Project Title: Secure Note-Taking API with JWT Authentication

---

1. Introduction

- Description: A secure Web API that allows users to register, log in, and manage their personal notes.
- Purpose: Ensure that only authenticated users can access their own notes using JWT-based authentication.

---

2. Tools and Technologies

- ASP.NET Core Web API
- C#
- Entity Framework Core (InMemory/SQL Server)
- JWT (JSON Web Token) for authentication
- Swagger/Postman for testing

---

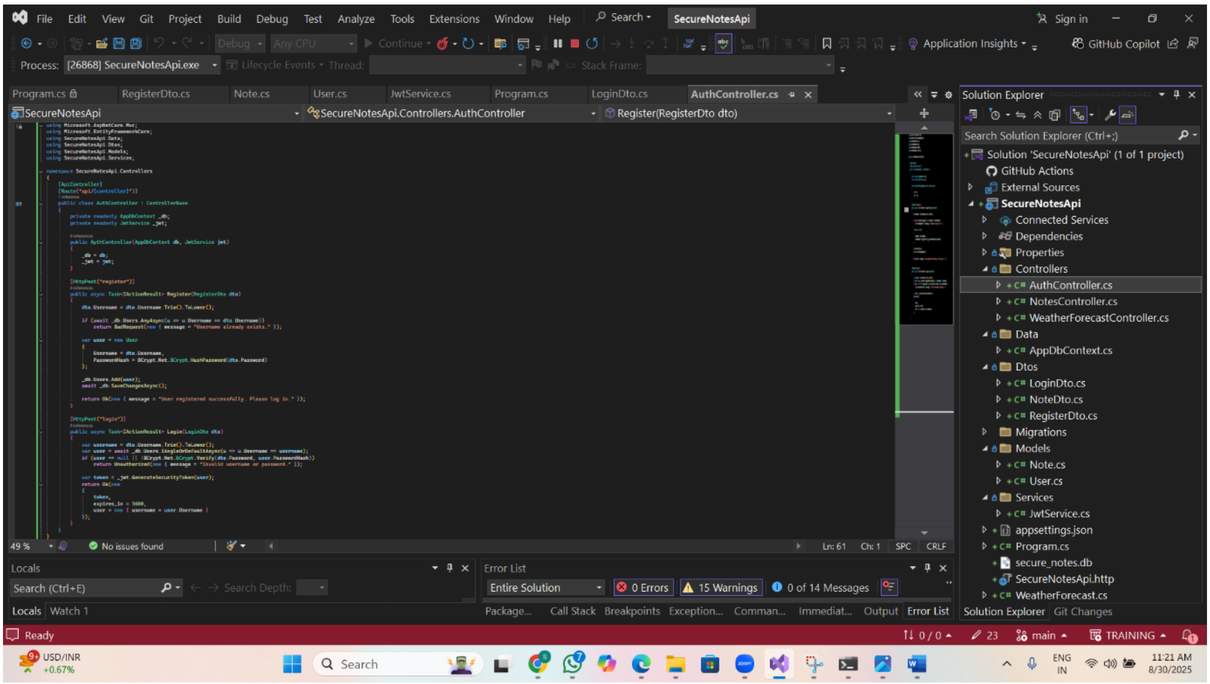
3. Project Structure

- Controllers:
  - AuthController → handles registration & login
  - NotesController → handles add, retrieve, update, delete notes
- Models: User, Note, RegisterModel, LoginModel
- Services: Authentication & JWT Token generation
- Program.cs / Startup.cs → configures authentication & middleware

---

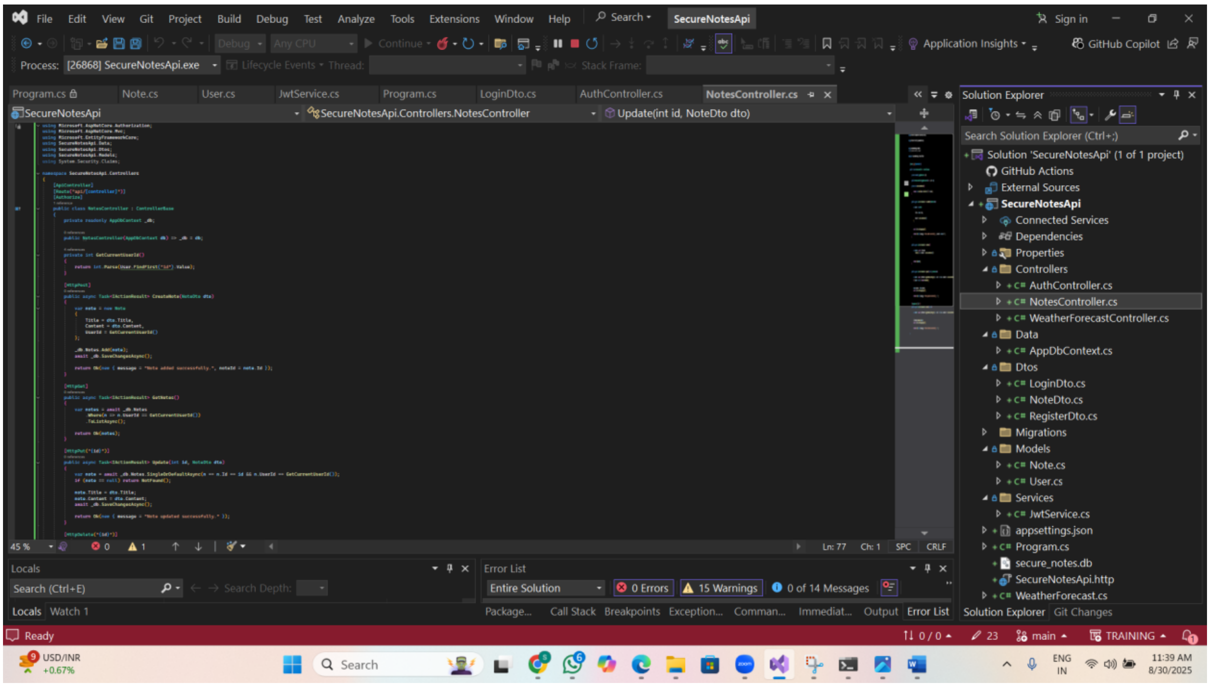
Screenshot of Code:

AuthController :



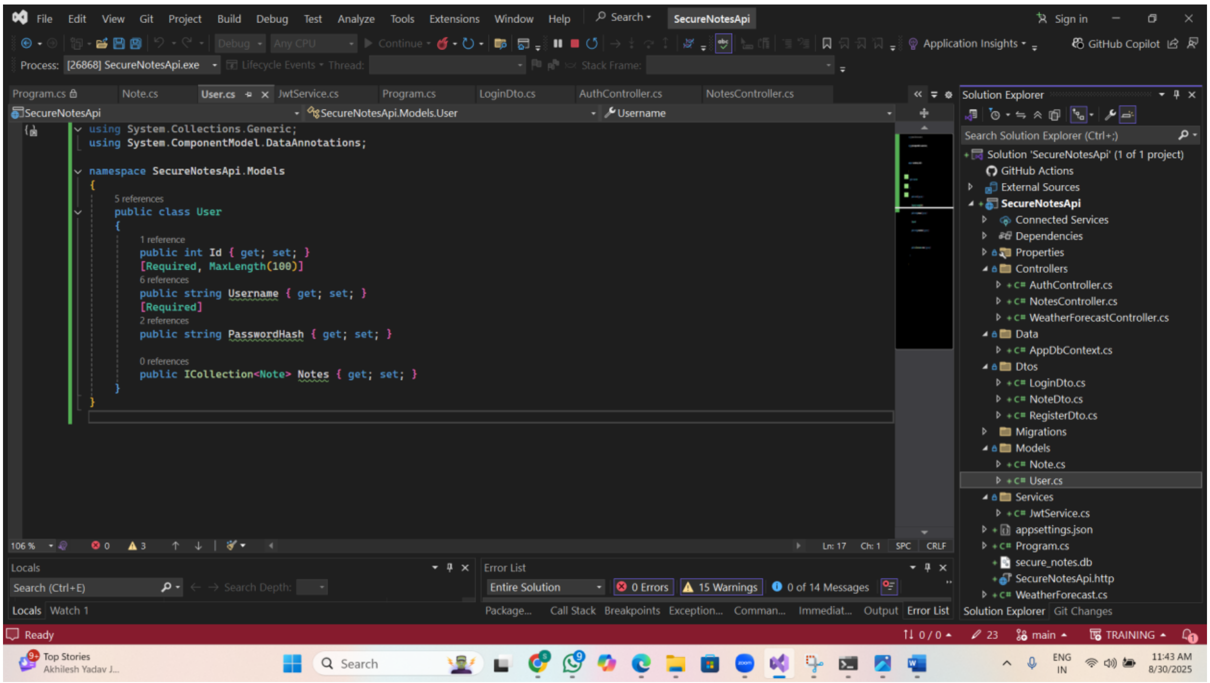
```csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SecureNotesApi.Data;
using SecureNotesApi.Dtos;
using SecureNotesApi.Models;
using SecureNotesApi.Services;

namespace SecureNotesApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthController : ControllerBase
    {
        private readonly AppDbContext _db;
        private readonly JwtService _jwt;

        public AuthController(AppDbContext db, JwtService jwt)
        {
            _db = db;
            _jwt = jwt;
        }

        [HttpPost("register")]
        public async Task<IActionResult> Register(RegisterDto dto)
        {
            dto.Username = dto.Username.Trim().ToLower();

            if (await _db.Users.AnyAsync(u => u.Username == dto.Username))
                return BadRequest(new { message = "Username already exists." });

            var user = new User
            {
                Username = dto.Username,
                PasswordHash = BCrypt.Net.BCrypt.HashPassword(dto.Password)
            };

            _db.Users.Add(user);
            await _db.SaveChangesAsync();

            return Ok(new { message = "User registered successfully. Please log in." });
        }

        [HttpPost("login")]
        public async Task<IActionResult> Login(LoginDto dto)
        {
            var username = dto.Username.Trim().ToLower();
            var user = await _db.Users.SingleOrDefaultAsync(u => u.Username == username);
            if (user == null || !BCrypt.Net.BCrypt.Verify(dto.Password, user.PasswordHash))
                return Unauthorized(new { message = "Invalid username or password." });

            var token = _jwt.GenerateSecurityToken(user);
            return Ok(new
            {
                token,
                expires_in = 3600,
                user = new { username = user.Username }
            });
        }
    }
}
```

NotesController:



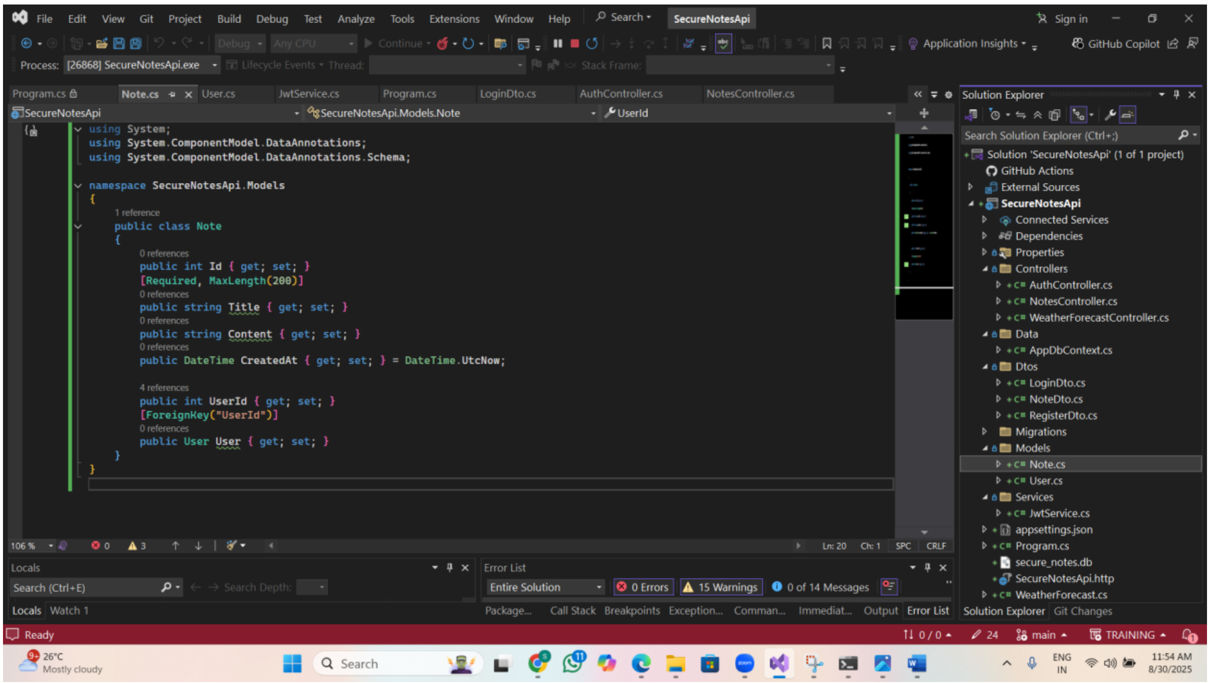
```csharp
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SecureNotesApi.Data;
using SecureNotesApi.Dtos;
using SecureNotesApi.Models;
using System.Security.Claims;

namespace SecureNotesApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize]
    public class NotesController : ControllerBase
    {
        private readonly AppDbContext _db;

        public NotesController(AppDbContext db) => _db = db;

        private int GetCurrentUserId()
        {
            return int.Parse(User.FindFirst("id").Value);
        }

        [HttpPost]
        public async Task<IActionResult> CreateNote(NoteDto dto)
        {
            var note = new Note
            {
                Title = dto.Title,
                Content = dto.Content,
                UserId = GetCurrentUserId()
            };

            _db.Notes.Add(note);
            await _db.SaveChangesAsync();

            return Ok(new { message = "Note added successfully.", noteId = note.Id });
        }

        [HttpGet]
        public async Task<IActionResult> GetNotes()
        {
            var notes = await _db.Notes
                .Where(n => n.UserId == GetCurrentUserId())
                .ToListAsync();

            return Ok(notes);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> Update(int id, NoteDto dto)
        {
            var note = await _db.Notes.SingleOrDefaultAsync(n => n.Id == id && n.UserId == GetCurrentUserId());
            if (note == null) return NotFound();

            note.Title = dto.Title;
            note.Content = dto.Content;
            await _db.SaveChangesAsync();

            return Ok(new { message = "Note updated successfully." });
        }

        [HttpDelete("{id}")]
```

User:



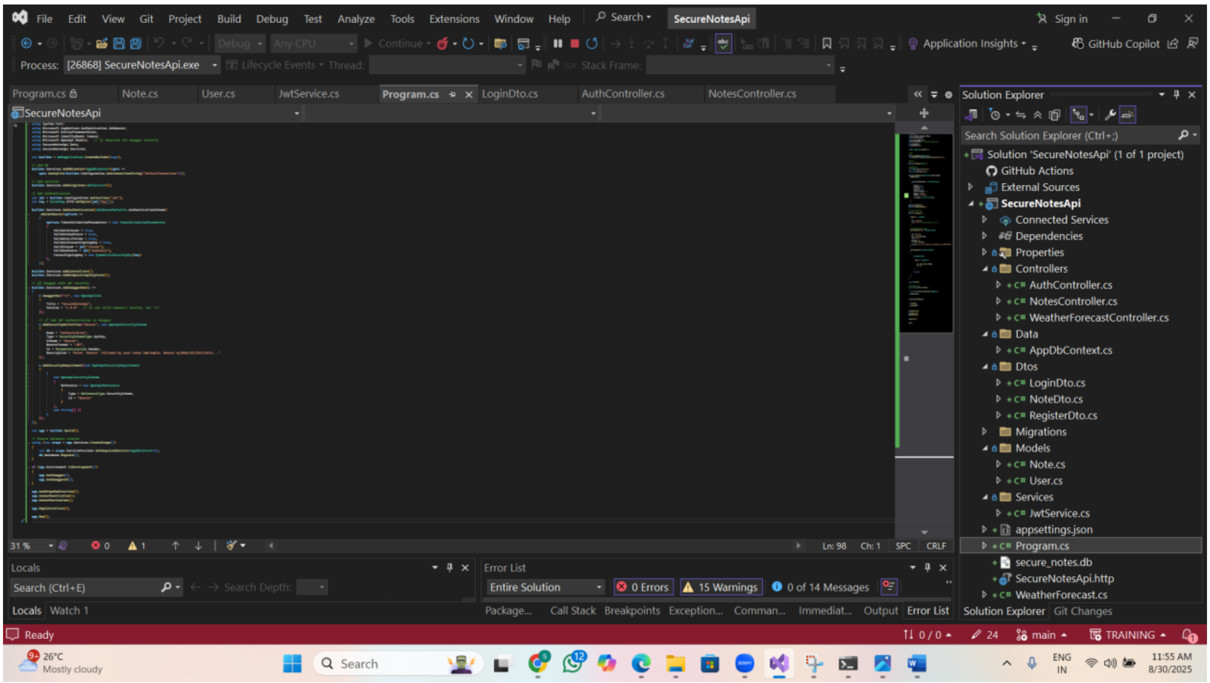
```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace SecureNotesApi.Models
{
    5 references
    public class User
    {
        1 reference
        public int Id { get; set; }
        [Required, MaxLength(100)]
        6 references
        public string Username { get; set; }
        [Required]
        2 references
        public string PasswordHash { get; set; }

        0 references
        public ICollection<Note> Notes { get; set; }
    }
}
```

Note:

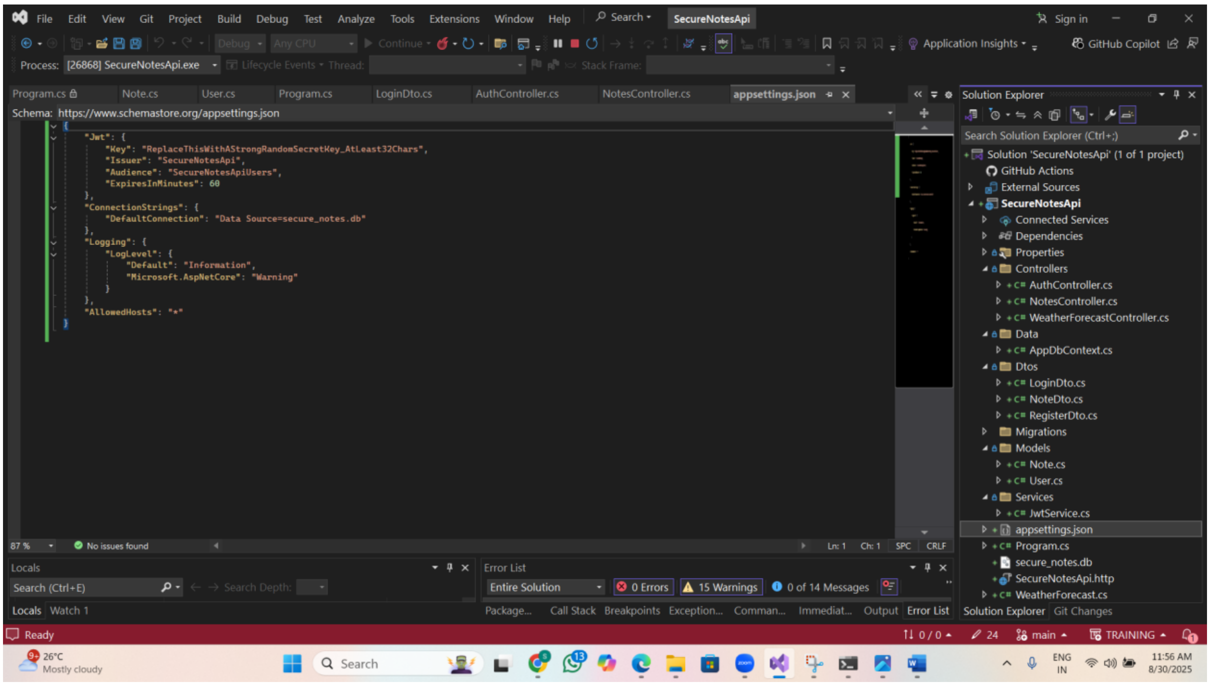


```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace SecureNotesApi.Models
{
    1 reference
    public class Note
    {
        0 references
        public int Id { get; set; }
        [Required, MaxLength(200)]
        0 references
        public string Title { get; set; }
        0 references
        public string Content { get; set; }
        0 references
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        4 references
        public int UserId { get; set; }
        [ForeignKey("UserId")]
        0 references
        public User User { get; set; }
    }
}
```

Program.cs:

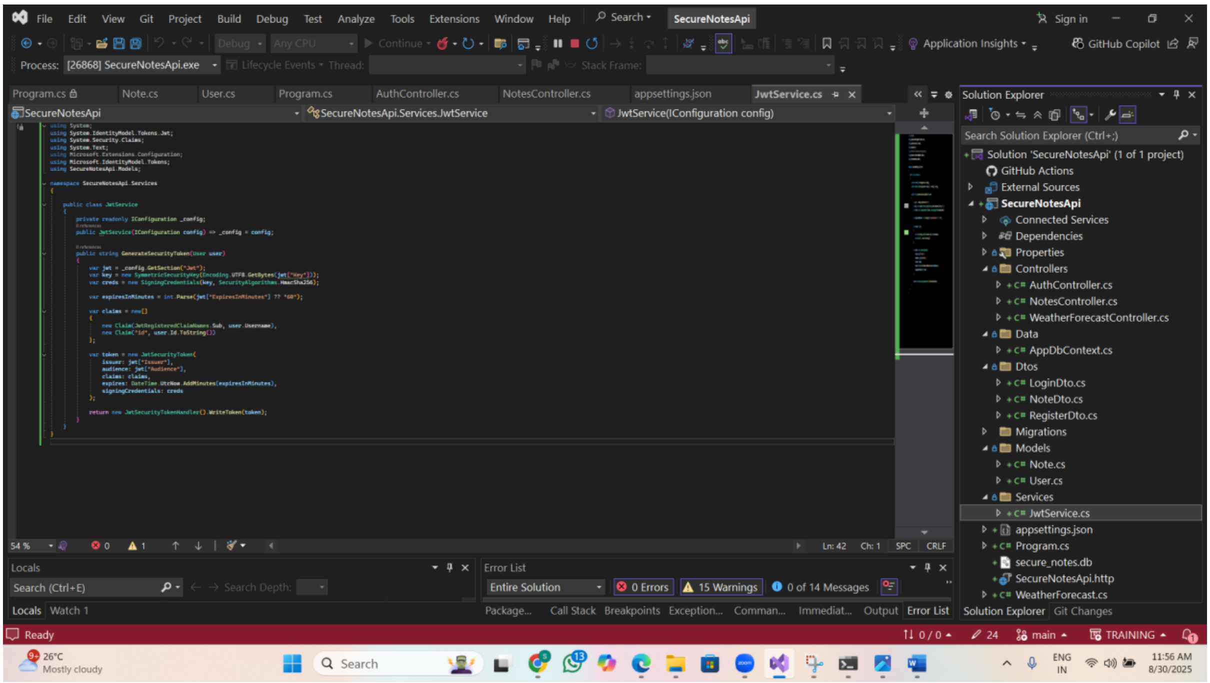


Appsetting.json:



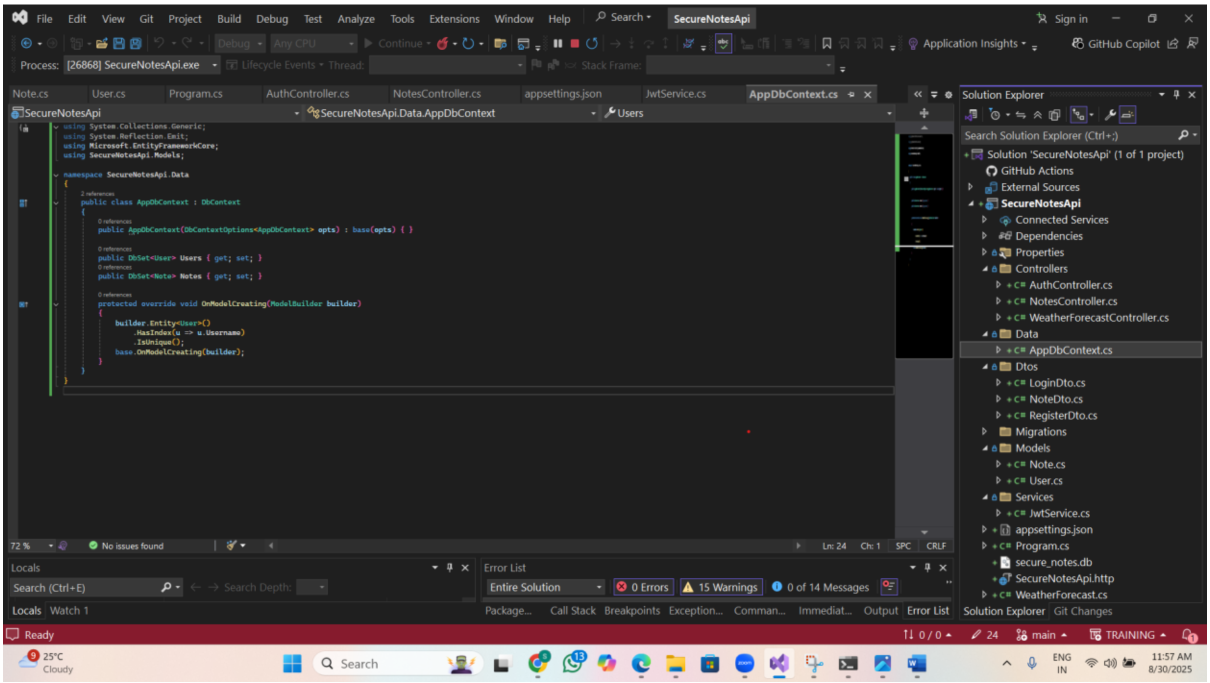
```
{
    "Jwt": {
        "Key": "ReplaceThisWithAStrongRandomSecretKey_AtLeast32Chars",
        "Issuer": "SecureNotesApi",
        "Audience": "SecureNotesApiUsers",
        "ExpiresInMinutes": 60
    },
    "ConnectionStrings": {
        "DefaultConnection": "Data Source=secure_notes.db"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning"
        }
    },
    "AllowedHosts": "*"
}
```

JwtService:



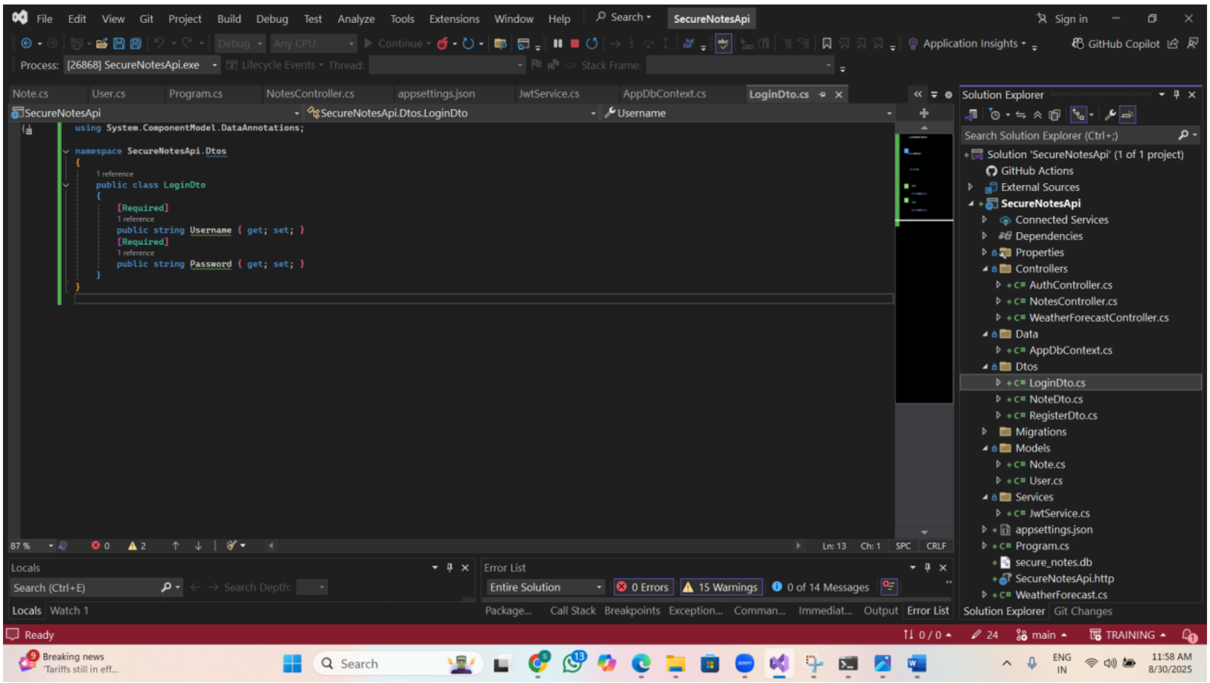
```
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using SecureNotesApi.Models;

namespace SecureNotesApi.Services
{

    public class JwtService
    {
        private readonly IConfiguration _config;
        public JwtService(IConfiguration config) => _config = config;

        public string GenerateSecurityToken(User user)
        {
            var jwt = _config.GetSection("Jwt");
            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwt["Key"]));
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var expiresInMinutes = int.Parse(jwt["ExpiresInMinutes"] ?? "60");

            var claims = new[]
            {
                new Claim(JwtRegisteredClaimNames.Sub, user.Username),
                new Claim("id", user.Id.ToString())
            };

            var token = new JwtSecurityToken(
                issuer: jwt["Issuer"],
                audience: jwt["Audience"],
                claims: claims,
                expires: DateTime.UtcNow.AddMinutes(expiresInMinutes),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

AppDbContext:



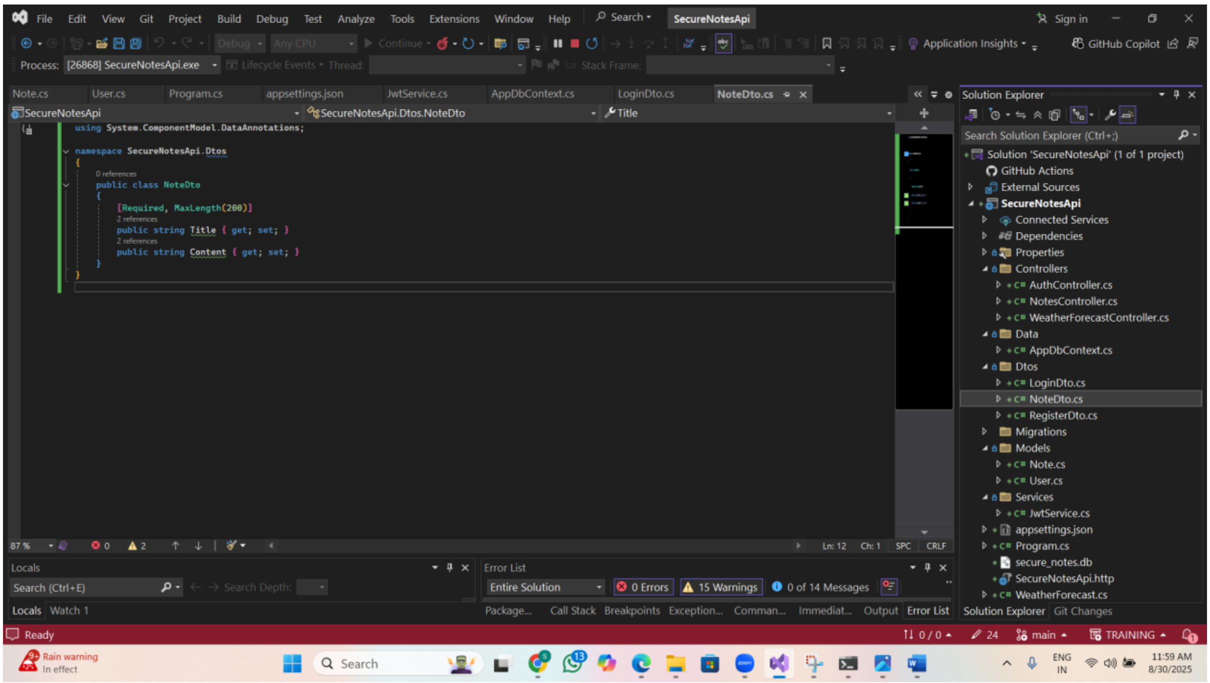
```
using System.Collections.Generic;
using System.Reflection.Emit;
using Microsoft.EntityFrameworkCore;
using SecureNotesApi.Models;

namespace SecureNotesApi.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> opts) : base(opts) { }

        public DbSet<User> Users { get; set; }
        public DbSet<Note> Notes { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<User>()
                .HasIndex(u => u.Username)
                .IsUnique();
            base.OnModelCreating(builder);
        }
    }
}
```

LoginDto:



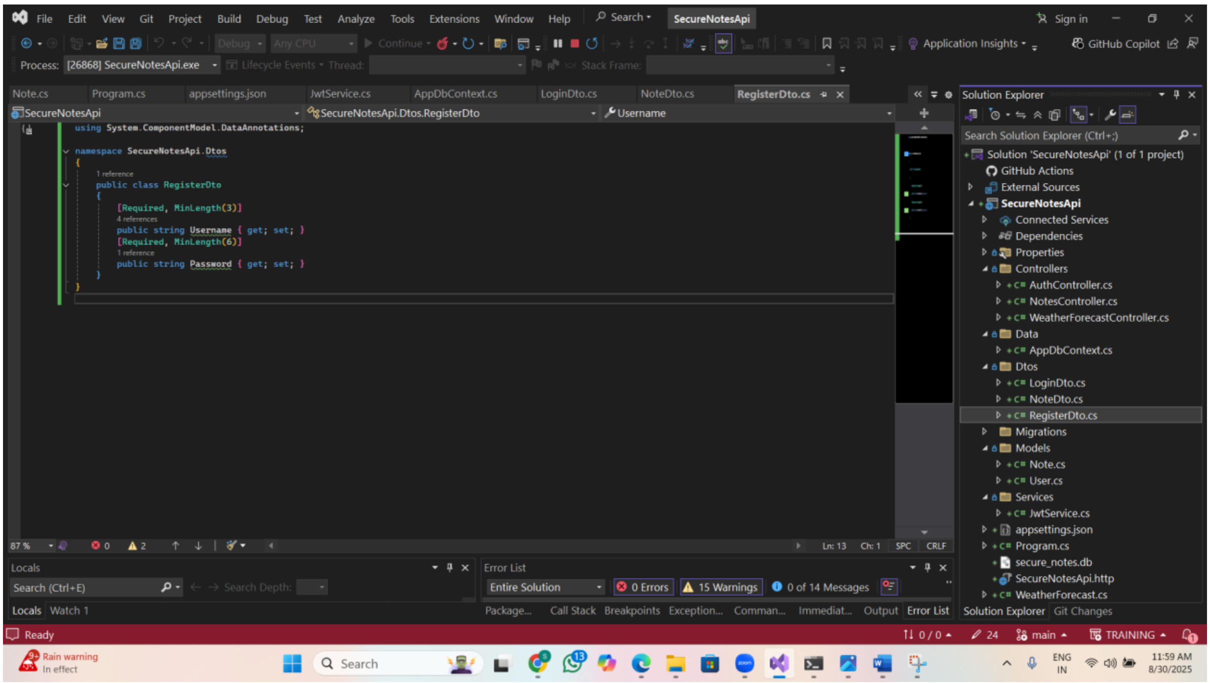
```
using System.ComponentModel.DataAnnotations;

namespace SecureNotesApi.Dtos
{
    public class LoginDto
    {
        [Required]
        public string Username { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

NoteDto:



```
using System.ComponentModel.DataAnnotations;

namespace SecureNotesApi.Dtos
{
    public class NoteDto
    {
        [Required, MaxLength(200)]
        public string Title { get; set; }
        public string Content { get; set; }
    }
}
```

RegisterDto:



```
using System.ComponentModel.DataAnnotations;

namespace SecureNotesApi.Dtos
{
    public class RegisterDto
    {
        [Required, MinLength(3)]
        public string Username { get; set; }
        [Required, MinLength(6)]
        public string Password { get; set; }
    }
}
```

4. Implementation Details

4.1 User Registration

- Endpoint: POST /api/auth/register
- Stores username & hashed password in DB.
- Validates unique username & strong password.

4.2 User Login

- Endpoint: POST /api/auth/login
- Verifies username & password.
- On success, returns JWT Token (with expiry time).

4.3 Manage Notes

- Endpoints:
    - POST /api/notes → Add note

  - GET /api/notes → Get all notes of logged-in user
  - PUT /api/notes/{id} → Update note
  - DELETE /api/notes/{id} → Delete note
- Only works if request contains valid JWT token in Authorization header.

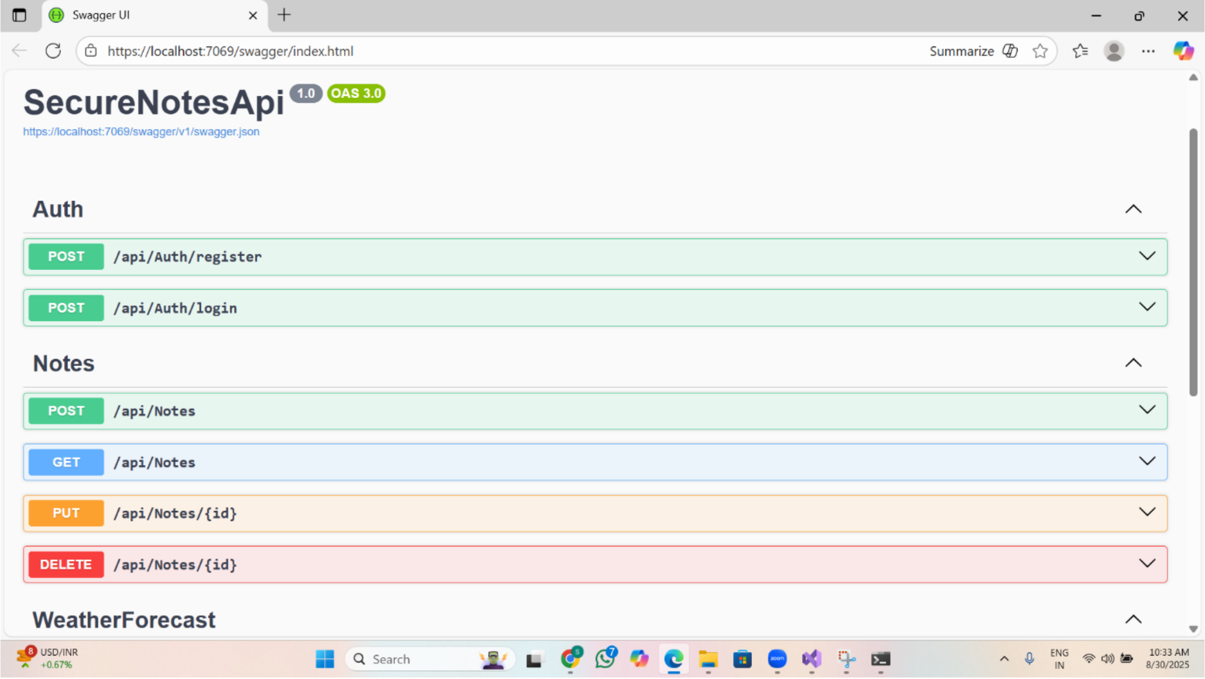
4.4 Security Features

- Passwords stored using hashing.
- JWT ensures only authenticated users can access their data.
- Input validation for username & password strength.

---

5. Sample Input & Output

Registration Input



SecureNotesApi 1.0 OAS 3.0

https://localhost:7069/swagger/v1/swagger.json

Auth

POST /api/Auth/register

POST /api/Auth/login

Notes

POST /api/Notes

GET /api/Notes

PUT /api/Notes/{id}

DELETE /api/Notes/{id}

WeatherForecast

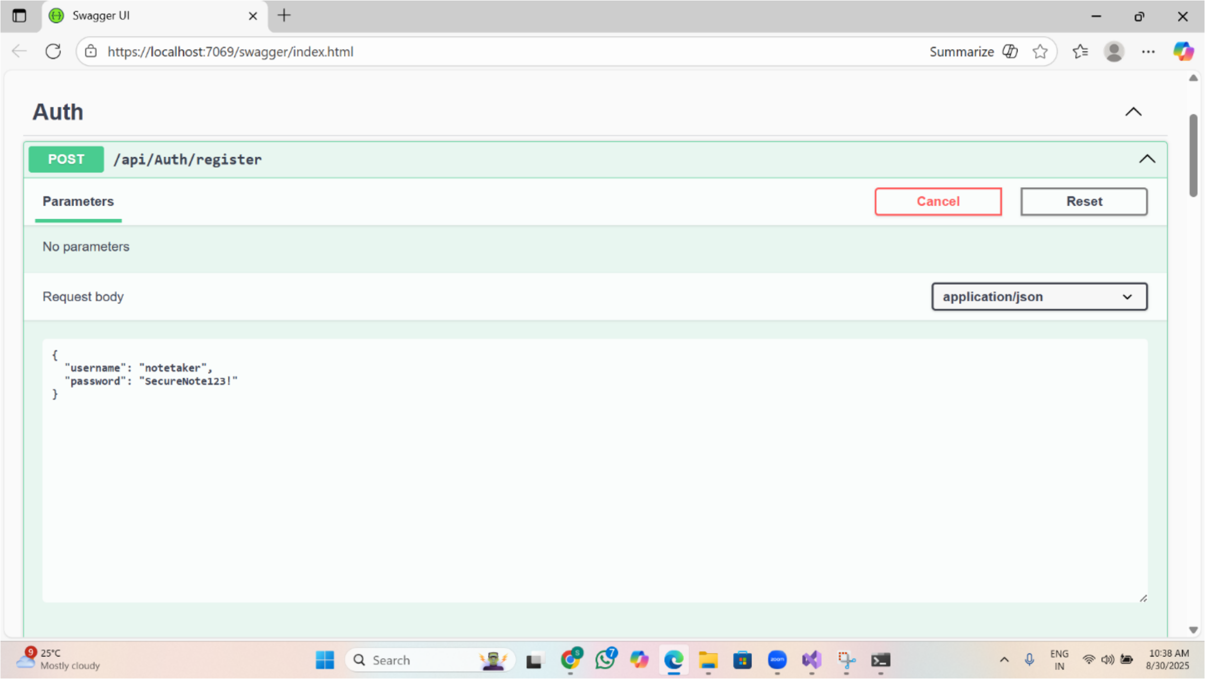
POST /api/auth/register

{

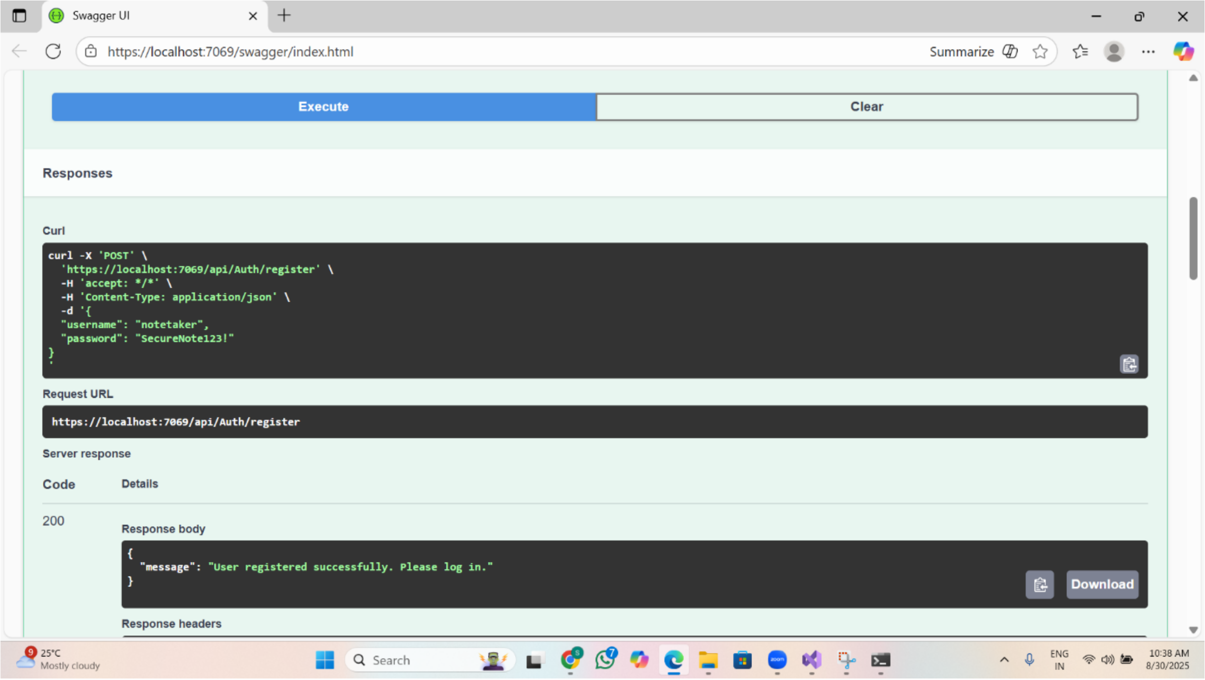
```
  "username": "notetaker",
  "password": "SecureNote123!"
}
```

Response

{ "message": "User registered successfully. Please log in." }



```
{
  "username": "notetaker",
  "password": "SecureNote123!"
}
```



Responses

Curl

```
curl -X 'POST' \
  'https://localhost:7069/api/Auth/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "notetaker",
  "password": "SecureNote123!"
}
'
```

Request URL

```
https://localhost:7069/api/Auth/register
```

Server response

Code Details

200

Response body

```
{
  "message": "User registered successfully. Please log in."
}
```

Response headers

Login Input

POST /api/auth/login

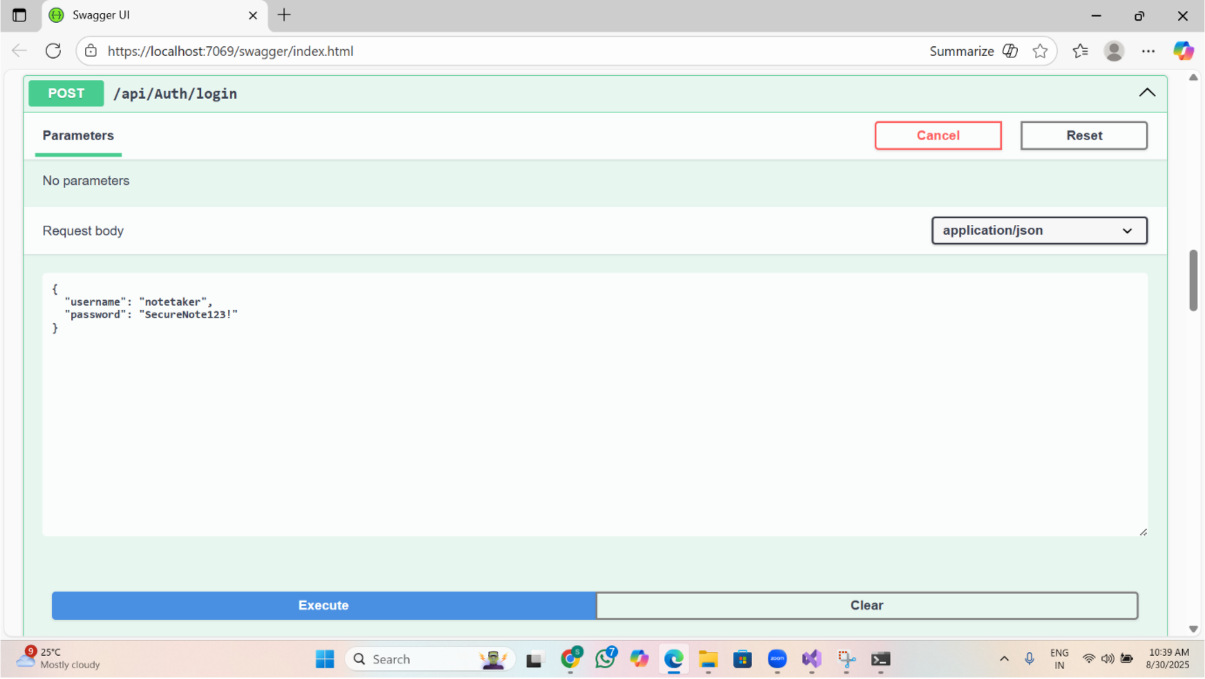
```
{
  "username": "notetaker",
  "password": "SecureNote123!"
}
```

Response

```
{
  "token": "<jwt-token>",
  "expires_in": 3600,
  "user": { "username": "notetaker" }
```

}

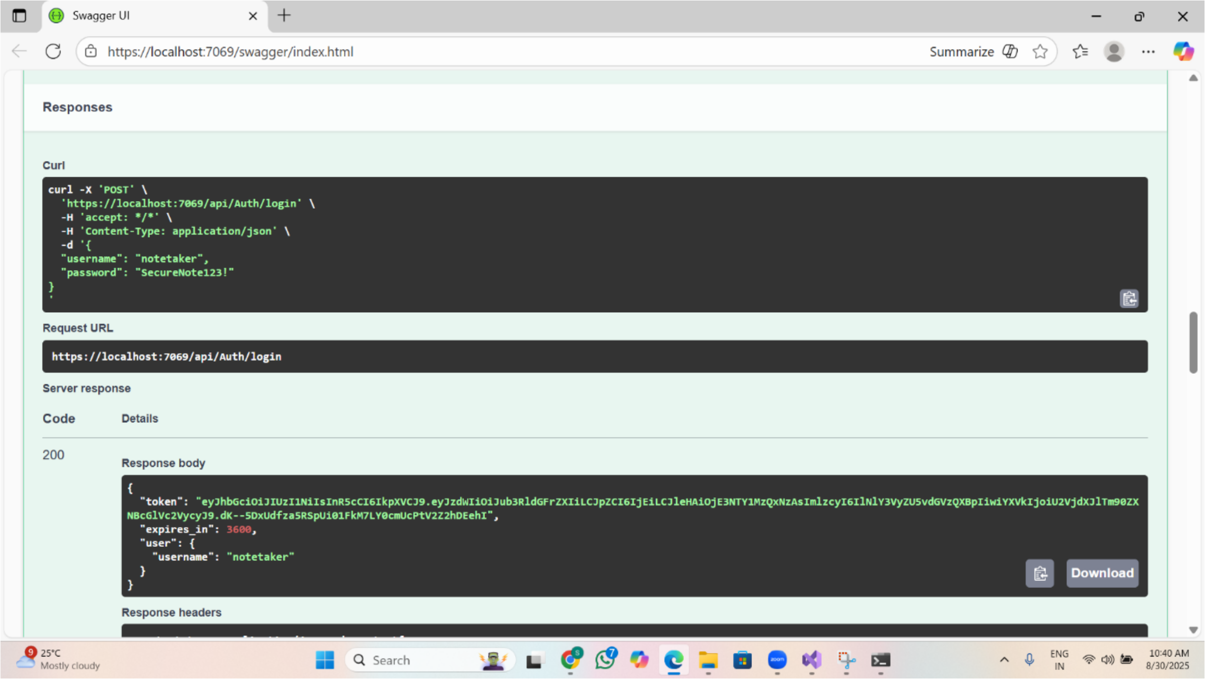
Add Note Input (with JWT token)



No parameters

Request body: application/json

```
{
  "username": "notetaker",
  "password": "SecureNote123!"
}
```



Responses

Curl

```
curl -X 'POST' \
  'https://localhost:7069/api/Auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "notetaker",
  "password": "SecureNote123!"
}'
```

Request URL

```
https://localhost:7069/api/Auth/login
```

Server response

Code: 200

Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJub3RldGFrZXIiLCJpZCI6IjEiLCJleHAiOjE3NTY1MzQxNzAsImlzcyI6IlNlY3VyZU5vdGVzQXBpIiwiYXVkIjoiU2VjdXJlTm90ZXNBcGlVc2VycyJ9.dK--5DxUdfza5RSpUi01FkM7LY0cmUcPtV2Z2hDEehI",
  "expires_in": 3600,
  "user": {
    "username": "notetaker"
  }
}
```

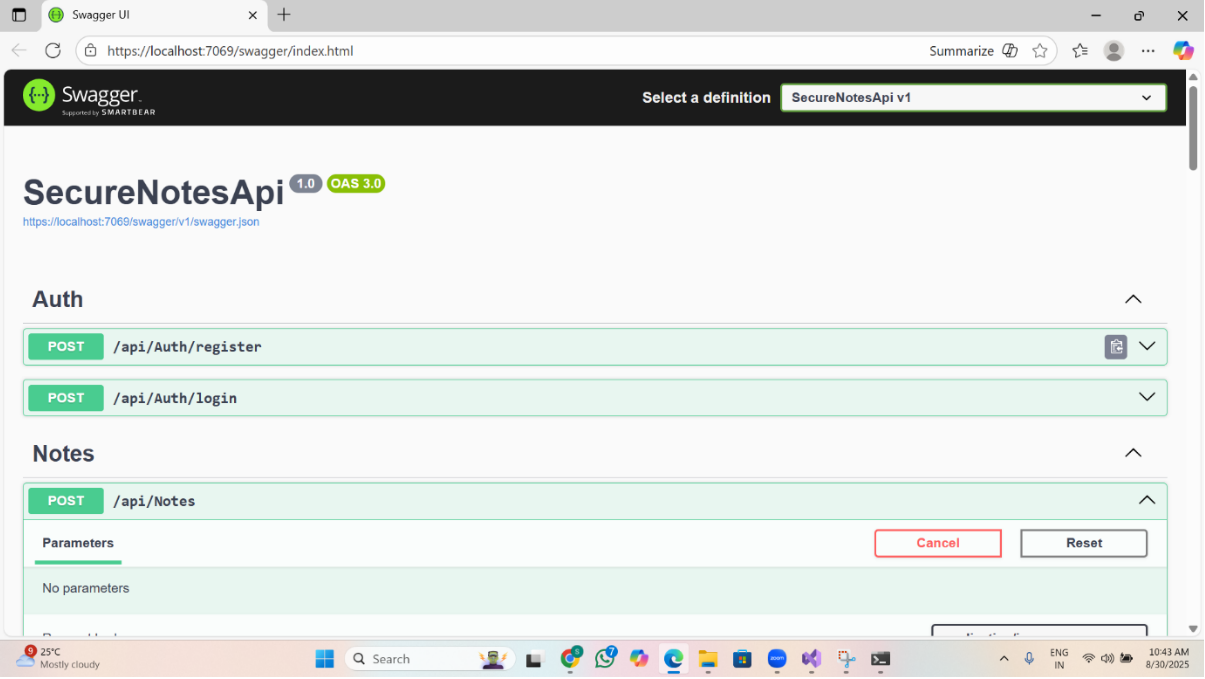
Response headers

POST /api/notes

```
{
  "title": "Grocery List",
  "content": "Eggs, Milk, Bread"
}
```

Response

```
{
  "message": "Note added successfully.",
  "noteId": 1
}
```



https://localhost:7069/swagger/index.html

Select a definition: SecureNotesApi v1

SecureNotesApi 1.0 OAS 3.0

https://localhost:7069/swagger/v1/swagger.json

Auth

POST /api/Auth/register

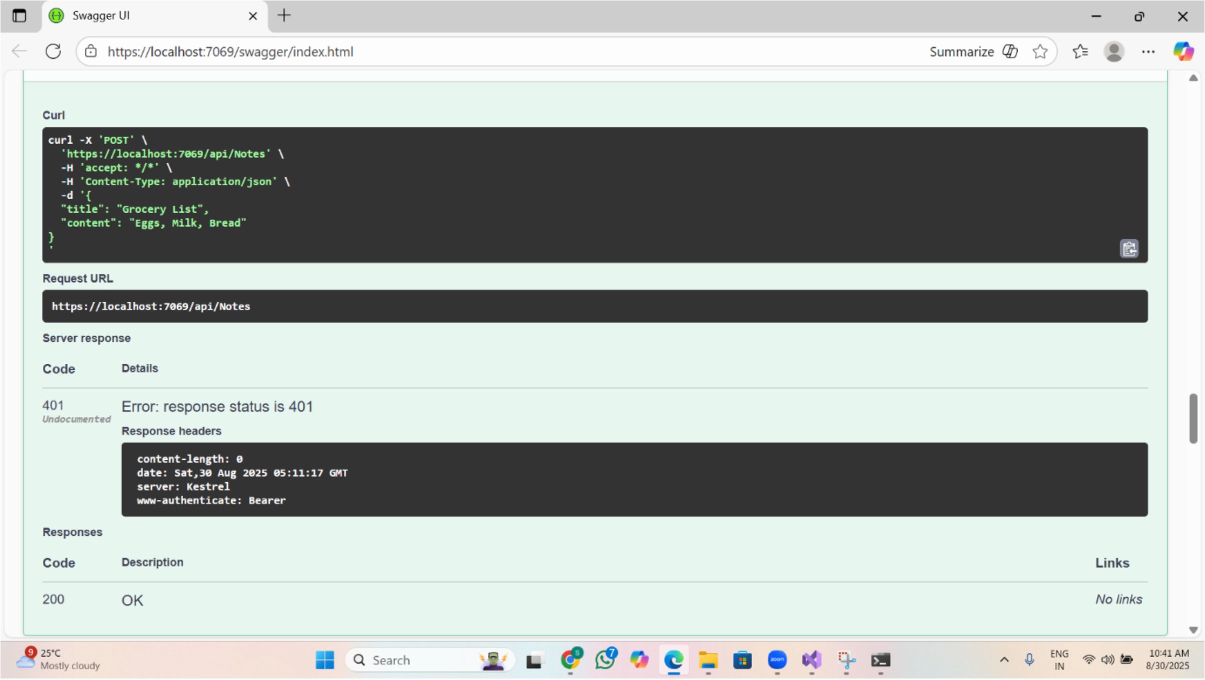
POST /api/Auth/login

Notes

POST /api/Notes

Parameters

No parameters



Curl

```
curl -X 'POST' \
  'https://localhost:7069/api/Notes' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "Grocery List",
  "content": "Eggs, Milk, Bread"
}
'
```

Request URL

```
https://localhost:7069/api/Notes
```

Server response

Code	Details
401 *Undocumented*	Error: response status is 401 Response headers

```
content-length: 0
date: Sat,30 Aug 2025 05:11:17 GMT
server: Kestrel
www-authenticate: Bearer
```

Responses

Code	Description	Links
200	OK	*No links*