Tech Ticketing System - Requirements Document

Project Overview

A web-based ticketing system for managing IT support requests within an organization. This system allows users to submit, track, and manage technical support tickets.

Functional Requirements

1. User Management

- **User Registration**: New users can create accounts with email verification
- **User Authentication**: Secure login/logout functionality
- **User Roles**: Three user types - Admin, Agent, and End User
- **Profile Management**: Users can update their profile information
- **Password Management**: Password reset functionality via email

2. Ticket Management

- **Ticket Creation**: End users can create new support tickets with:
  - Title and description
  - Priority level (Low, Medium, High, Critical)
  - Category (Hardware, Software, Network, Account, Other)
  - File attachments (optional)
- **Ticket Assignment**: Agents can be assigned to tickets manually or automatically
- **Ticket Status Tracking**: Tickets progress through statuses:
  - Open → In Progress → Resolved → Closed
- **Ticket Updates**: Users can add comments and updates to existing tickets
- **Ticket Search**: Search tickets by ID, title, status, or assignee

3. Dashboard and Reporting

- **User Dashboard**: Personalized view showing user's tickets and their status
- **Agent Dashboard**: View assigned tickets and workload
- **Admin Dashboard**: System overview with ticket statistics
- **Basic Reports**: Generate reports on ticket volume, resolution times, and common issues

4. Notification System

- **Email Notifications**: Send notifications for:
  - New ticket creation
  - Ticket assignment
  - Status changes
  - New comments
- **In-App Notifications**: Real-time notifications within the application

5. Knowledge Base (Optional Enhancement)

- **Article Management**: Create and manage help articles
- **Search Functionality**: Search knowledge base articles
- **Article Categories**: Organize articles by category

Non-Functional Requirements

1. Performance Requirements

- **Response Time**: Web pages should load within 3 seconds under normal load
- **Concurrent Users**: System should support up to 100 concurrent users
- **Database Performance**: Database queries should execute within 1 second
- **File Upload**: Support file attachments up to 10MB

2. Security Requirements

- **Authentication**: Secure user authentication using ASP.NET Core Identity
- **Authorization**: Role-based access control (RBAC)
- **Data Protection**: Sensitive data encrypted in transit and at rest
- **Input Validation**: All user inputs validated and sanitized
- **Session Management**: Secure session handling with appropriate timeouts
- **HTTPS**: All communication over encrypted connections

3. Reliability Requirements

- **Uptime**: System availability of 99% during business hours
- **Data Backup**: Daily automated backups of all system data
- **Error Handling**: Graceful error handling with user-friendly messages
- **Data Integrity**: Ensure data consistency and prevent corruption

4. Scalability Requirements

- **Horizontal Scaling**: Architecture should support adding more servers
- **Database Scaling**: Database should handle growing data volume
- **Load Balancing**: Support for load balancing across multiple instances

5. Usability Requirements

- **User Interface**: Clean, intuitive web interface
- **Responsive Design**: Mobile-friendly responsive layout
- **Accessibility**: WCAG 2.1 AA compliance for accessibility
- **Browser Support**: Support for modern browsers (Chrome, Firefox, Safari, Edge)

6. Compatibility Requirements

- **Operating System**: Cross-platform compatibility (Windows, Linux, macOS)
- **Database**: Microsoft SQL Server or PostgreSQL
- **Framework**: .NET 8.0 or later
- **Containerization**: Docker support for easy deployment

7. Maintenance Requirements

- **Code Quality**: Clean, documented, and maintainable code
- **Logging**: Comprehensive application logging using structured logging
- **Monitoring**: Basic health checks and monitoring endpoints
- **Documentation**: Technical documentation and user guides

Technical Architecture Requirements

1. Backend Technology Stack

- **Framework**: ASP.NET Core 9.0 Web API
- **Database**: Entity Framework Core with SQL Server/Postgres
- **Authentication**: ASP.NET Core Identity
- **Logging**: Built-in logging (ILogger)
- **API Documentation**: Swagger/OpenAPI/Scalar

2. Frontend Technology Stack

- **Framework**: Blazor .NET 9
- **CSS Framework**: Bootstrap 5
- **JavaScript**: Vanilla JavaScript or jQuery

3. Development Requirements

- **Version Control**: Git with feature branch workflow
- **Code Style**: C# coding standards and conventions
- **Testing**: Unit tests with xUnit framework

Learning Objectives

C# and ASP.NET Core

- Mediator & CQRS
- Dependency injection
- Middleware pipeline
- API development
- Authentication and authorization

Entity Framework Core

- Code-first approach
- Database migrations
- LINQ queries
- Relationships and navigation properties
- Data seeding

Git

- Repository management
- Branch management
- Merge conflicts resolution
- Pull request workflow
- Git best practices

Docker

- Container creation
- Docker Compose for multi-container applications
- Environment configuration
- Container orchestration basics

Success Criteria

- All functional requirements implemented and tested
- System meets performance benchmarks
- Security requirements validated
- Successful deployment using Docker
- Code quality standards maintained
- Documentation completed
- Interns demonstrate proficiency in all target technologies