Project Problem Statement: Customer Support Ticketing System

🎯 Goal

The **Customer Support Ticketing System (CSTS)** aims to provide a simple yet realistic platform for managing customer support requests raised by users and handled by support agents under an admin's supervision.

🔍 Project Overview

Every organization offering products or services needs a support system to address user issues or feedback.
The **CSTS** allows:

- Customers to raise support tickets.
- Agents to manage and respond to tickets.
- Admins to oversee ticket activity and assign tasks.

The application will be developed using the **.NET 8 Full Stack architecture**, combining:

- **ASP.NET Core Web API** (Backend)
- **Entity Framework Core** (ORM)
- **MS SQL Server** (Database)
- **React** (Frontend)
- **JWT Authentication** (Security)
- **NUnit** (Testing)

System Users / Roles

Role	Description
Admin	Manages users, assigns tickets to support agents, and monitors the ticket lifecycle.
Support Agent	Works on tickets assigned by the admin, updates status, and provides resolutions.
Customer/ User	Raises support tickets, views ticket progress, and closes tickets once resolved.

Core Modules

1. User Management Module

- Handles registration and authentication for all users.
- Includes roles: Admin, Agent, Customer.
- Implements JWT-based authentication and role-based authorization.
- Admin can create or deactivate users.

Key APIs

- `POST /api/auth/register`
- `POST /api/auth/login`
- `GET /api/users` (Admin only)
- `PUT /api/users/{id}/status`

2. Ticket Management Module

- Customers create tickets with details like issue type, description, and priority.
- Admin assigns tickets to agents.
- Agents update ticket status (In Progress / Resolved).
- Customers can close tickets once they confirm resolution.

Key APIs

- `POST /api/tickets` → Create ticket
- `GET /api/tickets` → List tickets
- `GET /api/tickets/{id}` → View ticket details
- `PUT /api/tickets/{id}` → Update ticket status
- `DELETE /api/tickets/{id}` → Delete ticket (Admin only)

3. Ticket Assignment & Workflow

- Admin views all open tickets.
- Admin assigns tickets to available agents.
- Agents receive assigned tickets and begin work.
- Tickets go through stages:
    - **New → Assigned → In Progress → Resolved → Closed**
- Each stage update is tracked with timestamps.

4. Comment / Discussion Module

- Each ticket allows threaded discussions between Customer ↔ Agent ↔ Admin.
- Stores comments linked to ticket ID and user ID.
- Enhances collaboration and audit trail.

Key APIs

- `POST /api/comments`
- `GET /api/comments/{ticketId}`

5. Dashboard & Reports (Optional for Demo)

- Admin dashboard showing:
    - Total Tickets
    - Open / In-Progress / Closed counts

    - Agent workload
- Graphical representation (using React chart library).

Database Design (Entities Overview)

Entity	Key Fields	Relationships
User	UserId, Name, Email, PasswordHash, Role, IsActive	1–M with Ticket, 1–M with Comment
Ticket	TicketId, Title, Description, Priority, Status, CreatedBy, AssignedTo, CreatedDate	M–1 with User (CreatedBy, AssignedTo)
Comment	CommentId, TicketId, UserId, Message, CreatedDate	M–1 with Ticket, M–1 with User

Use Cases

Use Case ID	Use Case Title	Primary Actor	Description
UC01	Register and Login	User	User registers and logs in to access the system.
UC02	Raise Support Ticket	Customer	Customer creates a new ticket with issue details.
UC03	Assign Ticket	Admin	Admin assigns a ticket to an available agent.
UC04	Update Ticket Status	Agent	Agent updates status as progress occurs.
UC05	Add Comment	Agent/Customer	Adds discussion/comment under a ticket.
UC06	Close Ticket	Customer	Marks a ticket as resolved and closed.
UC07	View Ticket Dashboard	Admin	Displays ticket distribution and workload summary.

Technology Scope

Layer	Technology / Tool	Purpose
Backend	ASP.NET Core 8 Web API	Business logic, REST endpoints
ORM	Entity Framework Core 8	DB access & migrations
Database	SQL Server	Data persistence
Frontend	React 18 + Vite	UI for user interactions
Auth	JWT Token	Secure API communication
Testing	NUnit	Unit testing controllers & services
Tools	Visual Studio 2022, VS Code, Postman, Swagger	Development & API testing

Sample Workflow

1. **Customer** registers and logs in.
2. Creates a **Ticket** → "Unable to reset password."
3. **Admin** reviews new tickets → assigns to Agent "Alex."
4. **Agent Alex** updates status to "In Progress," adds a comment with progress note.
5. Once resolved, **Agent** marks ticket "Resolved."
6. **Customer** verifies and closes the ticket.
7. **Admin Dashboard** shows all resolved tickets and agent performance.