# **Mini Project: Insurance Claim Request System**

## **Problem Statement**

Insurance agencies often receive claim requests through fragmented channels, leading to inefficiencies and missed communication. Your task is to build a backend system for a **Claim Request Portal** that enables registered agents to:

* Submit, view, and update insurance claim requests
* Automatically track claim statuses
* Secure the platform using **ASP.NET Core Identity** and **role-based authentication**

The system will use **ASP.NET Core MVC**, **EF Core** for database operations, and **REST APIs** to handle interactions. Authentication and role management will be implemented using **ASP.NET Core Identity** with authentication and middleware.

## **Functional Requirements**

### **Feature 1: Authentication & Authorization (Using ASP.NET Core Identity)**

* Register and login functionality for agents using ASP.NET Core Identity
* Authentication and session handling
* **Role-based access:**
  + **Agent:** Can create claims and view/update only their own claims
  + **Admin:** Can view, update, and delete all claims across all agents
* Middleware to log all API access attempts and block unauthenticated requests

### **Feature 2: Claim Management**

* Submit a new claim with details (type, description, policy number, claim amount, status)
* View submitted claims
* Update claim status (e.g., New, In Review, Approved, Rejected)
* Delete a claim entry (soft delete optional)

### **Feature 3: Admin Dashboard (Additional Feature)**

* View claim statistics: total claims, approved claims, rejected claims, pending claims
* Ability to filter claims by date range, status, or agent
* Export claim data as CSV/Excel

## **Deliverables**

### **UML Diagrams**

* ER Diagram: Agents, Claims
* Class Diagram: Claim, Agent, ClaimService, AuthService
* Sequence Diagram: Agent Login → Submit Claim

### **Code Implementation**

* ASP.NET Core Web API with MVC pattern
* Entity Framework Core for ORM
* **Controllers:**
  + ClaimController (CRUD)
  + AuthController (Login/Register/Logout)
* **Services:**
  + ClaimService for business logic
  + AuthService for authentication
* ASP.NET Core Identity for user management
* Auth for session handling
* Middleware for request logging and access control
* Dependency Injection for services and repository layers

## Database Schema

Table: Claims

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| ClaimId | INT (PK) | Unique ID for each claim |
| PolicyNumber | VARCHAR(50) | Associated policy number |
| ClaimType | VARCHAR(50) | Type of claim (Health, Auto, etc.) |
| Description | TEXT | Details of the claim |
| ClaimAmount | DECIMAL(10,2) | Amount being claimed |
| Status | VARCHAR(20) | New, In Review, Approved, Rejected |
| CreatedAt | DATETIME | Timestamp when claim was created |
| UpdatedAt | DATETIME | Timestamp when claim was last updated |
| AgentId | INT (FK) | Foreign key referencing Agent/User (UserId from ASPNETUsers Table) |

Table: AspNetUsers (default from Identity)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id | INT (PK) | Unique user ID |
| UserName | VARCHAR(100) | Login username |
| NormalizedUserName | VARCHAR(100) | Normalized username |
| Email | VARCHAR(100) | User email address |
| PasswordHash | VARCHAR(200) | Hashed password |
| SecurityStamp | VARCHAR(200) | Security token for auth |
| ConcurrencyStamp | VARCHAR(200) | Concurrency token |
| PhoneNumber | VARCHAR(20) | Optional phone number |
| Role | VARCHAR(50) | User role (Agent, Admin) |

### **API Endpoints**

#### **AuthController**

* POST /api/auth/register → Register a new agent account
* POST /api/auth/login → Authenticate user and create session
* POST /api/auth/logout → Logout and clear session

#### **ClaimController**

* POST /api/claims → Submit a new claim (**Agent only**)
* GET /api/claims →
  + Agent: View only their own claims
  + Admin: View all claims
* GET /api/claims/{id} → View details of a single claim
* PUT /api/claims/{id} → Update claim status/details
  + Agent: Can update description/amount while status = “New”
  + Admin: Can update status (In Review, Approved, Rejected)
* DELETE /api/claims/{id} → Delete a claim
  + Admin: Can delete any claim
  + Agent: Can delete their own claim only if status = “New”

#### **(Bonus) Admin Dashboard**

* GET /api/admin/claims/summary → Get claim statistics (total, approved, rejected, pending)
* GET /api/admin/claims/export → Export claims data as CSV/Excel

## **Workflows**

* **Login & Register:** Secure authentication using ASP.NET Core Identity
* **Claim Lifecycle:** Agent logs in → submits a claim → Admin reviews and updates status → Agent views updated claim status
* **Ownership Rule:** Agents can only view/edit their own claims; Admins have global access
* **Middleware logging:** Tracks every request with timestamp and endpoint name