

Customer Management System – Backend- User Guide

Backend : Dot Net Core, Web API, Entity Framework Core, SQL Server with Clear Architecture

✓ Steps to Setup and Run .NET Core Project (CMS)

▼ Step 1: Download or Clone the Project from GitHub

```
git clone <https://github.com/SheikGH/CMS_Backend.git>
```

Example: `git clone https://github.com/SheikGH/CMS_Backend.git`

■ Step 2: Open the Solution in Visual Studio

1. Open **Visual Studio 2022+**
 2. Click "**Open a project or solution**"
 3. Navigate to the downloaded folder
 4. Open `CMS.sln`
-

□ Step 3: Restore NuGet Packages

In **Visual Studio**:

- Right-click the **Solution** → **Restore NuGet Packages**
- Or use the terminal:

```
dotnet restore
```

⚙️ □ Step 4: Update `appsettings.json` (CMS.API/appsettings.json)

Example:

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=YOUR_SERVER_NAME;Database=CMS;Trusted_Connection=True;MultipleActiveResultSets=true"
  },
  "Jwt": {
    "Key": "Your_Secret_Key_Here",
    "Issuer": "CMSApp",
    "Audience": "CMSAppUsers",
    "DurationInMinutes": 60
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
```

```
        "Microsoft.AspNetCore": "Warning"
    },
    "AllowedHosts": "*"
}
```

🔑 Replace `YOUR_SERVER_NAME` with your actual SQL Server name (e.g., `localhost\\SQLEXPRESS`)
Replace `"Your_Secret_Key_Here"` with a **strong secret key**

❏ Step 5: Create Database

1. Open **SQL Server Management Studio (SSMS)**
2. Run:

```
CREATE DATABASE CMS;
```

📦 Step 6: Run Migrations and Update Database

In **Package Manager Console**:

1. Select **CMS.Infrastructure** as the default project
2. Run:

```
Add-Migration initialDb
Update-Database
```

Make sure `Startup Project` is set to `CMS.API`

📦 Step 7: Install Required NuGet Packages

◆ CMS.Core

Install:

```
Install-Package MediatR -Version 12.2.0
Install-Package Microsoft.AspNetCore.Authentication.JwtBearer -Version 6.0.27
Install-Package Microsoft.AspNetCore.Mvc.NewtonsoftJson -Version 6.0.31
Install-Package Microsoft.EntityFrameworkCore.Design -Version 7.0.16
Install-Package Newtonsoft.Json -Version 13.0.3
Install-Package Swashbuckle.AspNetCore -Version 6.5.0
```

◆ CMS.Infrastructure

```
Install-Package Microsoft.EntityFrameworkCore
Install-Package Microsoft.EntityFrameworkCore.Design
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore.Tools
```

◆ CMS.Application

```
Install-Package AutoMapper.Extensions.Microsoft.DependencyInjection -Version 12.0.1
Install-Package MediatR -Version 12.2.0
```

```
Install-Package Microsoft.Extensions.Configuration.Abstractions -Version 7.0.0
Install-Package Microsoft.IdentityModel.Tokens -Version 6.35.0
Install-Package System.IdentityModel.Tokens.Jwt -Version 6.35.0
```

◆ CMS.API

```
Install-Package Microsoft.EntityFrameworkCore -Version 7.0.16
Install-Package Microsoft.EntityFrameworkCore.Design -Version 7.0.16
Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 7.0.16
Install-Package Microsoft.EntityFrameworkCore.Tools -Version 7.0.16
```

►□ Step 8: Run the Application

1. Set **CMS.API** as the **Startup Project**
2. Press **F5** or click **Run**
3. Swagger UI should open:

<https://localhost:7067/swagger/index.html>

🔄 Step 9: Test Endpoints via Swagger or Postman

- ✓ **POST** /api/Auth/login
 - ✓ **GET** /api/Customers
 - ✓ **POST** /api/Customers
 - ✓ **PUT** /api/Customers/{id}
 - ✓ **DELETE** /api/Customers/{id}
-

✓ Project Structure Recap (Clean Architecture)

```
CMS/
├── CMS.Core/           → Domain Models, Interfaces
├── CMS.Infrastructure/ → EF Core, SQL, Repositories
├── CMS.Application/    → Services, DTOs, Mapping, Use Cases
└── CMS.API/           → Controllers, Middleware, Auth, Swagger
```

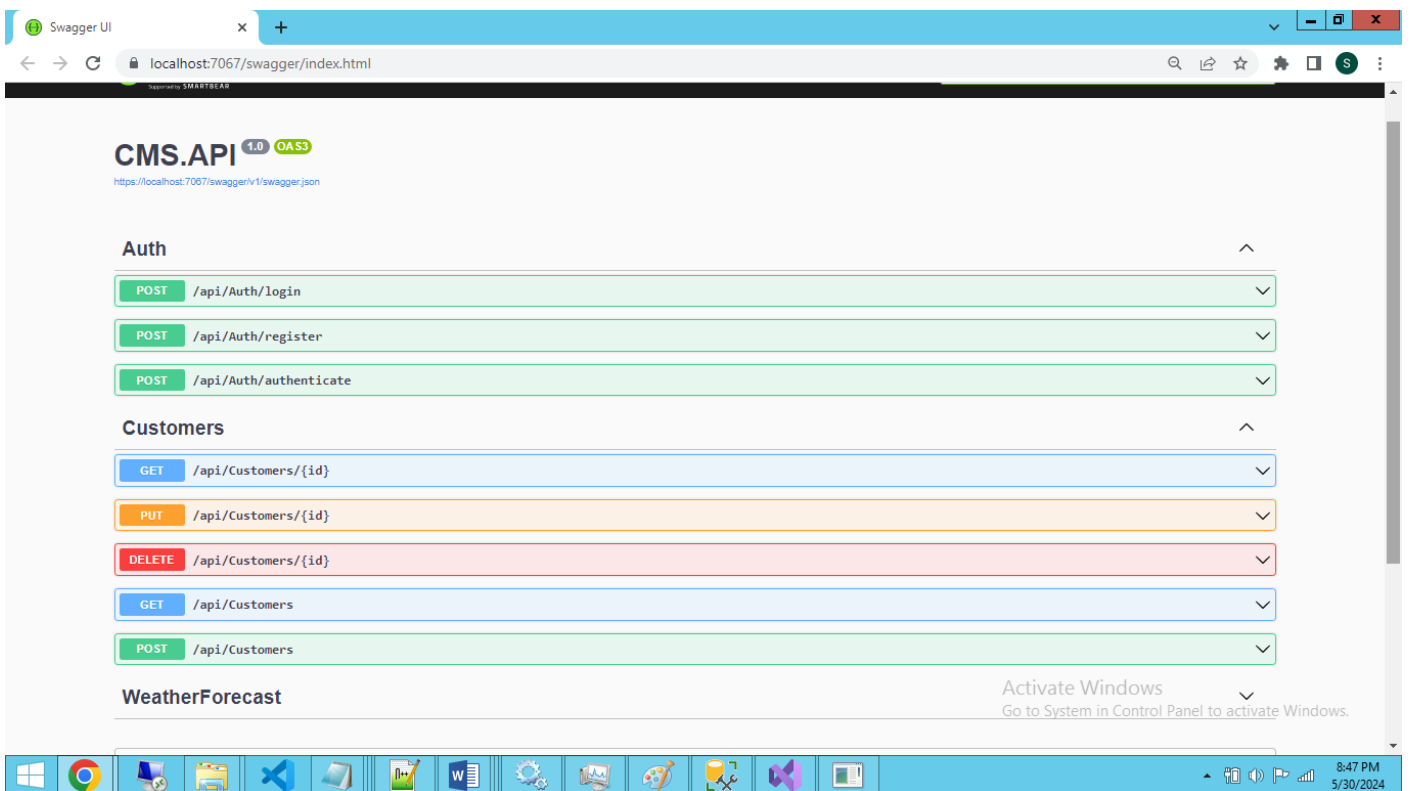
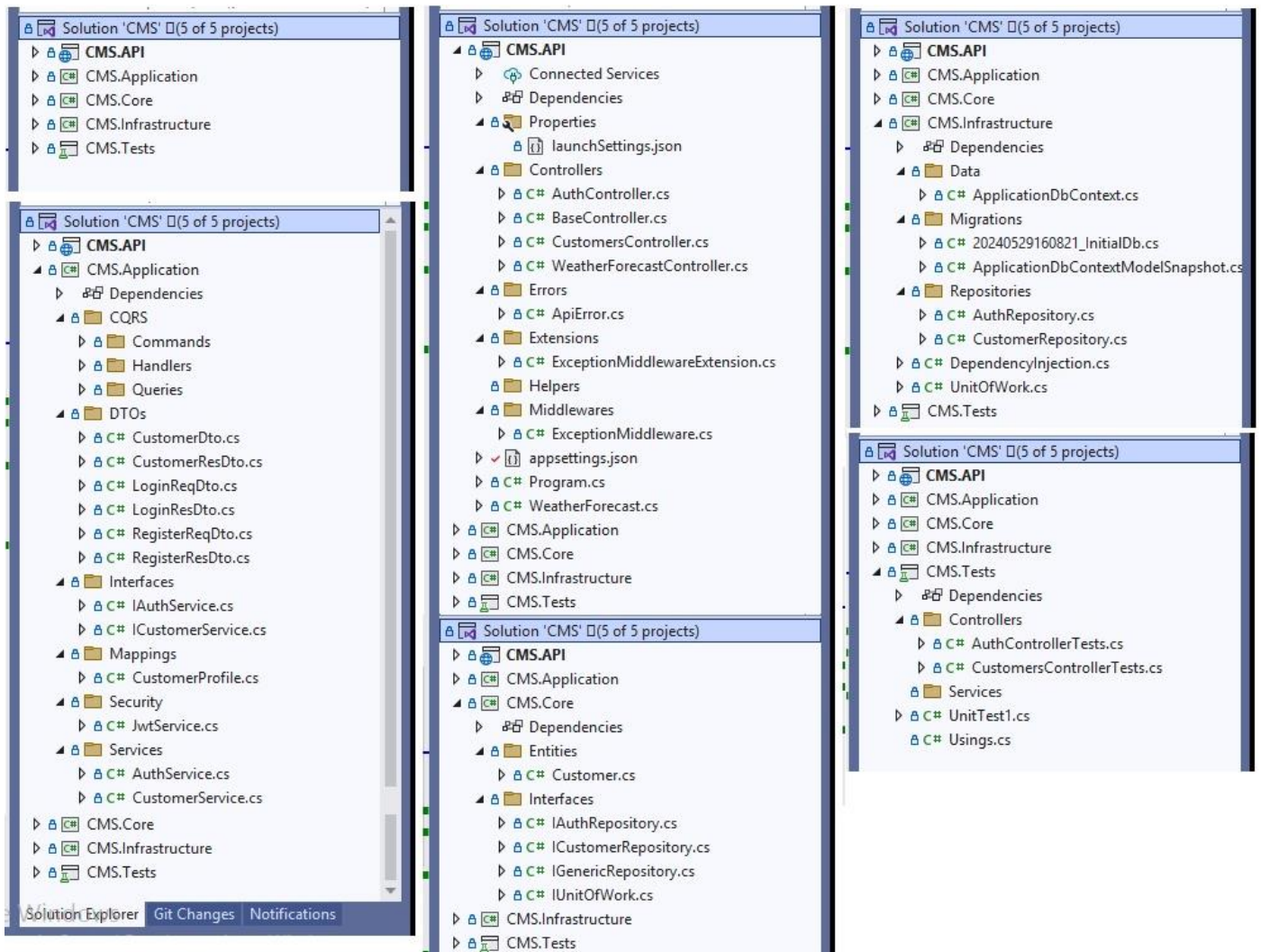
Backend: Dot Net Core, Web API, Entity Framework Core, SQL Server with Clear Architecture

✓ Backend API Summary

.NET Core Site URL: <https://localhost:7067/swagger/index.html>

Tech Stack: ASP.NET Core Web API + EF Core + SQL Server + JWT + Clean Architecture

Folder Structure: Clear Architecture

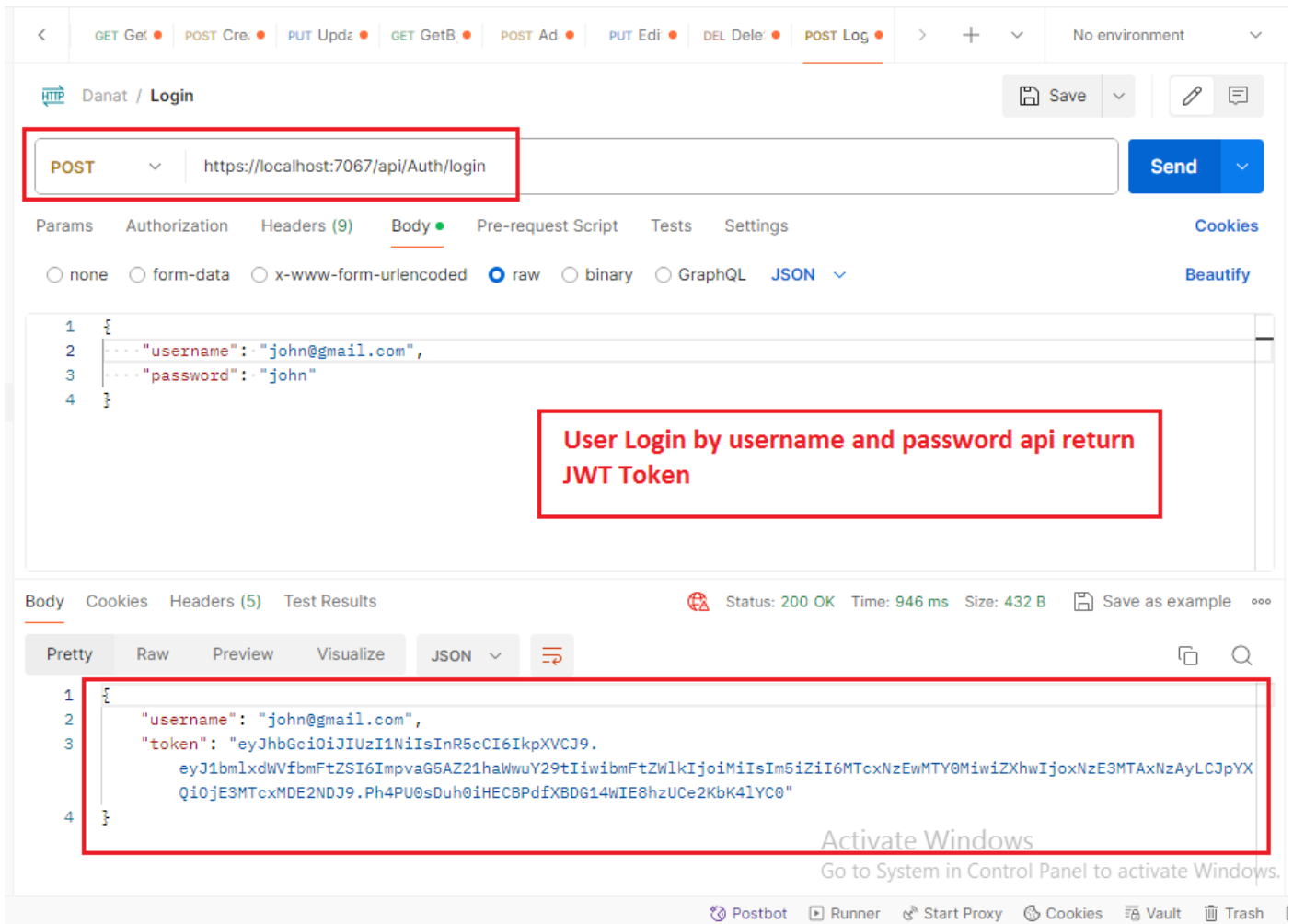


1. Authentication Endpoint

- ```
{
 "username": "john@gmail.com",
 "password": "john"
}
```

- The screenshot displays the Swagger UI for a REST API. The main section shows the **POST** endpoint `/api/Auth/login`. The request body is a JSON object: `{ "username": "john.doe@example.com", "password": "John@123" }`. The response body is a JSON object: `{ "username": "john.doe@example.com", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bm1xdWVmbmFtZSI6ImprVCJ9.eyJ1bm1xdWVmbmFtZSI6ImprVCJ9.eyJ1bm1xdWVmbmFtZSI6ImprVCJ9" }`. The **Execute** button is highlighted. Below the response body, the **Response headers** section shows: `access-control-allow-origin: *`, `content-length: 263`, `content-type: application/json; charset=utf-8`, `date: Thu, 15 May 2025 14:41:53 GMT`, and `server: Kestrel`. At the bottom, a table lists the responses: 

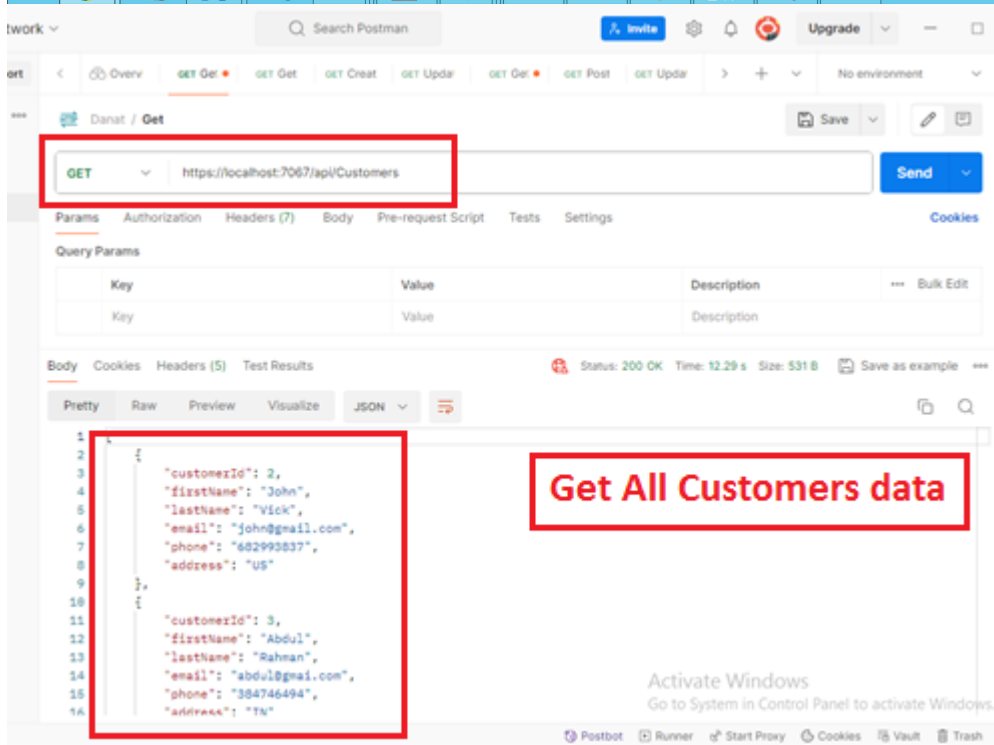
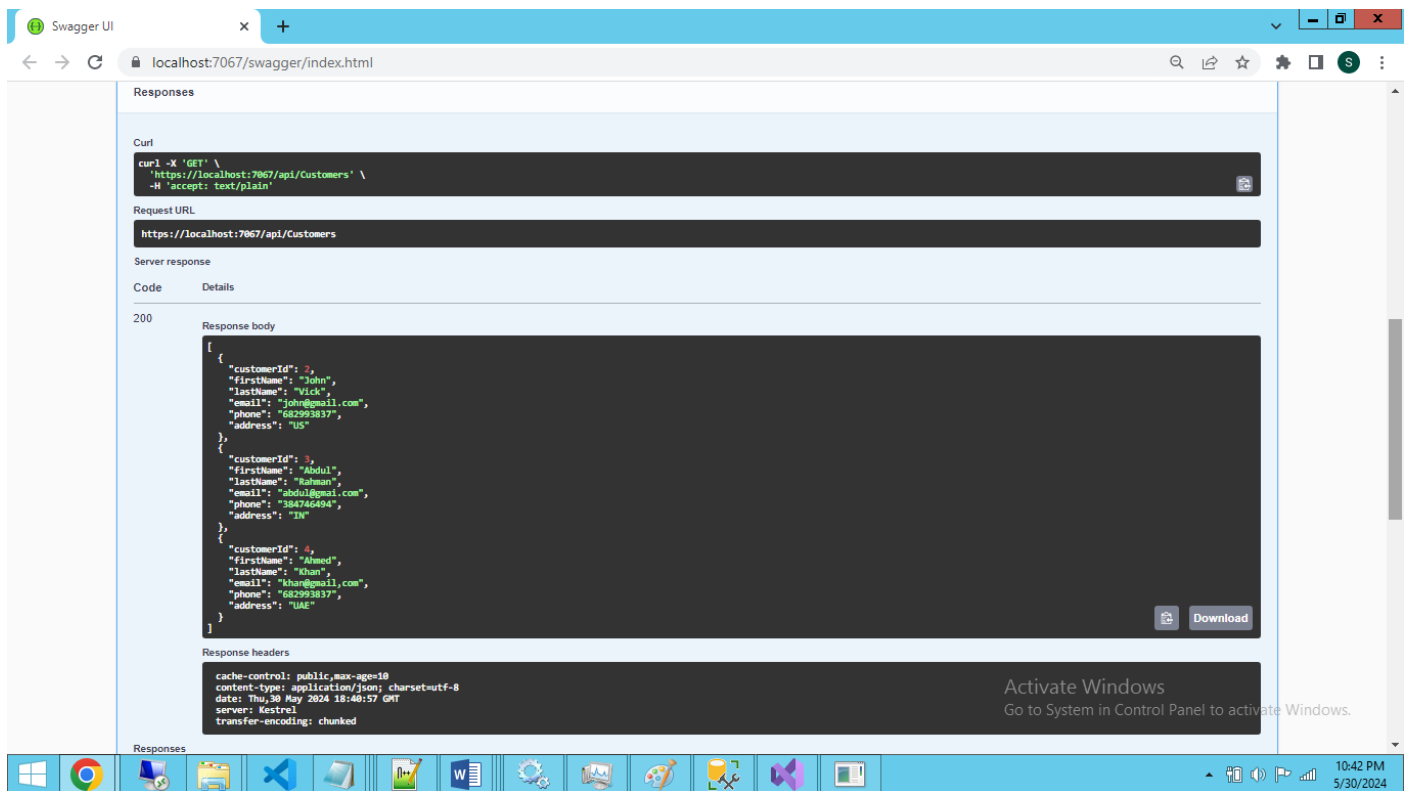
| Code | Description | Links    |
|------|-------------|----------|
| 200  | Success     | No links |



## 2. Get All Customers

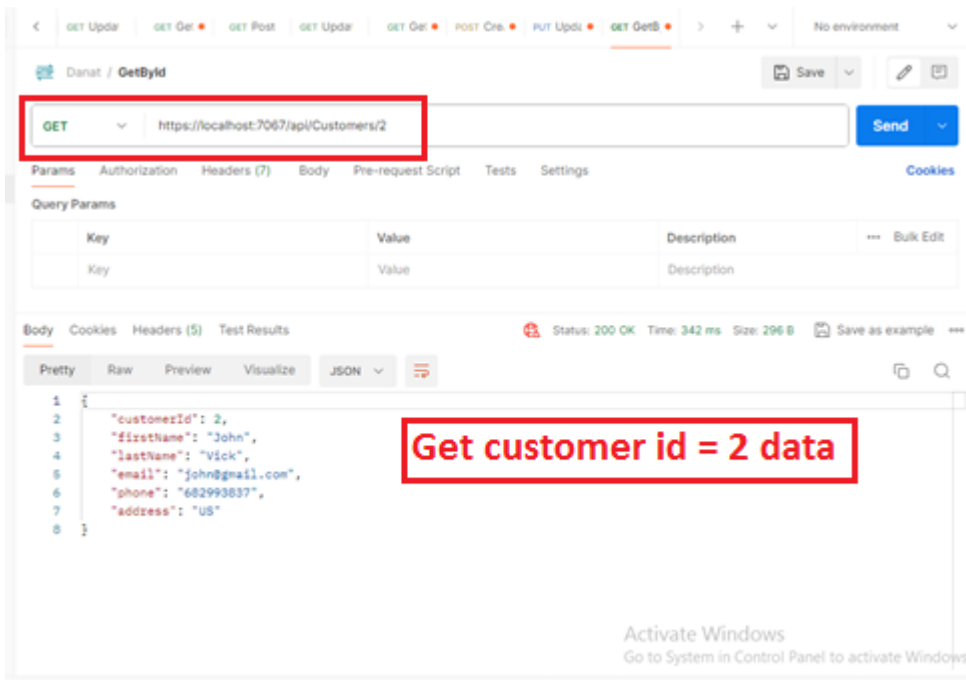
Authorization: Bearer <token>

```
[
 {
 "customerId": 1,
 "firstName": "John",
 "lastName": "Doe",
 "email": "john.doe@gmail.com",
 "phone": "1234567890",
 "address": "USA"
 },
 ...
]
```



### Q 3. Get Customer by ID

- **Purpose:** Retrieve a specific customer
- **URL:** GET `https://localhost:7067/api/Customers/{id}`
- **Example:** GET `/api/Customers/2`
- **Headers:** Requires JWT token



## + 4. Add New Customer

- **Purpose:** Add a new customer
- **URL:** POST https://localhost:7067/api/Customers
- **Payload:**

```
{ "firstName": "Martin", "lastName": "Luther", "email": "martin@gmail.com", "phone": "682993456", "address": "USA"}
```

- **Validation:**
  - Required: firstName, lastName, email, phone, address
  - Unique: email



The first screenshot shows a POST request to `https://localhost:7067/api/Customers` with a JSON body containing customer details. The response status is 201 Created. The second screenshot shows a GET request to the same URL, returning a list of customers in JSON format. A red box highlights the newly added customer in the response.

**POST Request:**

```
POST https://localhost:7067/api/Customers
{
 "firstName": "Martin",
 "lastName": "Luther",
 "email": "martin@gmail.com",
 "phone": "682993456",
 "address": "USA"
}
```

**POST Response:**

```
201 Created
{
 "customerId": 6,
 "firstName": "Martin",
 "lastName": "Luther",
 "email": "martin@gmail.com",
 "phone": "682993456",
 "address": "USA"
}
```

**GET Request:**

```
GET https://localhost:7067/api/Customers
```

**GET Response:**

```
200 OK
[
 {
 "customerId": 4,
 "firstName": "Ahmed",
 "lastName": "Khan",
 "email": "khan@gmail.com",
 "phone": "682993837",
 "address": "UAE"
 },
 {
 "customerId": 7,
 "firstName": "Martin",
 "lastName": "Luther",
 "email": "martin@gmail.com",
 "phone": "682993456",
 "address": "USA"
 }
]
```

## 5. Edit Existing Customer

- **Purpose:** Update customer details
- **URL:** PUT `https://localhost:7067/api/Customers/{id}`
- **Example:** PUT `/api/Customers/7`
- **Payload:**

```
{
 "customerId": "7",
 "firstName": "Martin",
 "lastName": "King",
 "email": "king@gmail.com",
 "phone": "682993456",
 "address": "UK"
}
```

- **Note:**
  - Validates ID exists
  - Ensures email uniqueness

The screenshot displays a REST client interface with the following details:

- Method:** PUT
- URL:** https://localhost:7067/api/Customers/7
- Body:** A JSON object representing a customer:

```
{ "customerId": "7", "firstName": "Martin", "lastName": "King", "email": "king@gmail.com", "phone": "682993456", "address": "UK"}
```
- Status:** 204 No Content
- Time:** 93 ms
- Size:** 116 B

The interface also shows tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Body tab is active, and the JSON is formatted. The status bar at the bottom indicates the request was successful with a 204 status code.

Postman interface showing a GET request to `https://localhost:7067/api/Customers`. The response is a JSON array of customer data. A red box highlights the second customer object, which has been edited to have `customerid: 7`.

GET `https://localhost:7067/api/Customers` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (5) Test Results Status: 200 OK Time: 27 ms Size: 647 B Save as example

Pretty Raw Preview Visualize JSON

```
19 {
20 "customerid": 4,
21 "firstName": "Ahmed",
22 "lastName": "Khan",
23 "email": "khan@gmail.com",
24 "phone": "682993837",
25 "address": "UAE"
26 },
27 {
28 "customerid": 7,
29 "firstName": "Martin",
30 "lastName": "King",
31 "email": "king@gmail.com",
32 "phone": "682993456",
33 "address": "UK"
34 }
]
```

Edited CustomerId 7 data

Activate Windows  
Go to System in Control Panel to activate Windows.

Postbot Runner Start Proxy Cookies Vault Trash

## ✖ 6. Delete Customer

- **Purpose:** Remove a customer by ID
- **URL:** DELETE `https://localhost:7067/api/Customers/{id}`
- **Example:** DELETE `/api/Customers/7`

DELETE https://localhost:7067/api/Customers/7

Status: 204 No Content Time: 63 ms Size: 116 B Save as example

1

Deleted CustomerId = 7 data

## ✓ Clean Architecture Layers (Recommended)

Your architecture should be structured like:

```
/Core
- Entities (Customer.cs)
- Interfaces (ICustomerRepository.cs)
- DTOs (CustomerDto.cs)
- Validation (FluentValidation / DataAnnotations)

/Application
- Services (CustomerService.cs)
- UseCases (e.g., GetAllCustomersQuery.cs)

/Infrastructure
- Data (AppDbContext.cs)
- Repositories (CustomerRepository.cs)

/API
- Controllers (CustomerController.cs, AuthController.cs)
- Middleware (JWT, ExceptionHandling)
- Startup Configuration
```

## 🔒 Security

- Use [Authorize] attribute on controller or action level
- Validate JWT token in each request
- Return 401 Unauthorized if the token is missing or invalid

---

# Frontend ↔ Backend Connection

| Frontend Route       | Backend API Endpoint       |
|----------------------|----------------------------|
| /register            | POST /api/Customers        |
| /login               | GET /api/Auth/login        |
| /customer-list       | GET /api/Customers         |
| Inline Edit Customer | PUT /api/Customers/{id}    |
| Delete Button Click  | DELETE /api/Customers/{id} |

---