# InvMIS (Inventory Management System) - Complete Rebuild Guide

Includes tasks, file locations, and full code snippets. Database: InventoryDB

## Contents

## Task #01 - Solution & Project Structure

```
Create solution 'InvMIS' and 4 projects:
- InvMIS.API (ASP.NET Core Web API)
- InvMIS.Domain (Class Library) - Entities
- InvMIS.Application (Class Library) - Interfaces & Services
- InvMIS.Infrastructure (Class Library) - DbContext & Repositories

Folder layout (recommended):
InvMIS/
  InvMIS.API/
    Controllers/
    Program.cs
    appsettings.json
  InvMIS.Application/
    Interfaces/
    Services/
    DTOs/
    Validation/
  InvMIS.Domain/
    Entities/
  InvMIS.Infrastructure/
    Data/
    Repositories/
    Migrations/
```

## Task #02 - EF Core & DbContext

Install packages in InvMIS.Infrastructure:
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL

Create file: InvMIS.Infrastructure/Data/InvMISDbContext.cs
Code:

```
using InvMIS.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace InvMIS.Infrastructure.Data
{
    public class InvMISDbContext : DbContext
    {
        public InvMISDbContext(DbContextOptions<InvMISDbContext> options)
            : base(options)
        {
        }

        public DbSet<Product> Products { get; set; } = null!;
        public DbSet<User> Users { get; set; } = null!;
        public DbSet<Category> Categories { get; set; } = null!;
        public DbSet<Supplier> Suppliers { get; set; } = null!;
        public DbSet<Stock> Stocks { get; set; } = null!;

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Product>()
                .HasOne(p => p.Category)
                .WithMany(c => c.Products)
                .HasForeignKey(p => p.CategoryId)
                .OnDelete(DeleteBehavior.Restrict);

            modelBuilder.Entity<Product>()
                .HasOne(p => p.Supplier)
```

```
                .WithMany(s => s.Products)
                .HasForeignKey(p => p.SupplierId)
                .OnDelete(DeleteBehavior.Restrict);

            modelBuilder.Entity<Stock>()
                .HasOne(s => s.Product)
                .WithMany()
                .HasForeignKey(s => s.ProductId)
                .OnDelete(DeleteBehavior.Cascade);
        }
    }
}
```

## Task #03 - Domain Entities (Code)

Place these files under InvMIS.Domain/Entities/

## Product.cs

```
namespace InvMIS.Domain.Entities
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; } = null!;
        public string SKU { get; set; } = null!;
        public decimal Price { get; set; }
        public int Quantity { get; set; }
        public string? Description { get; set; }

        // Relations
        public int? CategoryId { get; set; }
        public Category? Category { get; set; }

        public int? SupplierId { get; set; }
        public Supplier? Supplier { get; set; }
    }
}
```

## Category.cs

```
namespace InvMIS.Domain.Entities
{
    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; } = null!;
        public string? Description { get; set; }
        public ICollection<Product>? Products { get; set; }
    }
}
```

## Supplier.cs

```
namespace InvMIS.Domain.Entities
{
    public class Supplier
    {
        public int Id { get; set; }
        public string Name { get; set; } = null!;
        public string? ContactInfo { get; set; }
        public ICollection<Product>? Products { get; set; }
    }
}
```

## Stock.cs

```
namespace InvMIS.Domain.Entities
{
    public class Stock
    {
        public int Id { get; set; }
        public int ProductId { get; set; }
        public int Quantity { get; set; }
        public DateTime LastUpdated { get; set; } = DateTime.UtcNow;
        public Product? Product { get; set; }
    }
```

```
}
```

## User.cs

```
namespace InvMIS.Domain.Entities
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; } = null!;
        public string PasswordHash { get; set; } = null!;
        public string Role { get; set; } = "User";
        public string? FullName { get; set; }
        public string? Email { get; set; }
    }
}
```

## Task #04 - Repository Pattern (Interface & Implementation)

IRepository<T> in InvMIS.Application/Interfaces/IRepository.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace InvMIS.Application.Interfaces
{
    public interface IRepository<T> where T : class
    {
        Task<IEnumerable<T>> GetAllAsync();
        Task<T?> GetByIdAsync(int id);
        Task AddAsync(T entity);
        Task UpdateAsync(T entity);
        Task DeleteAsync(int id);
    }
}
```

Repository<T> implementation in InvMIS.Infrastructure/Repositories/Repository.cs

```
using InvMIS.Application.Interfaces;
using InvMIS.Infrastructure.Data;
using Microsoft.EntityFrameworkCore;

namespace InvMIS.Infrastructure.Repositories
{
    public class Repository<T> : IRepository<T> where T : class
    {
        private readonly InvMISDbContext _context;
        private readonly DbSet<T> _dbSet;

        public Repository(InvMISDbContext context)
        {
            _context = context;
            _dbSet = context.Set<T>();
        }

        public async Task<IEnumerable<T>> GetAllAsync() => await _dbSet.ToListAsync();

        public async Task<T?> GetByIdAsync(int id) => await _dbSet.FindAsync(id);

        public async Task AddAsync(T entity)
        {
            await _dbSet.AddAsync(entity);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(T entity)
        {
            _dbSet.Update(entity);
            await _context.SaveChangesAsync();
        }

        public async Task DeleteAsync(int id)
        {
            var entity = await _dbSet.FindAsync(id);
            if (entity != null)
            {
                _dbSet.Remove(entity);
                await _context.SaveChangesAsync();
            }
        }
    }
}
```

# Task #05 - Services & Interfaces (Code)

IProductService and ProductService

```
using InvMIS.Domain.Entities;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace InvMIS.Application.Interfaces
{
    public interface IProductService
    {
        Task<IEnumerable<Product>> GetAllProductsAsync();
        Task<Product?> GetProductByIdAsync(int id);
        Task<Product> AddProductAsync(Product product);
        Task<Product> UpdateProductAsync(Product product);
        Task<bool> DeleteProductAsync(int id);
    }
}
```

ProductService implementation (InvMIS.Application/Services/ProductService.cs)

```
using InvMIS.Application.Interfaces;
using InvMIS.Domain.Entities;
using InvMIS.Infrastructure.Repositories;

namespace InvMIS.Application.Services
{
    public class ProductService : IProductService
    {
        private readonly Repository<Product> _productRepository;

        public ProductService(Repository<Product> productRepository)
        {
            _productRepository = productRepository;
        }

        public async Task<IEnumerable<Product>> GetAllProductsAsync()
        {
            return await _productRepository.GetAllAsync();
        }

        public async Task<Product?> GetProductByIdAsync(int id)
        {
            return await _productRepository.GetByIdAsync(id);
        }

        public async Task<Product> AddProductAsync(Product product)
        {
            if (string.IsNullOrWhiteSpace(product.Name))
                throw new System.Exception("Product Name is required.");

            await _productRepository.AddAsync(product);
            return product;
        }

        public async Task<Product> UpdateProductAsync(Product product)
        {
            if (product.Id <= 0)
                throw new System.Exception("Invalid Product Id.");

            await _productRepository.UpdateAsync(product);
            return product;
        }

        public async Task<bool> DeleteProductAsync(int id)
        {
            await _productRepository.DeleteAsync(id);
            return true;
        }
```

```
    }
}
```

# Task #06 - API Controllers (ProductController example)

InvMIS.API/Controllers/ProductController.cs

```
using InvMIS.Application.Interfaces;
using InvMIS.Domain.Entities;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace InvMIS.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class ProductController : ControllerBase
    {
        private readonly IProductService _productService;

        public ProductController(IProductService productService)
        {
            _productService = productService;
        }

        [HttpGet]
        public async Task<IActionResult> GetAll()
        {
            var products = await _productService.GetAllProductsAsync();
            return Ok(products);
        }

        [HttpGet("{id}")]
        public async Task<IActionResult> GetById(int id)
        {
            var product = await _productService.GetProductByIdAsync(id);
            if (product == null) return NotFound();
            return Ok(product);
        }

        [HttpPost]
        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Create([FromBody] Product product)
        {
            var created = await _productService.AddProductAsync(product);
            return CreatedAtAction(nameof(GetById), new { id = created.Id }, created);
        }

        [HttpPut("{id}")]
        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Update(int id, [FromBody] Product product)
        {
            if (id != product.Id) return BadRequest();
            var updated = await _productService.UpdateProductAsync(product);
            return Ok(updated);
        }

        [HttpDelete("{id}")]
        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Delete(int id)
        {
            await _productService.DeleteProductAsync(id);
            return NoContent();
        }
    }
}
```

## Task #07 - Swagger Setup

```
Install package in InvMIS.API:
dotnet add package Swashbuckle.AspNetCore

In Program.cs add:
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

And inside app build:
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```

## Task #08 - appsettings.json (Database + Jwt + Serilog)

Place in InvMIS.API/appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },

  "AllowedHosts": "*",

  "ConnectionStrings": {
    "DefaultConnection": "Host=msasaroar-msasaroar.g.aivencloud.com;Port=26272;Database=InventoryDB;Userr
  },

  "Jwt": {
    "Key": "YourSuperSecretKeyHere123456",
    "Issuer": "InventoryDB",
    "Audience": "InvMISAPIUsers",
    "ExpireMinutes": 60
  },

  "Serilog": {
    "MinimumLevel": {
      "Default": "Information",
      "Override": {
        "Microsoft": "Warning",
        "System": "Warning"
      }
    },
    "WriteTo": [
      { "Name": "Console" },
      {
        "Name": "File",
        "Args": {
          "path": "Logs/log-.txt",
          "rollingInterval": "Day"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithMachineName", "WithThreadId" ]
  }
}
```

# Task #09 - Authentication (JWT) + AuthController

Install packages in InvMIS.API:

```
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
```

AuthController (InvMIS.API/Controllers/AuthController.cs):

```
using InvMIS.Application.Interfaces;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace InvMIS.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly IUserService _userService;
        private readonly IConfiguration _configuration;

        public AuthController(IUserService userService, IConfiguration configuration)
        {
            _userService = userService;
            _configuration = configuration;
        }

        [HttpPost("register")]
        public async Task<IActionResult> Register(string username, string password)
        {
            var user = await _userService.RegisterAsync(username, password);
            return Ok(new { user.Id, user.Username, user.Role });
        }

        [HttpPost("login")]
        public async Task<IActionResult> Login(string username, string password)
        {
            var user = await _userService.LoginAsync(username, password);
            if (user == null) return Unauthorized("Invalid credentials");

            var jwtSettings = _configuration.GetSection("Jwt");
            var key = Encoding.UTF8.GetBytes(jwtSettings["Key"]!);

            var token = new JwtSecurityToken(
                issuer: jwtSettings["Issuer"],
                audience: jwtSettings["Audience"],
                claims: new[]
                {
                    new Claim(ClaimTypes.Name, user.Username),
                    new Claim(ClaimTypes.Role, user.Role)
                },
                expires: DateTime.UtcNow.AddMinutes(double.Parse(jwtSettings["ExpireMinutes"]!)),
                signingCredentials: new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorit
            );

            return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });
        }
    }
}
```

## Task #10 - Global Exception Middleware

InvMIS.API/Middleware/ExceptionMiddleware.cs

```
using InvMIS.Application.Common;
using System.Net;
using System.Text.Json;

namespace InvMIS.API.Middleware
{
    public class ExceptionMiddleware
    {
        private readonly RequestDelegate _next;
        private readonly ILogger<ExceptionMiddleware> _logger;

        public ExceptionMiddleware(RequestDelegate next, ILogger<ExceptionMiddleware> logger)
        {
            _next = next;
            _logger = logger;
        }

        public async Task InvokeAsync(HttpContext context)
        {
            try
            {
                await _next(context);
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Unhandled exception occurred.");

                context.Response.ContentType = "application/json";
                context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

                var response = new ErrorResponse
                {
                    StatusCode = context.Response.StatusCode,
                    Message = "An unexpected error occurred.",
                    Details = ex.Message
                };

                var options = new JsonSerializerOptions { PropertyNamingPolicy = JsonNamingPolicy.CamelCa

                await context.Response.WriteAsync(JsonSerializer.Serialize(response, options));
            }
        }
    }
}
```

## Task #11 - DbSeeder (Admin User Seed)

InvMIS.Infrastructure/Data/DbSeeder.cs

```
using InvMIS.Domain.Entities;
using Microsoft.EntityFrameworkCore;

namespace InvMIS.Infrastructure.Data
{
    public static class DbSeeder
    {
        public static async Task SeedAdminAsync(InvMISDbContext context)
        {
            if (!await context.Users.AnyAsync(u => u.Role == "Admin"))
            {
                context.Users.Add(new User
                {
                    Username = "admin",
                    PasswordHash = Convert.ToBase64String(
                        System.Security.Cryptography.SHA256.HashData(System.Text.Encoding.UTF8.GetBytes('
                    ),
                    Role = "Admin"
                });

                await context.SaveChangesAsync();
            }
        }
    }
}
```

## Task #12 - Program.cs (Final)

InvMIS.API/Program.cs (final consolidated)

```
using System.Text;
using InvMIS.Application.Interfaces;
using InvMIS.Application.Services;
using InvMIS.Domain.Entities;
using InvMIS.Infrastructure.Data;
using InvMIS.Infrastructure.Repositories;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using InvMIS.API.Middleware;

var builder = WebApplication.CreateBuilder(args);

// PostgreSQL connection
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<InvMISDbContext>(options =>
    options.UseNpgsql(connectionString));

// Repositories (generic)
builder.Services.AddScoped(typeof(Repository<>)); // registers generic repository

// Services
builder.Services.AddScoped<IProductService, ProductService>();
builder.Services.AddScoped<ICategoryService, CategoryService>();
builder.Services.AddScoped<ISupplierService, SupplierService>();
builder.Services.AddScoped<IStockService, StockService>();
builder.Services.AddScoped<IUserService, UserService>();

builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "InvMIS API", Version = "v1" });
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Name = "Authorization",
        Type = SecuritySchemeType.ApiKey,
        Scheme = "Bearer",
        BearerFormat = "JWT",
        In = ParameterLocation.Header,
        Description = "Enter JWT token as: Bearer {your token}"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            Array.Empty<string>()
        }
    });
});

// JWT Authentication
var jwtSettings = builder.Configuration.GetSection("Jwt");
var key = Encoding.UTF8.GetBytes(jwtSettings["Key"]!);

builder.Services.AddAuthentication(options =>
```

```
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = jwtSettings["Issuer"],
        ValidAudience = jwtSettings["Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(key)
    };
});

var app = builder.Build();

// Seed admin
using (var scope = app.Services.CreateScope())
{
    var context = scope.ServiceProvider.GetRequiredService<InvMISDbContext>();
    DbSeeder.SeedAdminAsync(context).Wait();
}

// Exception middleware
app.UseMiddleware<ExceptionMiddleware>();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

## Task #13-15 - Category, Supplier, Stock Services & Controllers

Interfaces and Services follow the same pattern as Product; Controllers similar to ProductController with Role-based rest

`SupplierController.cs and StockController.cs (see earlier in guide).`

# Final Testing Guide (Detailed)

```
1) Build solution:
   dotnet build

2) Create and apply migrations (from solution root):
   cd InvMIS.Infrastructure
   dotnet ef migrations add InitialCreate --startup-project ../InvMIS.API --project .
   dotnet ef database update --startup-project ../InvMIS.API --project .

3) Run API:
   cd ../InvMIS.API
   dotnet run

4) Open Swagger: https://localhost:5001/swagger
   - Use /api/Auth/register or /api/Auth/login to create/login users.
   - Default seeded admin: username: admin, password: Admin@123

5) Authorize in Swagger with Bearer {token} then test protected endpoints:
   - POST /api/Product (Admin only)
   - PUT /api/Product/{id} (Admin only)
   - DELETE /api/Product/{id} (Admin only)

6) Verify database tables (Products, Categories, Suppliers, Stocks, Users) exist in InventoryDB.
```

## Project Reference Map

```
InvMIS.Domain -> (no references)
InvMIS.Application -> InvMIS.Domain
InvMIS.Infrastructure -> InvMIS.Domain, InvMIS.Application
InvMIS.API -> InvMIS.Application, InvMIS.Infrastructure, InvMIS.Domain
```