results on add new Ticket with Missing Fields By Client	Title: “” Description: “The office printer located on the third floor Not Working”	Error message “fields required”	Error message “fields required”	pass
Tc-43	Check results on update Ticket info By Client	Title: “Printer Not Working” Description: “The office printer located on the 2 floors Not Working”	Ticket updated successfully	Ticket updated successfully	Pass
Tc-44	Check results on update Ticket info Missing Fields By Client	Title: “Printer Not Working” Description: “”	Error message “fields required”	Error message “fields required”	pass
Tc-45	Check results on delete Ticket by Client		Ticket deleted successfully	Ticket deleted successfully	Pass
Tc-46	Check results on show Ticket Activity logs by STM and STMe		The system should display the Activity logs.	The system should display the Activity logs.	Pass
Tc-47	Check results on entering a valid email and password.	Email: yaser@gmail.com Password: 1234578	Log in successfully	Log in successfully	Pass
Tc-48	Check results on entering an invalid email, or password.	Email: yaser@gmail.com Password: 1234578	Error message “Invalid email or password”	Error message “Invalid email or password”	Pass
Tc-49	Check results when a user email or password are empty, and the “login” button is pressed.	Email: Password: 1234578	Error message “This field is required “	Error message “This field is required “	Pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-50	Check results on completing the Register form correctly, and select Register	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	Sing-up successfully	Sing-up successfully	pass
Tc-51	Check results on entering an existing email.	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 1234578 Company: “spu”	Error message “user already exist with this email”	Error message “user already exist with this email”	Pass
Tc-52	Check results when a user chose a weak password (less than 8 characters).	User Name: “yaser” phone_number:”0993479013” Email: yaser@gmail.com Password: 123457 Company: “spu”	Error message “Must contain at least 8 characters”	Error message “Must contain at least 8 characters”	Pass
Tc-53	Check results when Client try to Register with Missing Fields	User Name: “yaser” phone_number:”” Email : yaser@gmail.com Password : 123457 Company: “spu”	Error message “fields required”	Error message “fields required”	pass
Tc-54	Check results when Client try to Register and company not found	User Name: “yaser” phone_number :”” Email: yaser@gmail.com Password: 123457 Company: “spu”	Error message “Company not found “	Error message “Company not found “	pass

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
Tc-55	Check if Admin can assign a Default Workflow to a company	User Name: nuha Email: nuha@gmail.com Workflow: Default	Company created with Default Workflow assigned	Company created with Default Workflow assigned	pass
Tc-56	Check if Admin can define a Customized Workflow	User Name: nuha Email: nuha@gmail.com Workflow: Steps = Step 1, Step 2, Step 3	Company saved with customized workflow steps	Custom steps saved	Pass
Tc-57	Check if workflow steps appear on ticket board for customized company	User Name: nuha Email: nuha@gmail.com Workflow: Customized	Steps (e.g., Step 1 → Step 2 → Step 3) appear as columns	Steps appear on board	Pass
Tc-58	Move a ticket from Step 1 to Step 2 in Customized Workflow	User Name: nuha Email: nuha@gmail.com Ticket status moved Step 1 → Step 2	Ticket status updated to Step 2	Status updated	pass
Tc-59	Move a ticket through all stages of Default Workflow	User Name: nuha Email: nuha@gmail.com Workflow: Open → Assign → In Progress → Done	Ticket reaches final status "Done"	Status changed through all stages	pass
Tc-60	Reject invalid workflow transition (e.g., skip a step)	User Name: nuha Email: nuha@gmail.com Dragged Step 1 → Step 3 directly	System rejects the move and shows validation error	Validation error shown	pass

6. Last version of RTM

Table 43 Last version of RTM

Req_ID	Title	SRS Section	System Design	Analysis (Use Cases)	Detailed Design	Coding	Test Cases	Changed Requests No
RE-FR-UM-1.1	The system shall allow the admin to create a new user profile by entering necessary information	FR(Add_User)	interface1.1	Seq1.1	Class1 (UM)	Code1.1	Tc-01, Tc-02, Tc-03, Tc-04, Tc-05	
RE-FR-UM-1.2	The system shall allow the Admin to edit user information	FR(Delete_User)	interface1.2	Seq1.2		Code1.2	Tc-08	
RE-FR-UM-1.3	The system shall allow the Admin to delete a user	FR(Edit_User)	interface1.3	Seq1.3		Code1.3	Tc-06, Tc-07	
RE-FR-UM-1.4	The system shall automatically assign a default role to a newly created user based on the actor performing the creation							
RE-FR-UM-1.4.1	When the Admin creates a new user, the system shall assign the role Support Team Manager by default							
RE-FR-UM-1.4.2	When a user registers through the system, the system shall assign the role Client by default							
RE-FR-UM-1.4.3	When a Support Team Manager creates a new user, the system shall assign the role Support Team Member by defaul							
RE-FR-TEM-2.1	The system shall allow Support Team Manager (STM) to add team to support teams	FR(Add_Team)	interface2.1	Seq2.1	Class2(TEM)	Code2.1	Tc-20, Tc-21	
RE-FR-TEM-2.2	STM shall be able to delete team from the support teams	FR(Delete_Team)	interface2.2	Seq2.2		Code2.2	Tc-23	
RE-FR-TEM-2.3	The system shall allow STM to edit team details	FR(Edit_Team)	interface2.3	Seq2.3		Code2.3	Tc-22	
RE-FR-TEM-2.4	The system shall allow the STM to view a list of all support teams	FR(Show_Team)	interface2.4	Seq2.4		Code2.4		
RE-FR-TIM-3.1	The system shall allow Clients to create anew ticket, entering required details such as issue description	FR(Create_Ticket)	interface3.1	Seq3.1	Class3(TIM)	Code3.1	Tc-41, Tc-42	
RE-FR-TIM-3.2	The system shall provide an option for the STMs to assign tickets to specific team for resolution.	FR(Assign_Ticket_ToTeam)	interface3.2	Seq3.2		Code3.2	Tc-34, Tc-38, Tc-39,	

Req_ID	Title	SRS Section		Analysis (Use Cases)				Changed Requests No
RE-FR-TIM-3.3	The system shall employ an NLP-based approach to analyze the content of the ticket and automatically estimate its priority based on keywords, urgency, and sentiment analysis.		interface3.3		Class4(TIM)	Code3.3	Tc-31	
RE-FR-TIM-3.4	The system shall provide functionality to view ticket details, including priority, category, company name, status of ticket, client name, and ticket description	FR(View_Ticket_Details)	interface3.4	Seq3.3		Code3.4	Tc-45	
RE-FR-TIM-3.5	The system shall allow authorized users to delete a ticket if required, with a record of the deletion captured in the ticket log	FR(Delete_Ticket)	interface3.5	Seq3.4		Code3.5	Tc-43, Tc-44	
RE-FR-TIM-3.6	The system shall allow authorized users to update a ticket, with a record of the deletion captured in the ticket log	FR(Update_Ticket)	interface3.6	Seq3.5		Code3.6	Tc-40	
RE-FR-TIM-3.7	The system shall allow Support Team Managers and Support Team Members to change the status of a ticket (e.g., "In Progress," "Done")	FR(Change_Ticket_Status)		Seq3.6		Code3.7		
RE-FR-TIM-3.8	The system shall allow Support Team members to view all tickets that have been assigned to their team	FR(Show_assigned_Tickets)		Seq3.7		Code3.8		
RE-FR-TIM-3.9	The system shall allow Support Team members to resolve tickets that have been assigned to them	FR(Solve_Ticket)	interface4.1	Seq3.8		Code4.1	Tc-24, Tc-25, Tc-26, Tc-27	
RE-FR-MM-4.1	The system shall allow the Support Team Manager (STM) to add new members to the system.	FR(Add_Member)		Seq4.1		Code4.2	Tc-28, Tc-29	
RE-FR-MM-4.2	The system shall enable the STM to add members to a specific support team	FR(Add_Member_ToTeam)	interface4.3	Seq4.2		Code4.3	Tc-30	
RE-FR-MM-4.3	The STM shall be able to change the status of a team member, including activating or deactivating the member	FR(Change_Status_Of_TeamMember)	interface4.4	Seq4.3		Code4.4	Tc-35, Tc-36, Tc-37	

Req_ID	Title	SRS Section	interface6.3	Analysis (Use Cases)		Code6.3	Tc-17, Tc-18	Changed Requests No
RE-FR-MM-4.4	Support team members shall be able to assign a ticket to themselves.	FR(Assign_Ticket_to_Me)	interface5.1	Seq4.4		Codet5.1	Tc-09, Tc-10, Tc-11	
RE-FR-AIM-5.1	The system shall allow all users to edit their account information.	FR(Update_Account)	interface5.2	Seq5.1		Code5.2		
RE-FR-AIM-5.2	Users shall be able to delete their accounts.	FR(Delete_Account)	interface6.1	Seq5.2	Class5(CPM)	Code6.1	Tc-13, Tc-14, Tc-15, Tc-16	
RE-FR-CPM-6.1	The system shall allow the Admin to add a company	FR(Add_Company)	interface6.2	Seq6.1		Code6.2	Tc-19	
RE-FR-CPM-6.2	The system shall allow the Admin to delete a company	FR(Delete_Company)	interface6.4	Seq6.2		Code6.4	Tc-12	
RE-FR-CPM-6.3	The system shall allow the Admin to edit company details	FR(Update_Company_Info)	interface_7	Seq6.3	Class6(COM)	Code7.1	Tc-32	
RE-FR-CPM-6.4	The Admin shall be able to view a list of all companies	FR(Show_Company)		Seq6.4		Code7.2	Tc-33	
RE-FR-COM-7.1	The system shall allow STM, Support Team Member, and Client to add comments on ticket.	FR(Add_Comment)	interface 8	Seq7.1	Class7(Auth)	Code8	Tc-50, Tc-51, Tc-52, Tc-53, Tc-54	
RE-FR-COM-7.2	The system shall allow STM, Support Team Member, and Client to viewa list of comments associated with their tickets	FR(Show_Comment)	interface 9	Seq7.2	Class7(Auth)	Code9	Tc-47, Tc-48, Tc-49	
RE-FR-RM-8.1	The system shall allow new clients to register by providing necessary information	FR(Register)	interface 10	Seq8.1		Code10		
RE-FR-SU-9.1	The system shall provide a login page for all users, requiring valid credentials (email and password) to access the system	FR(Login)	interface 11	Seq9.1				
RE-FR-SU-9.2	The system shall validate user credentials against the database during login attempts and display appropriate error messages for incorrect username/email or password	FR(Login)	interface 12	Seq9.2				
RE-FR-SU-9.3	The system shall redirect users to their respective dashboards or homepages based on their roles after a successful login	FR(Login)	interface 13	Seq9.3				
RE-FR-SU-9.4	The system shall provide a logout option for users, terminating their session securely	FR(Login)	interface 14	Seq9.4				

Req_ID	Title	SRS Section		Analysis (Use Cases)				Changed Requests No
RE-FR-SU-9.5	The system shall automatically log out inactive users after a predefined period of inactivity for security purposes	FR(Login)		Seq9.5				
RE-FR-SU-9.6	The system shall implement security features for login, including password hashing using a secure algorithm	FR(Login)		Seq9.6				
RE-NF-SC-1.1	The system shall be scalable to support an increasing number of users and tickets without performance degradation.							
RE-NF-SC-1.2	The system architecture shall allow the addition of new servers or resources to handle increased workloads dynamically							
RE-NF-AV-2.2	The system shall maintain uptime to ensure continuous availability of ticket management and support features							
RE-NF-INT-3.1	The system shall be capable of integrating with third-party customer support platforms							
RE-NF-INT-3.2	The system shall ensure data compatibility with external systems							
RE-NF-SEC-4.1	The system shall enforce secure authentication, requiring users to log in using strong passwords							
RE-NF-SEC-4.2	The system shall encrypt all sensitive data (e.g., passwords, user information)							
RE-NF-EX-5.1	The system shall be extensible to support multiple languages, allowing for easy integration of new language modules in the future without significant architectural changes							
RE-NF-EX-5.2	The system shall be designed to allow for seamless integration of additional AI components							

Req_ID	Title	SRS Section		Analysis (Use Cases)				Changed Requests No
RE-F-WF-8.1	The system shall allow the Admin to choose between a Default Workflow or a Customized Workflow							
RE-F-WF-8.2	Admin can define a sequence of steps in Customized Workflow							
RE-F-WF-8.3	STM and Members can transition tickets based on company's workflow							
RE-F-WF-8.4	Workflow visually displayed in Kanban-style interface							
RE-F-WF-8.5	Workflow configuration saved and applied per company							
RE-F-CAT-9.1	System uses BERT-based ML to analyze ticket content							
RE-F-CAT-9.2	System auto-assigns a category based on analysis							
RE-F-CAT-9.3	Category appears in ticket metadata							
RE-F-CAT-9.4	Categorization data saved for analytics and model improvement							
RE-F-WF-8.1	The system shall allow the Admin to choose between a Default Workflow or a Customized Workflow							
RE-F-WF-8.2	Admin can define a sequence of steps in Customized Workflow							
RE-F-WF-8.3	STM and Members can transition tickets based on company's workflow							
RE-F-WF-8.4	Workflow visually displayed in Kanban-style interface							
RE-F-WF-8.5	Workflow configuration saved and applied per company							
RE-F-CAT-9.1	System uses BERT-based ML to analyze ticket content							

RE-F-CAT-9.2	System auto-assigns a category based on analysis							
RE-F-CAT-9.3	Category appears in ticket metadata							
RE-F-CAT-9.4	Categorization data saved for analytics and model improvement							

Chapter 8 Report Overview

1. Introduction

This chapter provides an overview of the report structure and explains the purpose of each chapter. The objective is to guide the reader through the logical flow of the report, ensuring clarity in understanding how different sections contribute to the overall study and system development.

2. Report structure and purposes

- **Chapter 1: Introduction**

  This chapter lays the foundation of the report by introducing the project, identifying the problem statement, and defining the objectives. It also provides an overview of the proposed system and explains how the report is organized.

- **Chapter 2: Fundamental Concepts and Literature Review**

  This chapter presents the theoretical background and existing research relevant to the project. It covers key fundamental concepts related to the system and a literature review analyzing similar systems, technologies, and methodologies that influenced the project.

- **Chapter 3: Project Management**

  This section outlines the project management aspects, including the project charter, scope of work (SOW) document, Gantt chart for project planning, and risk management strategies. It ensures that the project is well-structured and managed efficiently.

- **Chapter 4: System Analysis**

  This chapter focuses on the analysis phase of the system, starting with the project timeline, followed by the Software Requirements Specification (SRS) document. It also includes requirements modeling, initial test cases, and the initial Requirement Traceability Matrix (RTM), ensuring comprehensive requirement analysis.

- **Chapter 5: System Design**

  This section provides a detailed explanation of the system design, including system architecture and the detailed design of system

components. It serves as a blueprint for system implementation, ensuring a structured approach to development.

- **Chapter 6: Artificial Intelligence Design and Techniques**
  This chapter focuses on the artificial intelligence aspects of the system. It explains the AI models and techniques used for ticket categorization, prioritization, and team assignment, as well as the design of the RAG-based AI chatbot. The chapter covers knowledge base management, Chat Engine and Solution Engine design, and AI feature control mechanisms.
- **Chapter 7: Practical Implementation**
  This chapter presents the practical implementation of the system. It describes the development environment, tools and technologies used, system interfaces, and integration between services. It also includes testing activities, execution of test cases, and the final version of the Requirement Traceability Matrix (RTM), demonstrating alignment between requirements and implementation.
- **Chapter 8: Report Overview**
  This chapter provides an overall summary of the project, highlighting key outcomes, system capabilities, and achieved objectives. It also discusses limitations and possible future enhancements of the system.

3. Summary

This chapter serves as a guide to understanding the structure of the report and how each section contributes to the project's development and documentation. By following this organization, the report ensures a logical progression from problem identification to system implementation, providing a comprehensive and structured presentation of the project.

References

- Freshdesk Support Desk. *Admin Setup Guide.* Available at: https://albaz-supportdesk.freshdesk.com/a/admin/setup-guide (Accessed: [Oct /2024]).
- Zoho Desk. *User Roles and Permissions Setup.* Available at: https://desk.zoho.com/agent/albaz/albaz/setup#setup/users-control/roles

(Accessed: [Oct /2024]).

- HubSpot. *Contact Management System.* Available at: https://app-eu1.hubspot.com/contacts/145554935/record/0-5/8548936643 (Accessed: [Oct /2024]).
- Django REST Framework. *Simple JWT Documentation.* Available at: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/index.html (Accessed: [Oct /2024]).
- Django REST Framework. *Authentication Guide.* Available at: https://www.django-rest-framework.org/api-guide/authentication/ (Accessed: [Oct /2024]).
- Injector. *Python Dependency Injection Framework.* Available at: https://injector.readthedocs.io/en/latest/ (Accessed: [Oct /2024]).
- Padilla, J. *Django REST Framework JWT Documentation.* Available at: https://jpadilla.github.io/django-rest-framework-jwt/ (Accessed: [Oct /2024]).
- Django Documentation. *Many-to-One Relationships in Models.* Available at: https://docs.djangoproject.com/en/5.1/topics/db/models/#many-to-one-relationships (Accessed: [Nov/2024]).
- GeeksforGeeks. *State Design Pattern.* Available at: https://www.geeksforgeeks.org/state-design-pattern/ (Accessed: [Nov/2024]).
- SpacyTextBlob. *Sentiment Analysis with spaCy and TextBlob.* Available at: https://spacytextblob.netlify.app/ (Accessed: [Nov/2024]).
- *Hugging Face. BERT: Pre-trained contextual representations. Available at:* https://huggingface.co/transformers/model_doc/bert.html (Accessed: [may/2025]).

Appendices

RTM