IT Support Ticket System

Technical Documentation

Contents

1 Introduction

The IT Support Ticket System is a Java-based application designed to streamline the process of reporting, tracking, and resolving IT-related issues within an organization. This system enables employees to report technical problems through a user-friendly interface, while providing IT support staff with efficient tools to manage and resolve these issues.

The purpose of this document is to provide a technical overview of the system architecture, design decisions, and implementation details to aid in understanding and future development of the system.

2 Problem Statement

Organizations frequently encounter challenges in managing IT support requests effectively, leading to inefficiencies in problem resolution and reduced productivity. The specific challenges include:

- **Lack of Centralized Tracking:** Without a dedicated system, IT issues are often reported through various channels (email, phone, in-person), making them difficult to track and prioritize.
- **Poor Visibility:** Employees lack visibility into the status of their reported issues, leading to frustration and redundant follow-ups.
- **Inefficient Communication:** Back-and-forth communication between employees and IT staff is often unstructured and time-consuming.
- **No Audit Trail:** Without proper logging, it's challenging to analyze patterns in IT issues or evaluate support performance.
- **Inconsistent Priority Assignment:** Without a standardized system, critical issues might not receive appropriate priority, delaying resolution of business-critical problems.

These issues collectively result in extended resolution times, decreased employee satisfaction, and inefficient utilization of IT support resources.

3 Solution

The IT Support Ticket System addresses these challenges through a comprehensive ticketing solution that provides structure to the reporting, tracking, and resolution of IT issues. The system is designed with the following key features:

- **Ticket Creation:** Employees can create detailed tickets with title, description, priority, and category.
- **Status Tracking:** All tickets progress through defined statuses (New, In Progress, Resolved).
- **Role-Based Access:** Different capabilities for Employees and IT Support staff.
- **Audit Logging:** Comprehensive tracking of all ticket changes and interactions.
- **Search & Filter:** Capabilities to quickly find relevant tickets.

The system follows a three-tier architecture with a Java Swing client for the presentation layer, Spring Boot REST API for the application layer, and Oracle SQL database for data persistence.

4 System Design

4.1 Architecture Overview

The system follows a three-tier architecture:

- **Presentation Layer**: Java Swing client application
- **Application Layer**: Spring Boot REST API
- **Data Layer**: Oracle SQL database

This separation of concerns allows for independent development, flexibility to replace components, and better scalability.

4.2 Use Cases

The main use cases for the system include:

- **For Employees:**

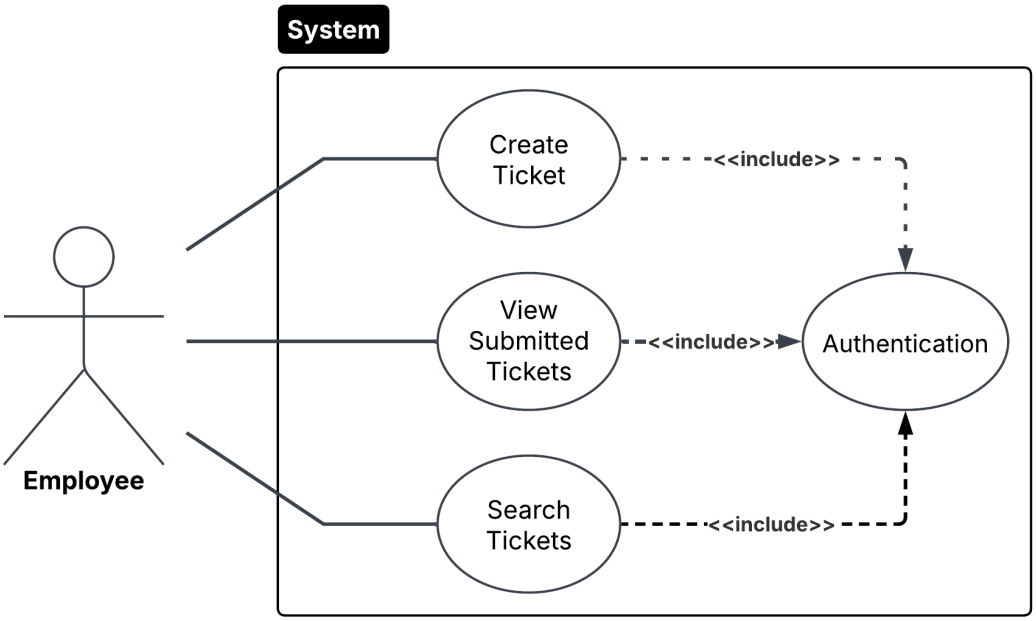

Figure 1: Employee use case diagram.

  - Create new tickets
  - View their own tickets
  - Search and filter tickets

- **For IT Support Staff:**

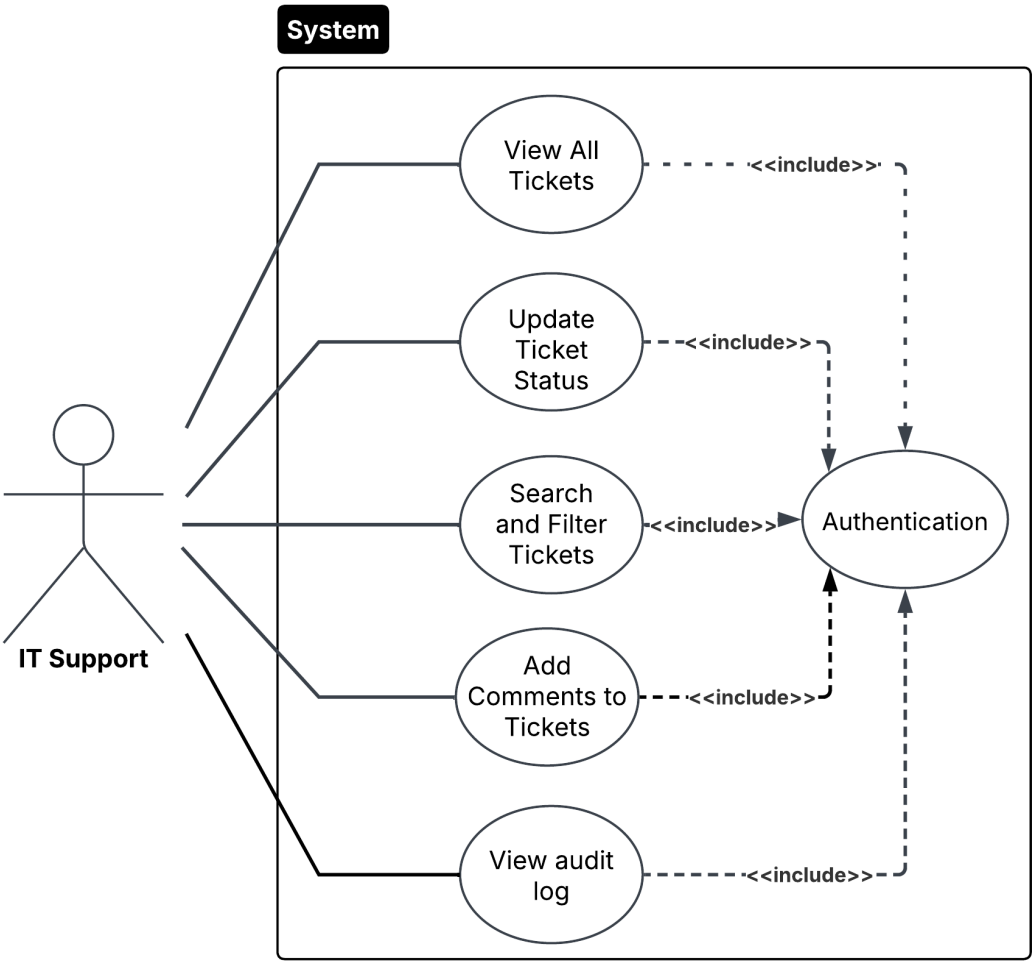

Figure 2: IT Support use case diagram.

  - View all tickets
  - Update ticket status
  - Add comments to tickets
  - View audit logs
  - Search and filter tickets

- **For the System:**
  - Create audit logs automatically
  - Set creation date on new tickets

4.3 Domain Model

The core entities in the system include:

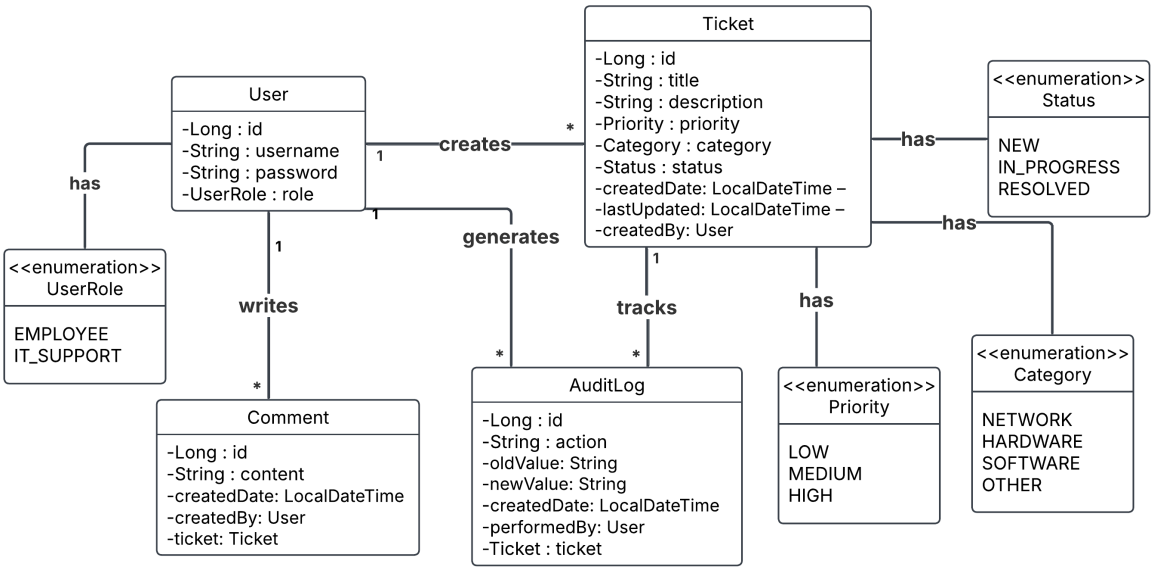

Figure 3: class diagram.

- **User:** Represents an employee or IT support staff member
  - id: Long
  - username: String
  - password: String (encrypted)
  - role: Role (EMPLOYEE or IT_SUPPORT)
- **Ticket:** Represents an IT support request
  - id: Long
  - title: String
  - description: String
  - priority: Priority (LOW, MEDIUM, HIGH)
  - category: Category (NETWORK, HARDWARE, SOFTWARE, OTHER)
  - status: Status (NEW, IN_PROGRESS, RESOLVED, CLOSED)
  - createdDate: LocalDateTime
  - lastUpdated: LocalDateTime
  - createdBy: User
- **TicketComment:** Represents a comment on a ticket
  - id: Long
  - content: String
  - createdDate: LocalDateTime
  - createdBy: User

  - ticket: Ticket

- **AuditLog:** Records changes to tickets
  - id: Long
  - action: String
  - oldValue: String
  - newValue: String
  - createdDate: LocalDateTime
  - performedBy: User
  - ticket: Ticket

4.4 Key Relationships

The main relationships between entities are:

- A User can create many Tickets (one-to-many)
- A Ticket can have many Comments (one-to-many)
- A Ticket can have many AuditLogs (one-to-many)
- A User can create many Comments (one-to-many)
- A User can perform many actions recorded in AuditLogs (one-to-many)

4.5 Workflow: Ticket Creation

The ticket creation process follows these steps:

1. The Employee fills in ticket details in the Swing UI
2. The Employee submits the ticket
3. The UI sends the ticket data to the API Client
4. The API Client sends a POST request to the REST API
5. The REST API forwards the request to the Ticket Service
6. The Ticket Service validates the data and the user
7. The Ticket Service creates a new Ticket entity with status NEW
8. The Ticket Service saves the ticket to the database
9. The success response is returned through the same chain back to the UI
10. The UI displays a success message to the Employee

4.6 Database Schema

The database consists of four main tables:

- **USERS:** Stores user information
- **TICKETS:** Stores ticket information
- **TICKET_COMMENTS:** Stores comments on tickets
- **AUDIT_LOGS:** Stores audit log entries

Foreign key relationships connect these tables to maintain data integrity.

5 Technologies Used

5.1 Backend

- **Java 17:** The primary programming language
- **Spring Boot 3.x:** Framework for building the REST API
- **Spring Security:** For authentication and authorization
- **Spring Data JPA:** For database access and ORM
- **Hibernate:** Object-relational mapping for database interactions
- **Lombok:** For reducing boilerplate code
- **Swagger/OpenAPI:** For API documentation

5.2 Frontend

- **Java Swing:** For building the desktop client UI
- **MigLayout:** For flexible UI layouts
- **RestTemplate:** For client-server communication

5.3 Database

- **Oracle SQL:** For data persistence
- **SQL:** For database queries and schema definition

5.4 Testing

- **JUnit 5:** For unit testing
- **Mockito:** For mocking dependencies in tests

5.5 DevOps

- **Maven:** For build automation
- **Docker:** For containerization and deployment
- **Docker Compose:** For orchestrating multi-container applications

6 Screenshots from the app

6.1 Login screen

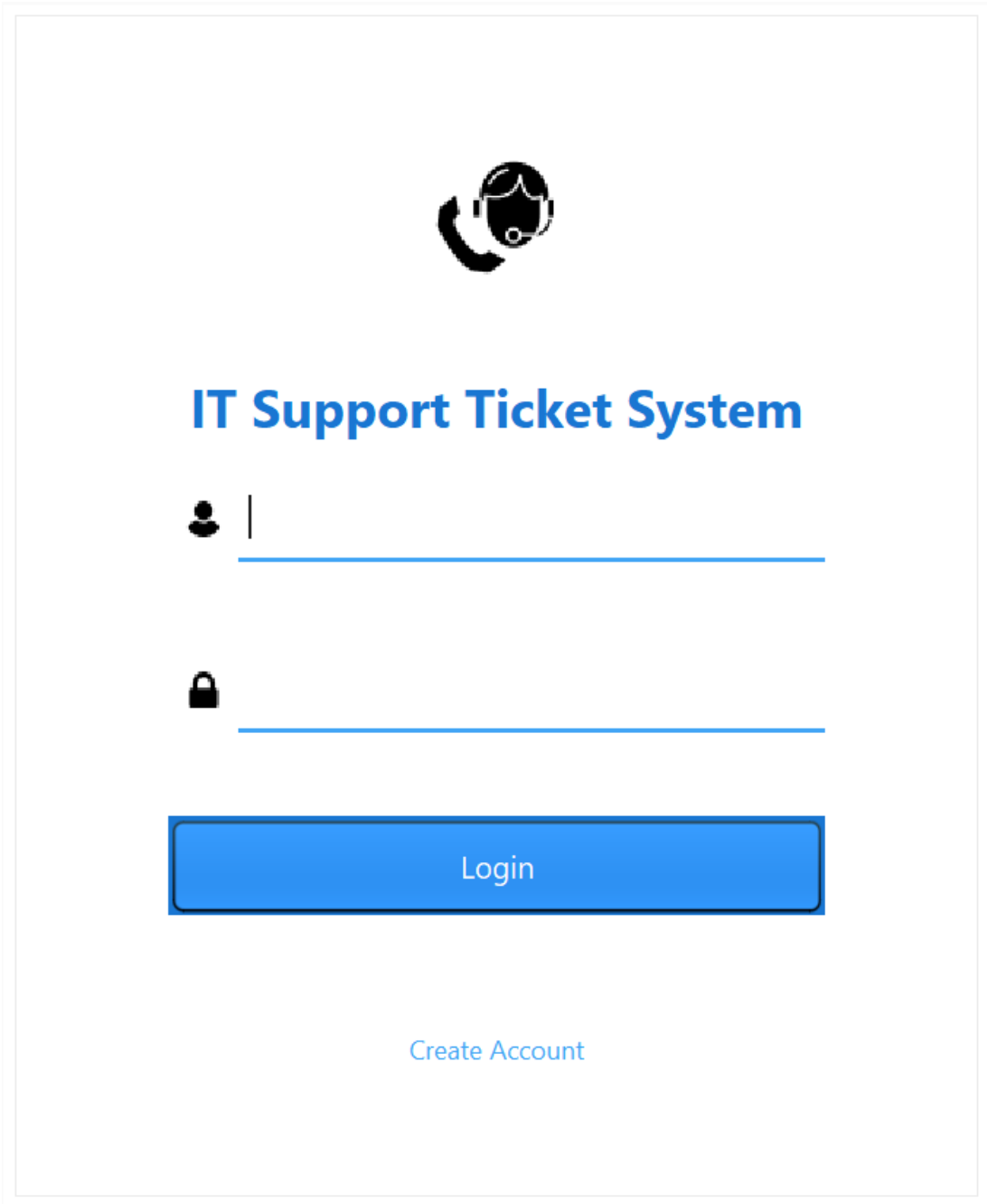

Figure 4: login screen.

6.2 Registration screen

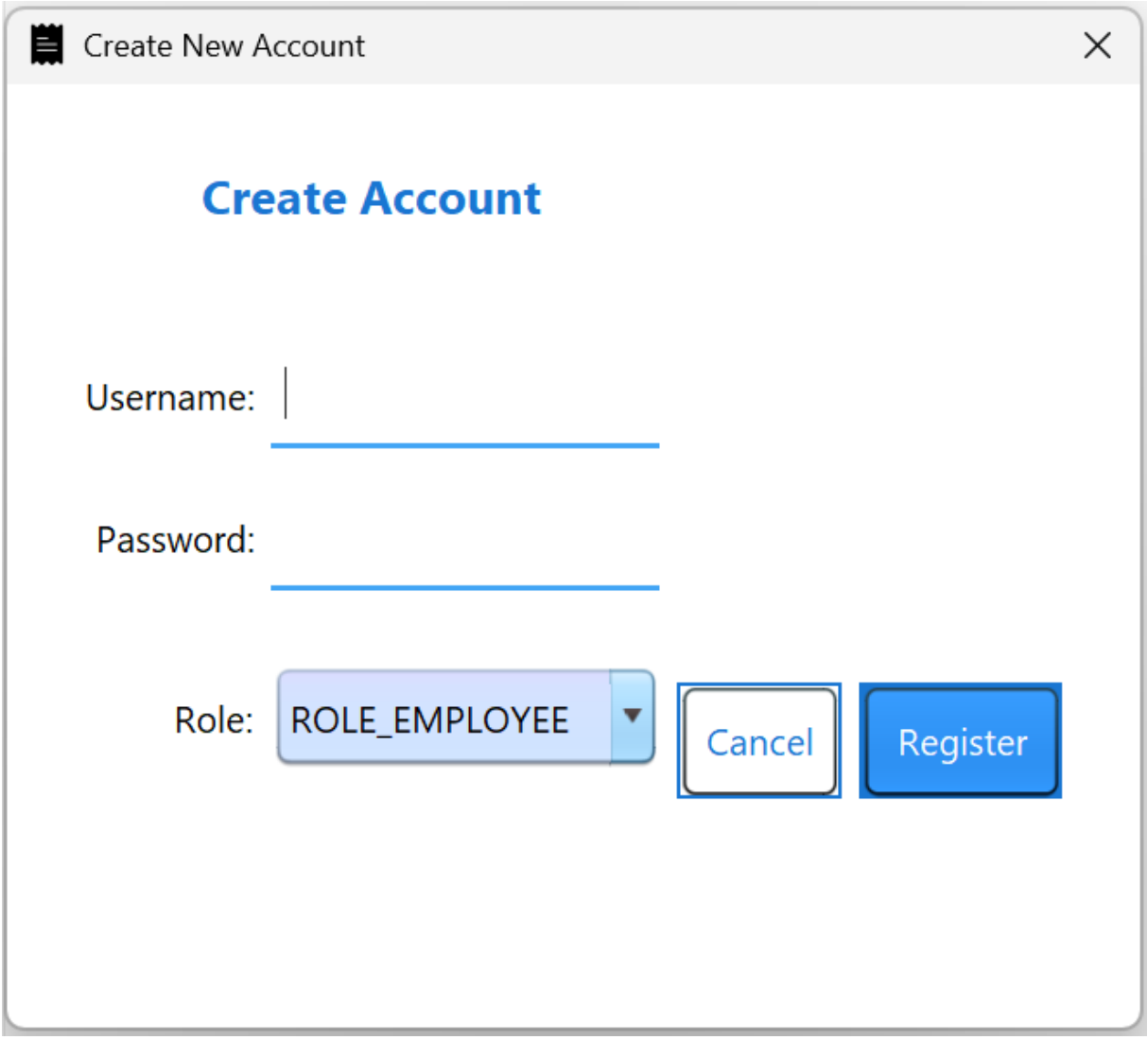

Figure 5: Registration screen.

6.3 Main screen

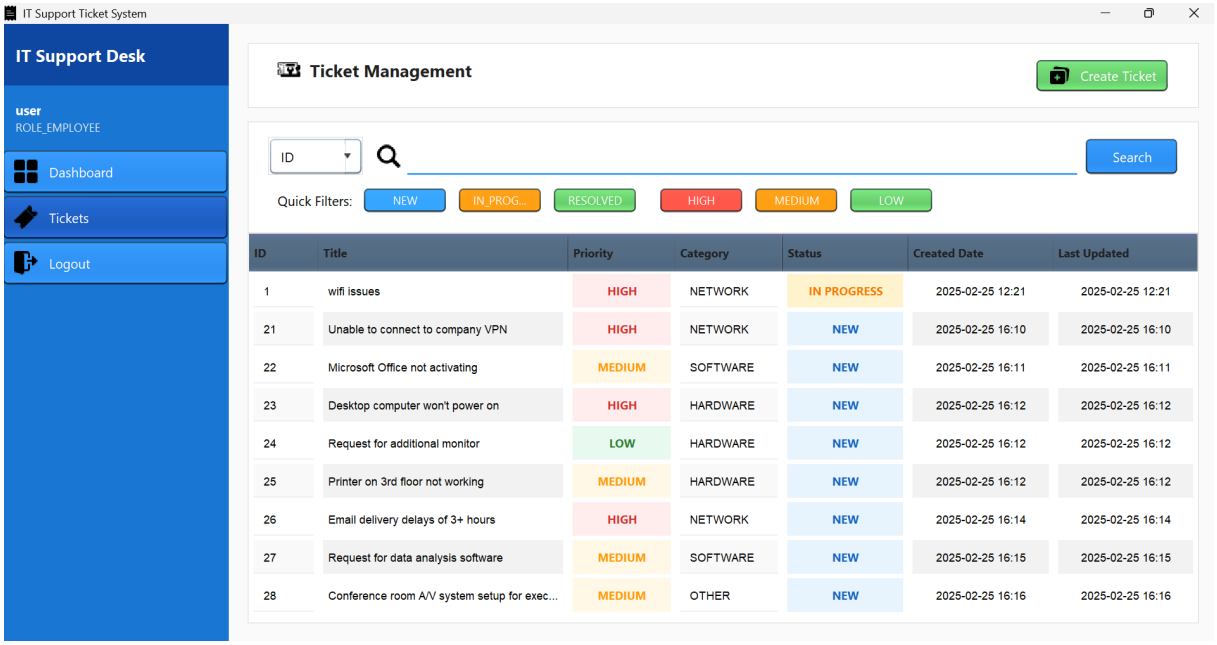

Figure 6: Main screen

6.4 Add ticket screen

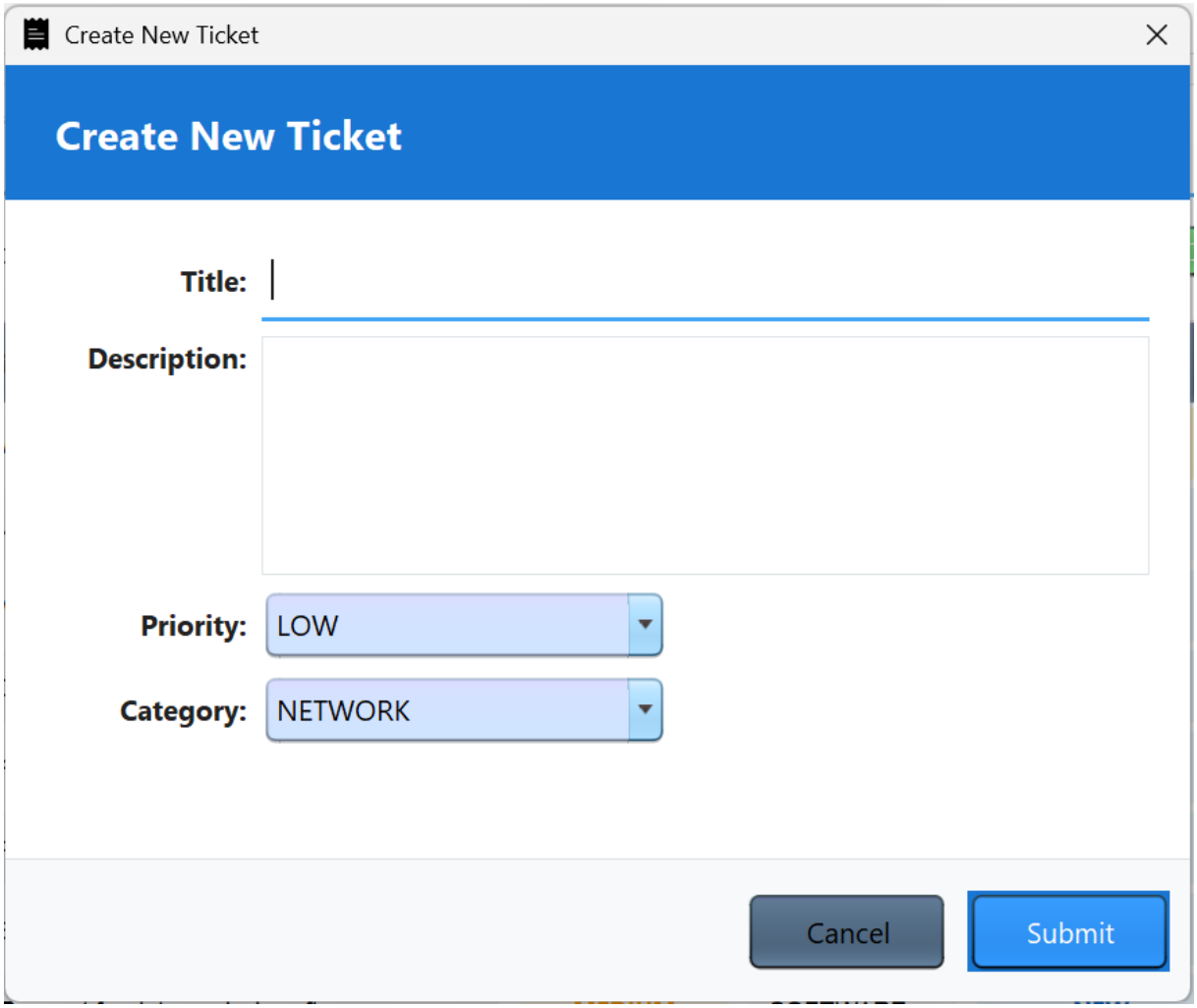

Figure 7: add new ticket screen.

6.5 Add comment window

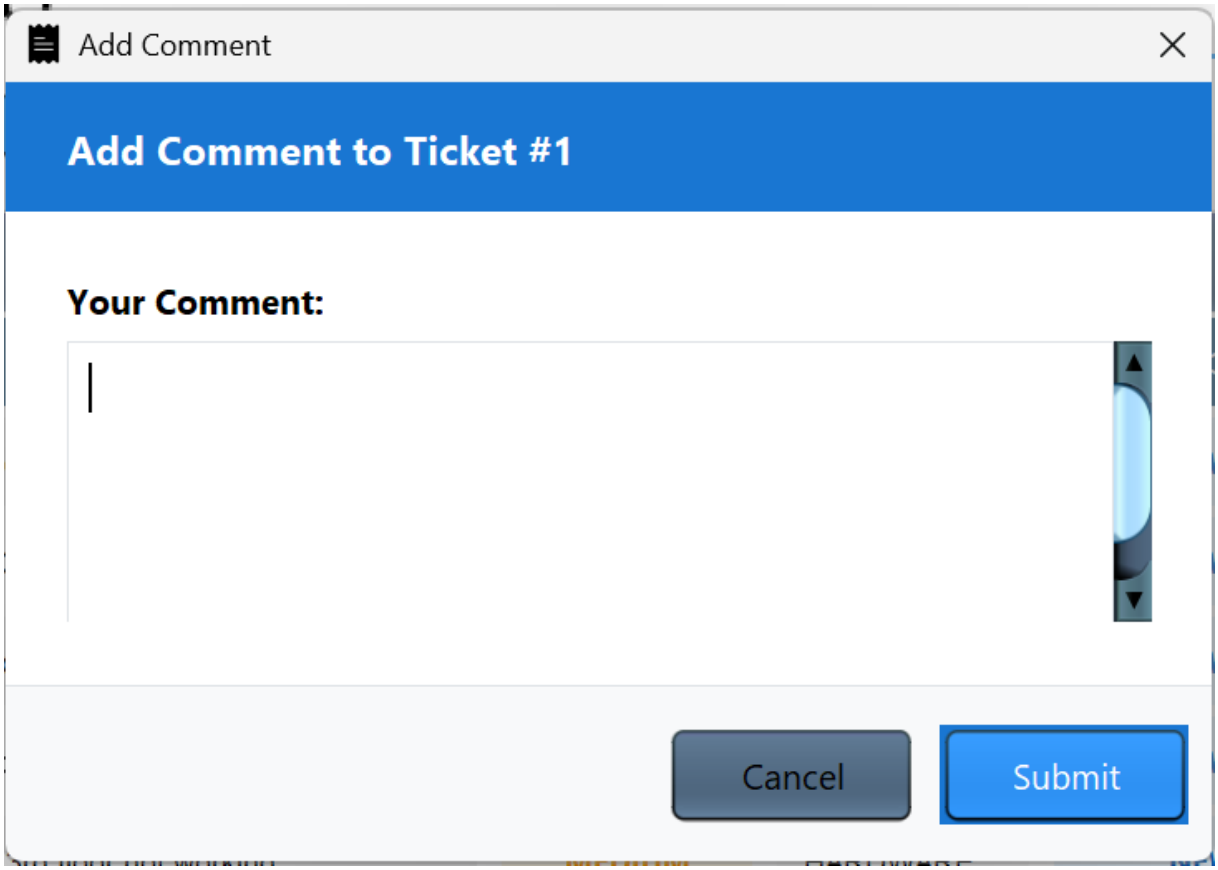

Figure 8: add comment window

6.6 AuditLog screen

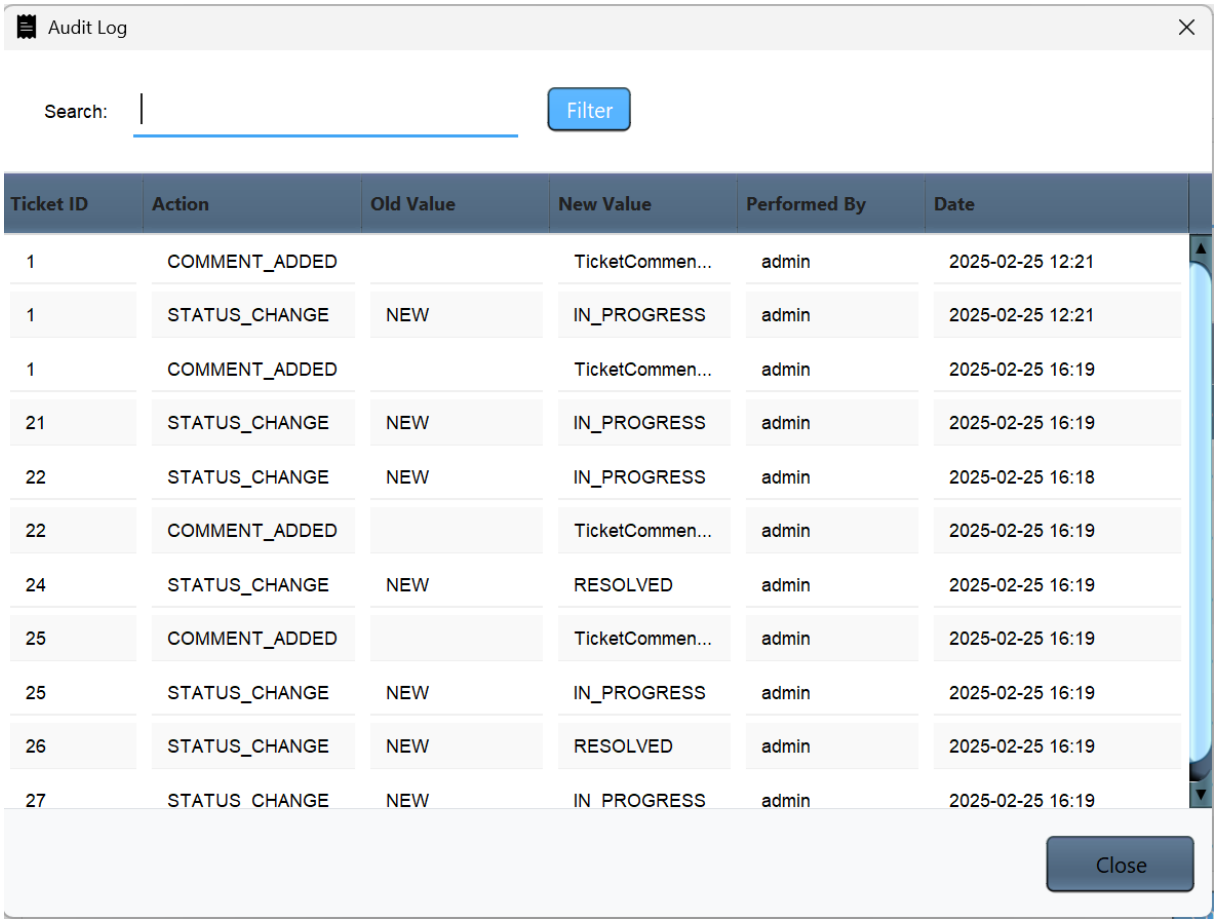

Figure 9: logs window

7 Conclusion

The IT Support Ticket System provides a comprehensive solution for managing IT support requests within an organization. By centralizing issue reporting and tracking, the system improves communication, increases visibility, and enhances the overall efficiency of the IT support process.

The system's modular architecture allows for easy maintenance and future enhancements. The use of industry-standard technologies ensures compatibility with existing systems and facilitates long-term support.