Software Requirements Specification (SRS) for Technical Support CRM Portal

Developers: Kumar Satyam, Adarsh Sen Singh, Kartik Singh
Timeline: 9 Weeks

Table of Contents

---

1. Introduction

1.1 Purpose

The **Technical Support CRM Portal** aims to streamline the process of issue resolution within technical companies by improving collaboration between teams, reducing response times, and ensuring efficient management of ticket lifecycles. The portal will centralize all support activities, from ticket creation to resolution, and offer tools for real-time communication, SLA enforcement, and team workload management.

1.2 Document Conventions

- **SRS**: Software Requirements Specification - outlines the functional and non-functional requirements of the system.
- **SLA**: Service Level Agreement - defines the expected response and resolution times for support tickets.
- **CRM**: Customer Relationship Management - a system to manage customer interactions and data.
- **UI**: User Interface - the visual components through which users interact with the system.
- **RBAC**: Role-Based Access Control - a security approach restricting system access based on user roles.

1.3 Intended Audience and Reading Suggestions

This document is intended for the following groups:

- **Software Development Team**: To understand technical requirements and implementation details.
- **Project Managers**: To monitor the project's scope, deliverables, and progress.
- **Support and Technical Teams**: To familiarize with system functionalities and workflows.
- **Stakeholders and Admins**: To grasp the broader business context and configuration options.

1.4 Product Scope

The CRM Portal will serve as a comprehensive tool for technical support management, featuring:

- **Role-based UIs**: Different interfaces tailored for Customers, Support Teams, Technical Teams, Team Leads, and Admins.
- **Dynamic Ticket Management**: Tickets can be created, assigned, updated, resolved, and closed, with custom statuses for flexibility.
- **Real-Time Collaboration**: Tools like internal team chats, live notifications, and audit logs to foster effective communication.

- **Scalability**: The system is designed to manage over 5,000 tickets per month and accommodate multiple teams without compromising performance.

1.5 References

- Official Documentation for **Node.js**, **Express.js**, and **MongoDB**
- **BullMQ** and **Redis** Documentation for task scheduling and caching
- **Socket.io** and **WebSocket** Documentation for real-time updates
- Industry Best Practices for **REST API** Design and **Security Protocols**

---

2. Overall Description

2.1 Product Perspective

The CRM Portal is a standalone application, inspired by platforms like Jira, but specifically tailored for technical support teams within companies. It is not a sub-component of any other system but can integrate with third-party tools such as GitHub and Slack for enhanced functionality. Deployment will utilize modern tools like Render, Docker, and Nginx for scalability and performance.

2.2 Product Functions

- **Ticket Lifecycle Management**: Allows users to create, assign, update, resolve, and close support tickets efficiently.
- **Role-Based Dashboards**: Customized views for different user roles to display relevant information and actions.
- **SLA Management**: Automated alerts for SLA breaches, and priority escalations based on ticket age and backlog.
- **Team Management**: Enables Admins and Team Leads to create teams, manage members, and configure system settings.
- **Real-Time Notifications**: Immediate updates on ticket status, SLA breaches, and internal team communications.

2.3 User Classes and Characteristics

- **Customer**: Users who submit issues, track their ticket status, and receive resolution updates.
- **Support Team**: Responsible for triaging tickets, setting priorities, and assigning tasks to the technical team.
- **Technical Team**: Focuses on resolving assigned tickets, adding detailed comments, and attaching relevant files.

- **Team Lead**: Manages team workloads, enforces SLAs, and oversees ticket assignments and resolutions.
- **Admin**: Has full access to system configurations, team management, and role-permission settings.

2.4 Operating Environment

- **Backend**: Node.js with Express.js for building scalable REST APIs.
- **Database**: MongoDB, hosted locally for data persistence and flexibility.
- **Caching & Queues**: Redis, using BullMQ for task scheduling and efficient workload distribution.
- **WebSockets**: Socket.io integrated with Redis for real-time updates and notifications.
- **Deployment**: Render for cloud deployment, Docker for containerization, PM2 for process management, and Nginx as a reverse proxy for load balancing and security.
- **Testing**: Tools like Jest, Supertest, Chai, Mocha, and @shelf/jest-mongodb to ensure robust testing of the system.

2.5 Design and Implementation Constraints

- **Local Database Hosting**: MongoDB will be hosted on the local server, ensuring data privacy and control.
- **Role-Permission Mapping**: Flexible access control through dynamic role-permission assignments.
- **Decoupled WebSockets**: Notifications will be managed separately to enhance performance and reduce system load.
- **Security Measures**: Implementation of rate limiting, CSRF protection, and encrypted data storage to safeguard against vulnerabilities.

2.6 User Documentation

- **User Manuals**: Step-by-step guides for Customers, Support Teams, and Admins to navigate the system.
- **Technical Documentation**: Detailed explanation of system architecture, APIs, and development guidelines for the technical team.
- **API Documentation**: Comprehensive documentation for third-party integrations and API usage.

2.7 Assumptions and Dependencies

- All users will have stable internet connectivity for accessing the system.
- Redis will be correctly configured for caching and queuing functionalities.
- The deployment environment will include proper security measures like firewalls to protect the system.

---

3. System Features

3.1 Ticket Management

- **Lifecycle Stages**: Tickets progress through stages: *Open → In Progress → Resolved → Closed*.
- **Sub-Tickets**: Allows for breaking down complex issues into smaller, manageable tasks.
- **Multi-Team Sharing**: Tickets can be shared across teams for collaborative problem-solving.
- **Reassignment**: Team Leads can reassign misrouted tickets to appropriate teams.

3.2 SLA Enforcement & Automation

- **Automated Alerts**: Notifications at 75% SLA expiry and upon breach to ensure timely responses.
- **Priority Escalation**: Automatic escalation based on ticket age and backlog volume to prioritize critical issues.

3.3 Internal Collaboration

- **Chronological Comments**: All updates are logged in chronological order for transparency and if something goes wrong, it can be reversed.
- **Audit Logs**: Detailed records of ticket modifications for accountability.
- **Real-Time Messaging**: Internal chat functionality for team discussions separate from customer-facing updates.

3.4 Customizable Workflows & Filters

- **Custom Statuses**: Admins can define additional ticket statuses beyond the default lifecycle.
- **Advanced Filters**: Filter tickets based on SLA breaches, unresolved blockers, shared tickets, and priority levels.

3.5 Notification System

- **Real-Time Alerts**: Instant notifications for new ticket assignments, comments, and SLA breaches.
- **In-App Messaging**: Direct messages within the CRM, including @mentions to notify specific users.

---

4. Nonfunctional Requirements

4.1 Performance Requirements

- **Dashboard Loading**: Dashboards should load in under 2 seconds even with over 100 concurrent users.
- **Ticket Handling**: The system should manage up to 5,000 tickets monthly without performance degradation.

4.2 Security Requirements

- **Role-Based Access Control (RBAC)**: Restricts access to specific features based on user roles.
- **Data Encryption**: AES-256 encryption for sensitive data storage.
- **Security Protocols**: CSRF protection, rate limiting, and multi-factor authentication for secure system access.

4.3 Usability Requirements

- **Intuitive Design**: User-friendly and responsive UI with dark mode support for accessibility.
- **Accessibility Compliance**: Interfaces must comply with WCAG standards to ensure usability for all users.

4.4 Scalability Requirements

- **Dynamic Team Addition**: The system should support the addition of new teams without performance issues.
- **Microservices Architecture**: The system design should allow for future expansion through microservices.

---

5. Development Phase

The development of the Technical Support CRM Portal will be carried out in structured phases, with each phase focusing on prioritized features to ensure a smooth and efficient rollout. This

approach allows for the most critical components to be developed and tested first, providing a foundation for iterative improvements based on user feedback.

5.1 Priority 1: Core Infrastructure & Basic Ticketing

- **Core Setup**:
  - Establish development environments, repositories, and CI/CD pipelines.
  - Set up Node.js with Express.js and local MongoDB for backend infrastructure.
  - Configure Redis for caching and queuing using BullMQ.
  - Deploy using Docker, PM2, and Nginx for process management and reverse proxy.
- **Authentication & Authorization**:
  - Implement secure user authentication with JWT and Role-Based Access Control (RBAC).
  - Develop user registration, login, and password recovery modules.
- **Basic Ticket Lifecycle**:
  - Enable ticket creation, viewing, and basic status transitions (Open, In Progress, Resolved, Closed).
  - Provide a basic user interface for customers and support teams to manage tickets.

5.2 Priority 2: Enhanced Ticket Management & SLA Automation

- **Advanced Ticket Management**:
  - Introduce sub-ticket creation for managing complex issues.
  - Implement multi-team ticket sharing and reassignment capabilities.
- **SLA Enforcement & Alerts**:
  - Integrate automated alerts at 75% SLA expiry and on breaches.
  - Develop priority escalation mechanisms based on ticket age and backlog.
- **Dashboard Enhancements**:
  - Build role-specific dashboards for Support Teams, Technical Teams, and Team Leads.
  - Integrate real-time updates using WebSockets (Socket.io).

5.3 Priority 3: Internal Collaboration & Custom Workflows

- **Collaboration Tools**:
  - Implement internal team messaging and @mentions within the CRM.
  - Develop chronological logging of ticket comments and audit trails.
- **Customizable Workflows**:
  - Allow admins to define custom ticket statuses and workflows.
  - Introduce advanced filtering options (e.g., SLA breaches, unresolved blockers).
- **Notification Enhancements**:

    - Improve notification systems to provide real-time alerts for ticket status changes and updates.

5.4 Priority 4: Additional Features & Integrations

- **Team Management**:
    - Develop tools for Team Leads to create, manage teams, and assign or reassign tickets.
    - Implement workload balancing to prevent team member overload.
- **Third-Party Integrations**:
    - Integrate with external platforms like GitHub, Slack, and calendar services (Google Calendar/Outlook).
    - Enable automated syncing of ticket deadlines with personal calendars.
- **Analytics and Reporting**:
    - Build advanced reporting modules and performance dashboards.
    - Implement audit logs and change history tracking for accountability.
- **Future Enhancements**:
    - Prepare the system for additional modules like public knowledge bases, automated response suggestions, and predictive analytics.

6. Other Requirements

6.1 Business Rules

- **Ticket Reassignment**: Only Team Leads and Admins can reassign tickets to prevent unauthorized changes.
- **SLA Enforcement**: Any breach in SLA will automatically escalate the issue to the next support level.
- **Real-Time Notifications**: Users must be notified immediately of any ticket status changes.
- **Immutable Audit Logs**: All logs must be stored securely and remain unaltered for at least one year.

Appendix A: To Be Determined List

1. Final SLA configurations for different ticket types.
2. Specific third-party integrations like GitHub and Slack are to be implemented.
3. Customizable dashboard components tailored to different user roles.

Appendix B: Models and Diagrams

B.1 Use Case Diagram

- **Actors**: Customer, Support Team, Technical Team, Team Lead, Admin
- **Use Cases**: Submit Ticket, Assign Ticket, Resolve Ticket, Reassign Ticket, Manage Teams, Configure System Settings

B.2 Class Diagram

- **Classes**: User, Ticket, Team, Notification, SLA, AuditLog
- **Relationships**: User-Role association, Ticket-Team assignment, SLA enforcement on Ticket

B.3 Sequence Diagram

- **Scenario**: Ticket Lifecycle from Submission to Closure
- **Steps**:
    1. The customer submits a ticket
    2. Support Team assigns ticket to Technical Team
    3. The technical team resolves the ticket
    4. Admin reviews and closes ticket

B.4 Data Flow Diagram

- **Processes**: Ticket Creation, SLA Monitoring, Notification Dispatch, Team Management
- **Data Stores**: MongoDB (Tickets, Users, Teams), Redis (Cache, Queues)

B.5 Entity-Relationship Diagram

- **Entities**: User, Team, Ticket, Notification, SLA
- **Relationships**: Users belong to Teams, Tickets are assigned to Teams, Notifications are linked to Users, and Tickets