Caselet: Building an API Core Project for a Hotel Booking System

VASANTHABALAN M

Client: XYZ Hotels

Overview: XYZ Hotels is a renowned hotel chain operating globally. To streamline their reservation process and provide an efficient booking experience to their customers, XYZ Hotels plans to develop an API Core Project using a code-first approach. The project should include CRUD operations, filtering

capabilities, count functionality, JWT token authentication, and handle the one-to-many relationship between hotels and rooms.

Challenge: XYZ Hotels needs a comprehensive API Core Project that enables customers to make reservations, hotel staff to manage room availability, and provides secure access through JWT token authentication. Additionally, the project should handle the one-to-many relationship between hotels and rooms, where each hotel can have multiple rooms.

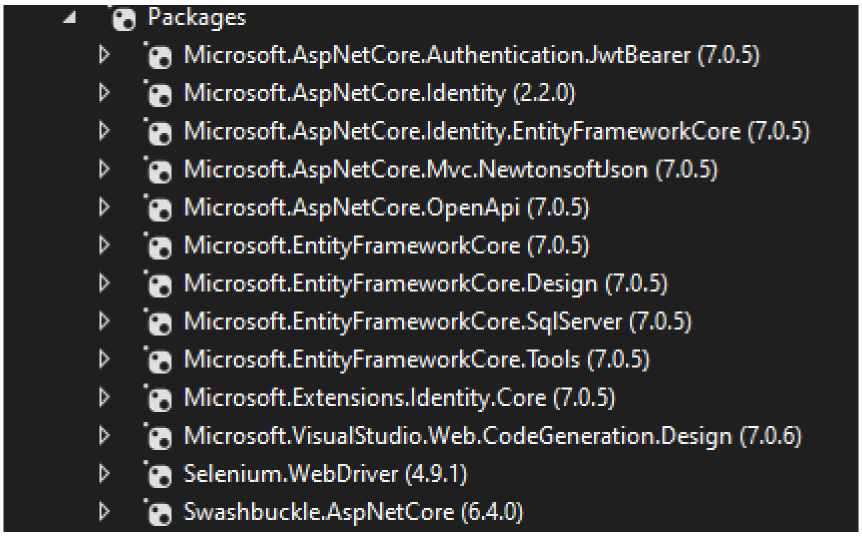
Objectives:

1. **CRUD Operations**: Develop APIs to support CRUD operations for managing hotel information, including creating new hotels, updating hotel details, retrieving hotel information, and deleting hotels.
2. **Filtering**: Implement filtering capabilities to allow customers to search and filter hotels based on criteria such as location, price range, or amenities.
3. **Count Functionality**: Enable users to obtain counts of available rooms in specific hotels, providing insights into room availability for better decision-making.
4. **JWT Token Authentication**: Implement a secure authentication mechanism using JWT tokens to ensure that only authorized users can access the API endpoints, safeguarding customer and hotel data.
5. **One-to-Many Relationship**: Establish an efficient solution to handle the one-to-many relationship between hotels and rooms, where each hotel can have multiple rooms.
6. **Exception Handling**: Implement try-catch blocks to handle exceptions gracefully, providing meaningful error messages and ensuring the system's stability.
7. **Repository Pattern**: Apply the repository pattern to separate the data access layer from the

business logic, promoting code modularity and maintainability.

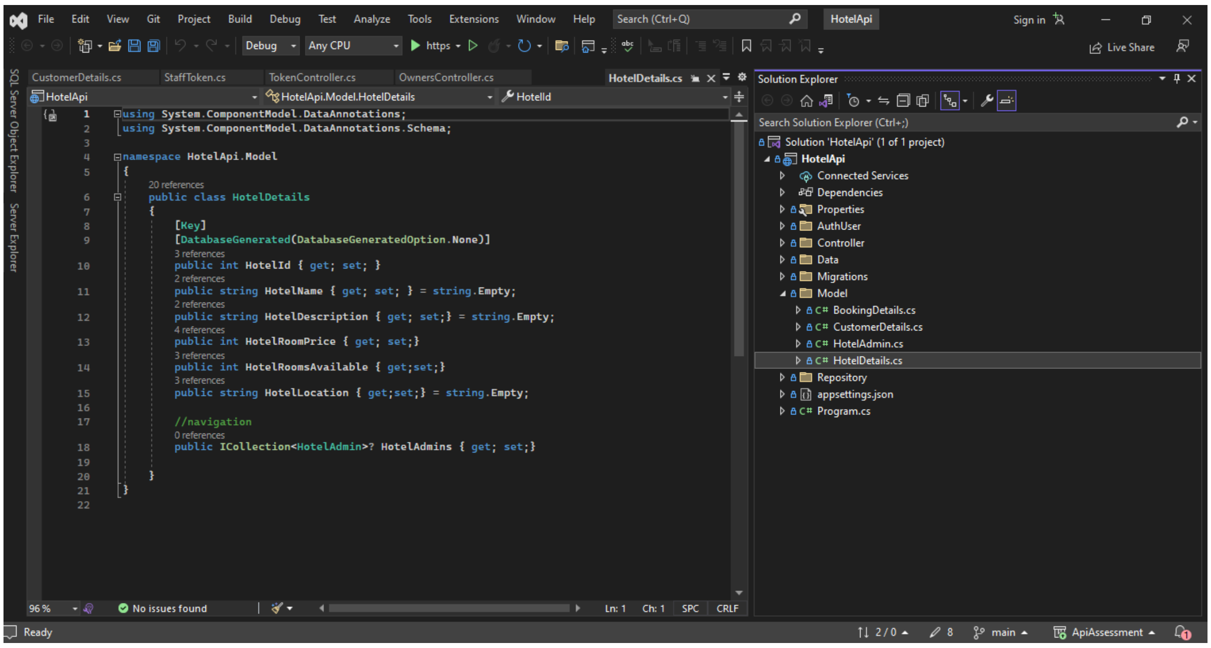
Solution:

STEP 1 : install nuget packages



```
Packages
    Microsoft.AspNetCore.Authentication.JwtBearer (7.0.5)
    Microsoft.AspNetCore.Identity (2.2.0)
    Microsoft.AspNetCore.Identity.EntityFrameworkCore (7.0.5)
    Microsoft.AspNetCore.Mvc.NewtonsoftJson (7.0.5)
    Microsoft.AspNetCore.OpenApi (7.0.5)
    Microsoft.EntityFrameworkCore (7.0.5)
    Microsoft.EntityFrameworkCore.Design (7.0.5)
    Microsoft.EntityFrameworkCore.SqlServer (7.0.5)
    Microsoft.EntityFrameworkCore.Tools (7.0.5)
    Microsoft.Extensions.Identity.Core (7.0.5)
    Microsoft.VisualStudio.Web.CodeGeneration.Design (7.0.6)
    Selenium.WebDriver (4.9.1)
    Swashbuckle.AspNetCore (6.4.0)
```

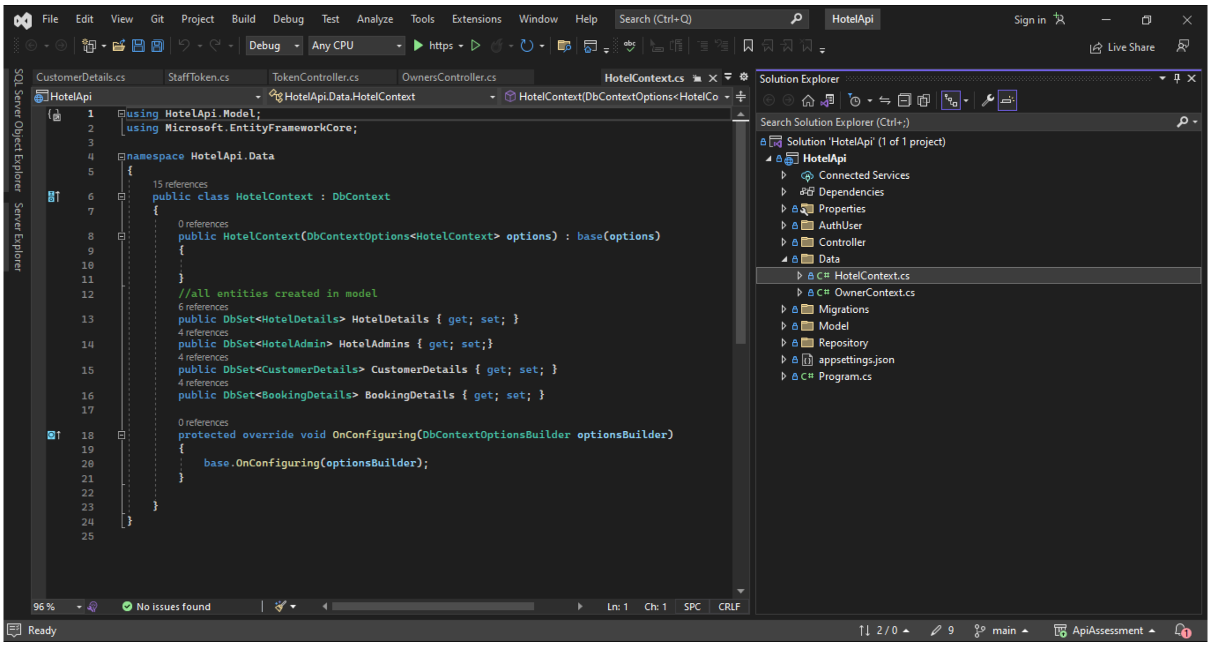
STEP 2 : Create Model



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

namespace HotelApi.Model
{
    public class HotelDetails
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int HotelId { get; set; }
        public string HotelName { get; set; } = string.Empty;
        public string HotelDescription { get; set;} = string.Empty;
        public int HotelRoomPrice { get; set;}
        public int HotelRoomsAvailable { get;set;}
        public string HotelLocation { get;set;} = string.Empty;

        //navigation
        public ICollection<HotelAdmin>? HotelAdmins { get; set;}

    }
}
```

```
Solution 'HotelApi' (1 of 1 project)
  HotelApi
    Connected Services
    Dependencies
    Properties
    AuthUser
    Controller
    Data
    Migrations
    Model
      BookingDetails.cs
      CustomerDetails.cs
      HotelAdmin.cs
      HotelDetails.cs
    Repository
    appsettings.json
    Program.cs
```

STEP 3 : Create Data(Datacontext)



```
using HotelApi.Model;
using Microsoft.EntityFrameworkCore;

namespace HotelApi.Data
{
    15 references
    public class HotelContext : DbContext
    {
        0 references
        public HotelContext(DbContextOptions<HotelContext> options) : base(options)
        {

        }
        //all entities created in model
        6 references
        public DbSet<HotelDetails> HotelDetails { get; set; }
        4 references
        public DbSet<HotelAdmin> HotelAdmins { get; set;}
        4 references
        public DbSet<CustomerDetails> CustomerDetails { get; set; }
        4 references
        public DbSet<BookingDetails> BookingDetails { get; set; }

        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
        }

    }
}
```

STEP 4: Connection string (appsetings.json)

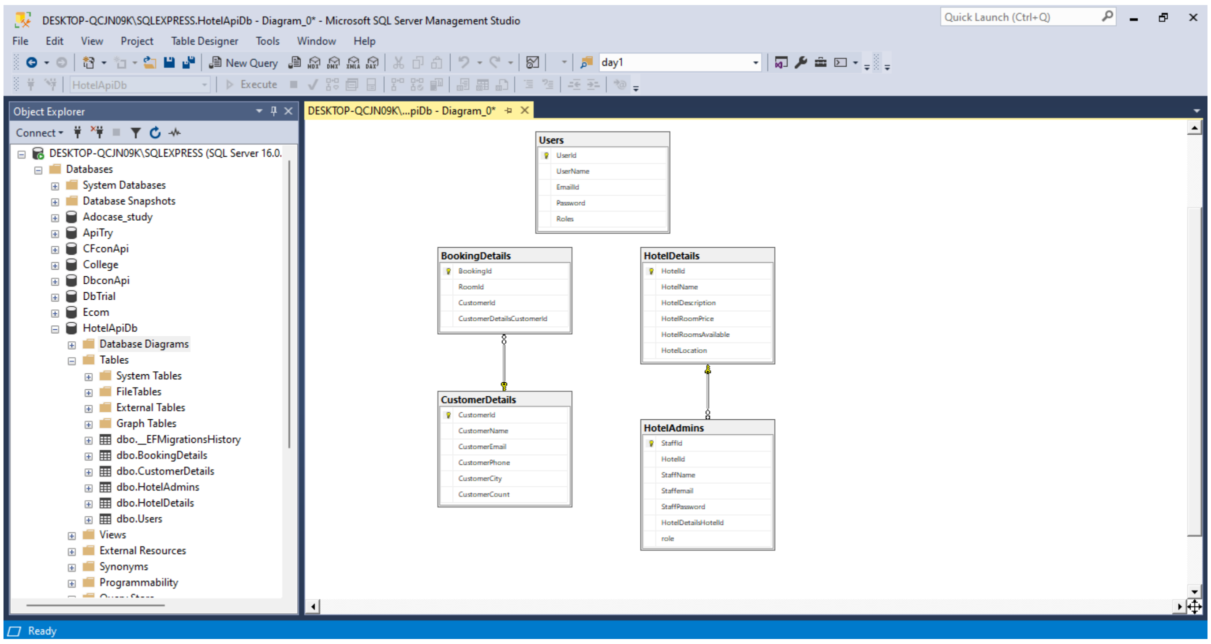
```
"ConnectionStrings": {
    "HotelApi": "data source = .\\SQLEXPRESS; initial catalog = HotelApiDb;
integrated security=SSPI;TrustServerCertificate=True;",
    }
```

```
Program.cs

builder.Services.AddDbContext<HotelContext>(optionsAction: options =>
options.UseSqlServer(builder.Configuration.GetConnectionString(name:
"HotelApi")));
```

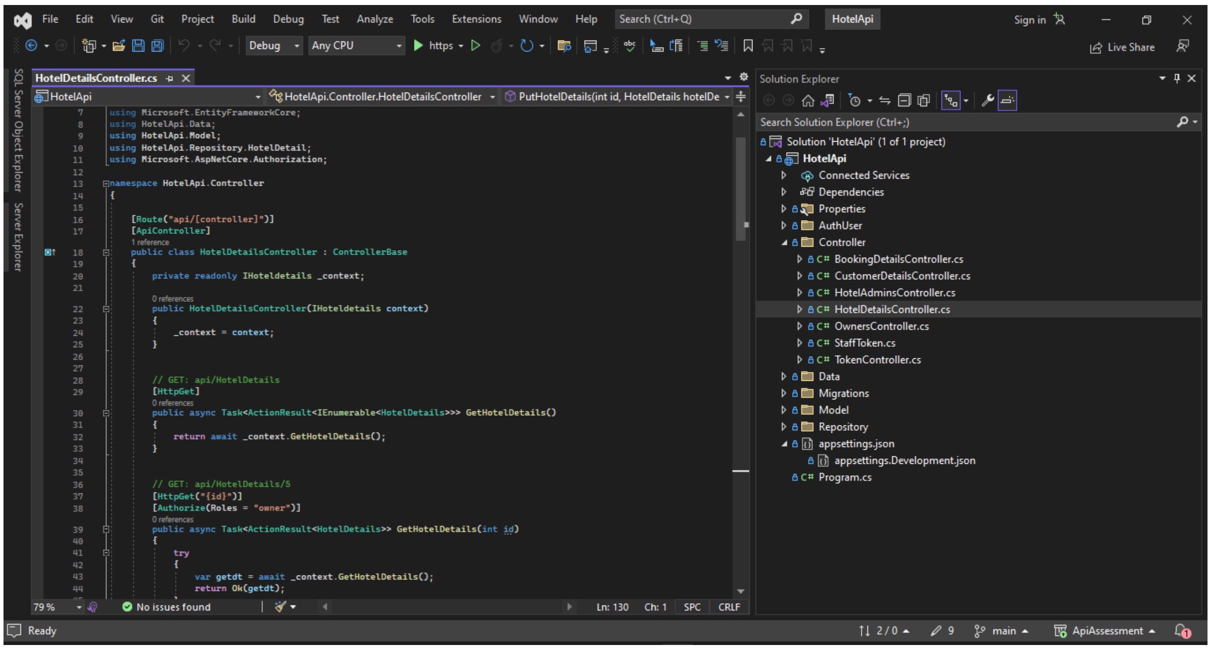
STEP 5 : Add Migration

Dotnet ef migrations add hoteldb

Dotnet ef databse update

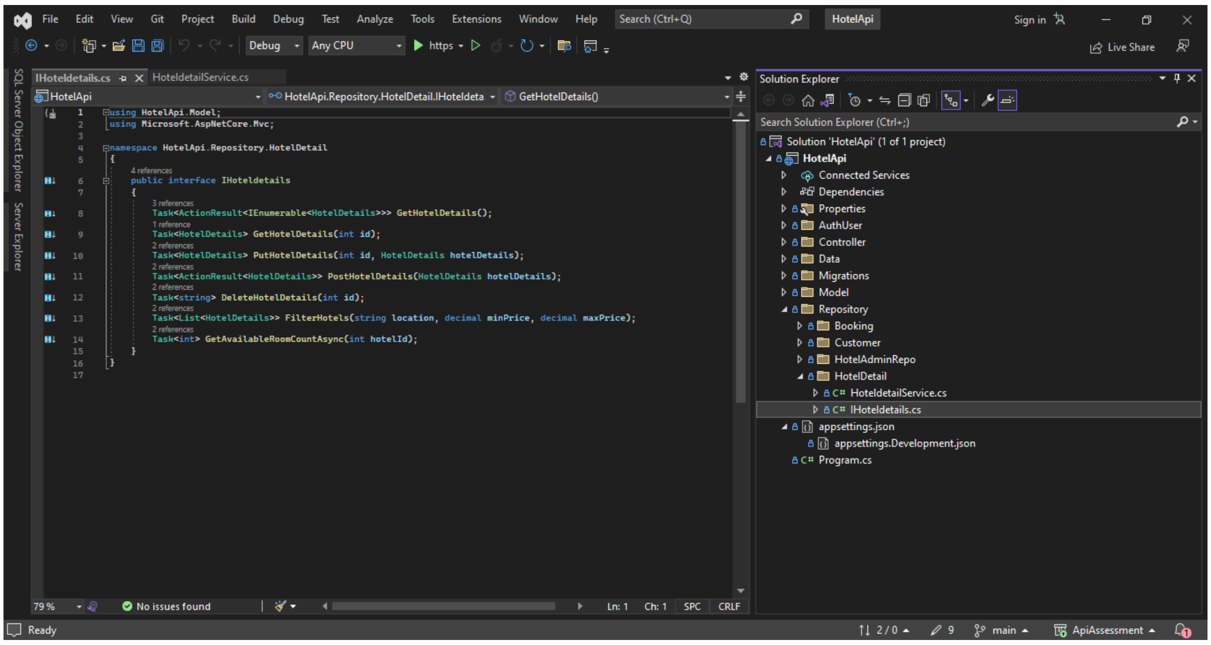

DESKTOP-QCJN09K\SQLEXPRESS.HotelApiDb - Diagram_0* - Microsoft SQL Server Management Studio
Object Explorer
HotelApiDb
Users: UserId, UserName, EmailId, Password, Roles
BookingDetails: BookingId, RoomId, CustomerId, CustomerDetailsCustomerId
HotelDetails: HotelId, HotelName, HotelDescription, HotelRoomPrice, HotelRoomsAvailable, HotelLocation
CustomerDetails: CustomerId, CustomerName, CustomerEmail, CustomerPhone, CustomerCity, CustomerCount
HotelAdmins: StaffId, HotelId, StaffName, Staffemail, StaffPassword, HotelDetailsHotelId, role

STEP 6 : Create Controller


HotelDetailsController.cs
Solution Explorer
Solution 'HotelApi' (1 of 1 project)
HotelApi
Controller
BookingDetailsController.cs
CustomerDetailsController.cs
HotelAdminsController.cs
HotelDetailsController.cs
OwnersController.cs
StaffToken.cs
TokenController.cs

```
using Microsoft.EntityFrameworkCore;
using HotelApi.Data;
using HotelApi.Model;
using HotelApi.Repository.HotelDetail;
using Microsoft.AspNetCore.Authorization;

namespace HotelApi.Controller
{

    [Route("api/[controller]")]
    [ApiController]
    public class HotelDetailsController : ControllerBase
    {
        private readonly IHoteldetails _context;

        public HotelDetailsController(IHoteldetails context)
        {
            _context = context;
        }


        // GET: api/HotelDetails
        [HttpGet]
        public async Task<ActionResult<IEnumerable<HotelDetails>>> GetHotelDetails()
        {
            return await _context.GetHotelDetails();
        }


        // GET: api/HotelDetails/5
        [HttpGet("{id}")]
        [Authorize(Roles = "owner")]
        public async Task<ActionResult<HotelDetails>> GetHotelDetails(int id)
        {
            try
            {
                var getdt = await _context.GetHotelDetails();
                return Ok(getdt);
```

STEP 7 : Create Repository(service class)



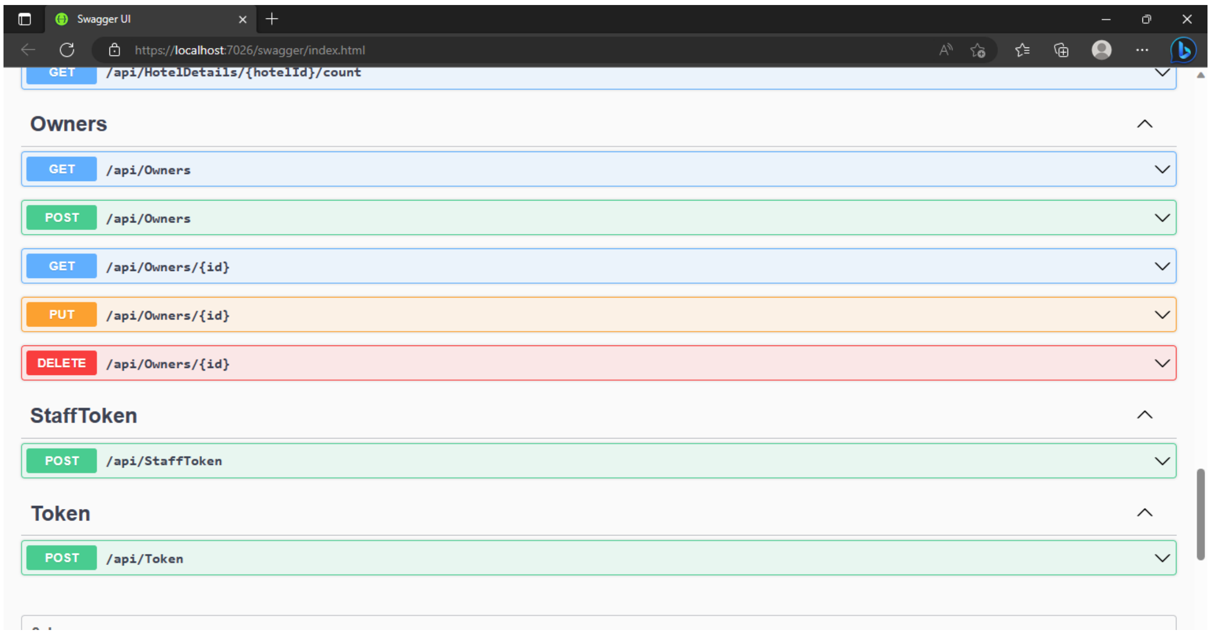
```
using HotelApi.Model;
using Microsoft.AspNetCore.Mvc;

namespace HotelApi.Repository.HotelDetail
{
    4 references
    public interface IHoteldetails
    {
        3 references
        Task<ActionResult<IEnumerable<HotelDetails>>> GetHotelDetails();
        1 reference
        Task<HotelDetails> GetHotelDetails(int id);
        2 references
        Task<HotelDetails> PutHotelDetails(int id, HotelDetails hotelDetails);
        2 references
        Task<ActionResult<HotelDetails>> PostHotelDetails(HotelDetails hotelDetails);
        2 references
        Task<string> DeleteHotelDetails(int id);
        2 references
        Task<List<HotelDetails>> FilterHotels(string location, decimal minPrice, decimal maxPrice);
        2 references
        Task<int> GetAvailableRoomCountAsync(int hotelId);
    }
}
```

STEP 8 : create Authentication

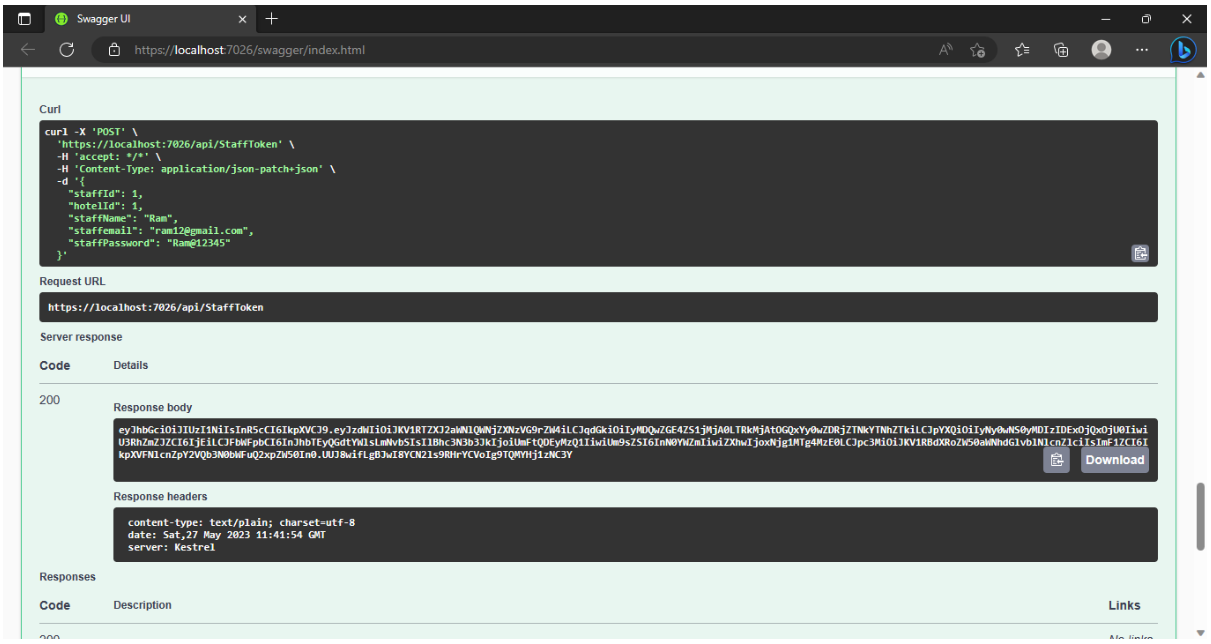
Similar to create a model,datacontext,create connection,add context in program.cs the migration in same db

STEP 9 : To give [Authorize (Role=”staff”)] for necessary file controllers

STEP 10: After executed it shows the staff login to access token



After get username password from the admin it generate token



Curl

```
curl -X 'POST' \
  'https://localhost:7026/api/StaffToken' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json-patch+json' \
  -d '{
    "staffId": 1,
    "hotelId": 1,
    "staffName": "Ram",
    "staffemail": "ram12@gmail.com",
    "staffPassword": "Ram@12345"
  }'
```

Request URL

```
https://localhost:7026/api/StaffToken
```

Server response

Code | Details
200

Response body

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJKV1RTZXJ2aWNlQWNjZXNzVG9rZW4iLCJqdGkiOiIyMDQwZGE4ZS1jMjA0LTRkMjAtOGQxYy0wZDRjZTNkYTNhZTkiLCJpYXQiOiIyNy0wNS0yMDIzIDExOjQxOjU0IiwiU3RhZmZJZCI6IjEiLCJFbWFpbCI6InJhbTEyQGdtYWlsLmNvbSIsIlBhc3N3b3JkIjoiUmFtQDEyMzQ1IiwiUm9sZSI6InN0YWZmIiwiZXhwIjoxNjg1MTg4MzE0LCJpc3MiOiJKV1RBdXRoZW50aWNhdGlvblNlcnZlciIsImF1ZCI6IkpXVFNlcnZpY2VQb3N0bWFuQ2xpZW50In0.UUJ8wifLgBJwI8YCN21s9RHrYCVoIg9TQMYHj1zNC3Y
```

Response headers

```
content-type: text/plain; charset=utf-8
date: Sat,27 May 2023 11:41:54 GMT
server: Kestrel
```

Responses

Code | Description | Links

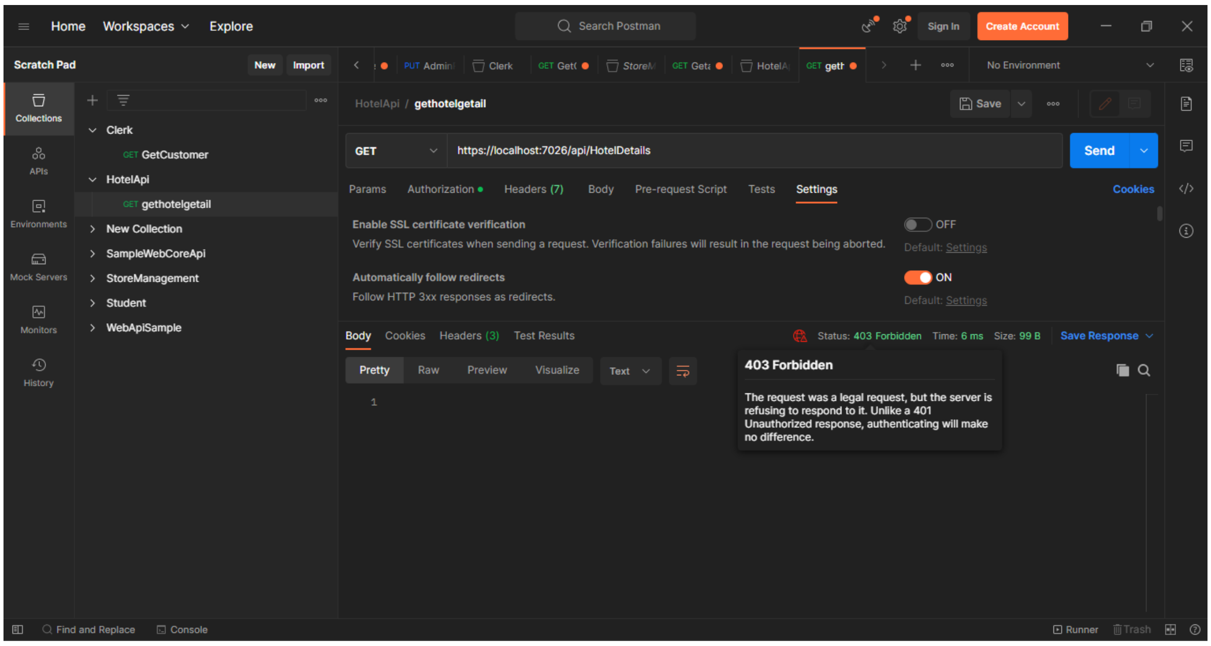
If it put postman it wont display the details of customer it shows **403** forbidden message.

Because I authorize only for **role=staff** it will check in JWT.io website it will display the output!!..

For eg: I give authorize for owner table after generate token to hotel details

Because I authorize only for **role=user**

it will check in JWT.io website it will display the token who generate!!..



HotelApi / gethotelgetall

GET https://localhost:7026/api/HotelDetails

Settings

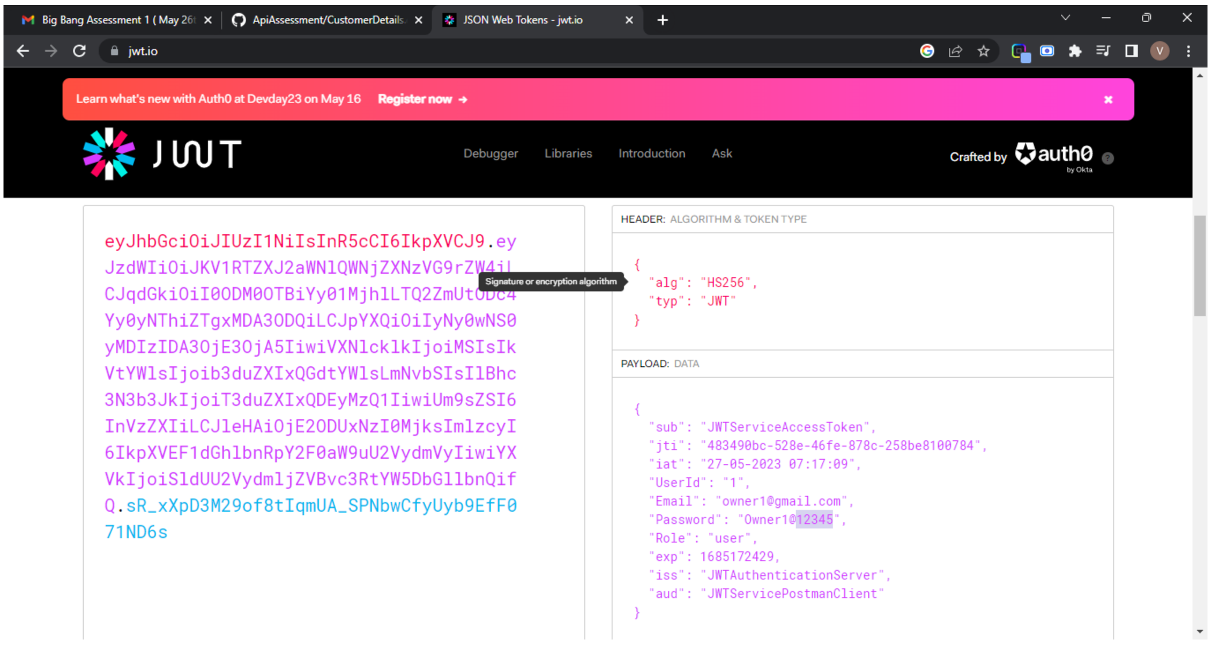
Enable SSL certificate verification — OFF
Verify SSL certificates when sending a request. Verification failures will result in the request being aborted.

Automatically follow redirects — ON
Follow HTTP 3xx responses as redirects.

Status: 403 Forbidden Time: 6 ms Size: 99 B

403 Forbidden

The request was a legal request, but the server is refusing to respond to it. Unlike a 401 Unauthorized response, authenticating will make no difference.


Big Bang Assessment 1 (May 26
ApiAssessment/CustomerDetails
JSON Web Tokens - jwt.io
jwt.io
Learn what's new with Auth0 at Devday23 on May 16
Register now →
JWT
Debugger
Libraries
Introduction
Ask
Crafted by auth0 by Okta
Signature or encryption algorithm

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiJKV1RTZXJ2aWNlQWNjZXNzVG9rZW4iL
CJqdGkiOiI0ODM0OTBiYy01MjhlLTQ2ZmUtODc4
Yy0yNThiZTgxMDA3ODQiLCJpYXQiOiIyNy0wNS0
yMDIzIDA3OjE3OjA5IiwiVXNlcklkIjoiMSIsIk
VtYWlsIjoib3duZXIxQGdtYWlsLmNvbSIsIlBhc
3N3b3JkIjoiT3duZXIxQDEyMzQ1IiwiUm9sZSI6
InVzZXIiLCJleHAiOjE2ODUxNzI0MjksImlzcyI
6IkpXVEF1dGhlbnRpY2F0aW9uU2VydmVyIiwiYX
VkIjoiSldUU2VydmljZVBvc3RtYW5DbGllbnQif
Q.sR_xXpD3M29of8tIqmUA_SPNbwCfyUyb9EfF0
71ND6s
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "JWTServiceAccessToken",
  "jti": "483490bc-528e-46fe-878c-258be8100784",
  "iat": "27-05-2023 07:17:09",
  "UserId": "1",
  "Email": "owner1@gmail.com",
  "Password": "Owner1@12345",
  "Role": "user",
  "exp": 1685172429,
  "iss": "JWTAuthenticationServer",
  "aud": "JWTServicePostmanClient"
}
```