**Department Of Veterans**

Project - User Demographic Management

1. **Technology requirement**

Backend: ASP.NET 4.8 or .Net Core 3.1 +

Front End: JS framework (Angular/React) or MVC (Razor)

Code first or Database first with all scripts required for set up

API – document with Swagger, endpoint

User Case

1. Login with username and password
2. Register new user
3. User can be added by manual or loading from <https://randomser.me/>
4. User Management (add/edit/delete)
5. **Technical approach**

With that information, I came up with below solution and technical design

Setting up application

Backend (Rest API)

* ASP.NET Core Web API 6 – **security with token base**
* Clean architect principles (easy to **maintenance** and **scalable** if we can build from scratch)
* Code first
* Can be implemented parallel with FE team

Client

* Built using MVC (ASP.NET Core 6) – can be started when having **model** from BE team.

Class libraries

* .NET 6 class libraries
* AutoMapper library (mapping from one type to another type)
* Fluent Validation (Nuget package – Adding a custom Validator)
* EntityFramework Core
* HttpClientFactory
* FE Code is generated from NSwagStudio

Design Pattern

* Using Meditor pattern (MediatR in .NET)
* Using Dependency of Inversion pattern
* Using CQRS pattern (Command – Query Responsibility Segregation)
* API Gateway pattern
* Single Responsibility (SOLID) – Separate create/update/query command
* DRY (export, email function, HttpClientFactory feature from webUI)

1. **Project Info**

The full source code can be download from GitHub as below:

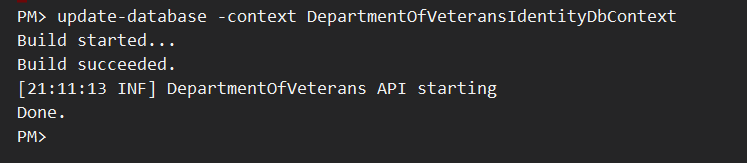
<https://github.com/origami3011/DepartmentOfVeterans.git>

In other to run the application, we need to following below

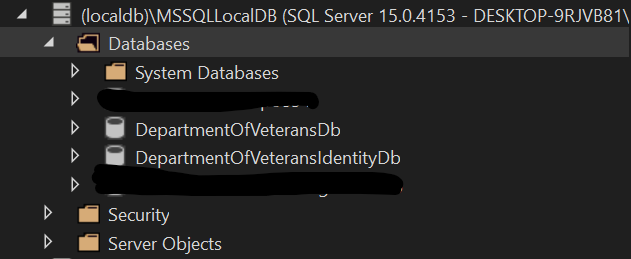
Run update-database at **Package Manager Console** for both database DepartmentOfVeteransDb and DepartmentOfVeteransIdentityDb

update-database -context DepartmentOfVeteransDbContext

update-database -context DepartmentOfVeteransIdentityDbContext

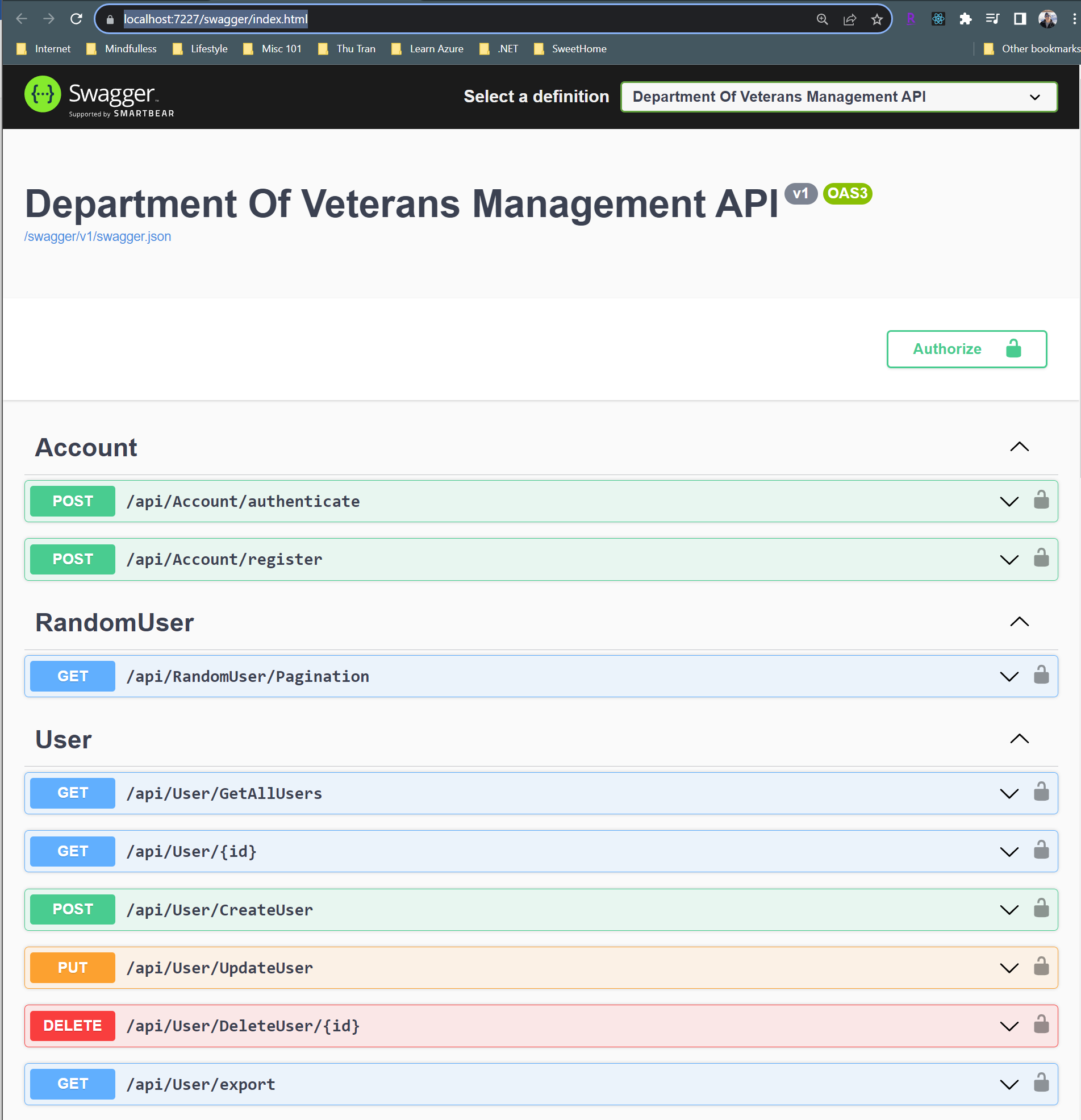


If there is no issue occurs, we will see two databases like this from SQL Server Object Explorer

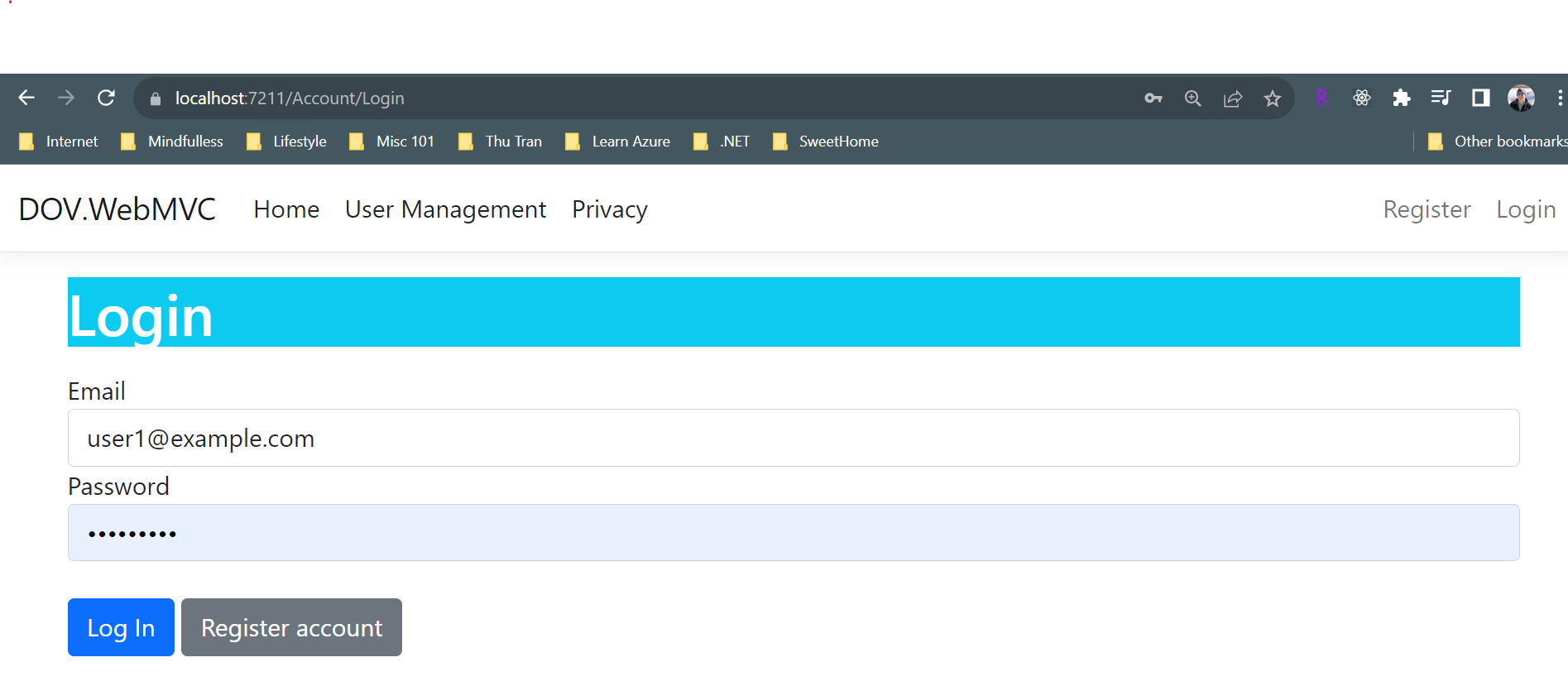


The application requires API service start first.

Api - Right click on DepartmentOfVeterans.Api > View > View in browser (chrome)



App - Right click on DepartmentOfVeterans.WebMVC > View > View in browser (chrome)



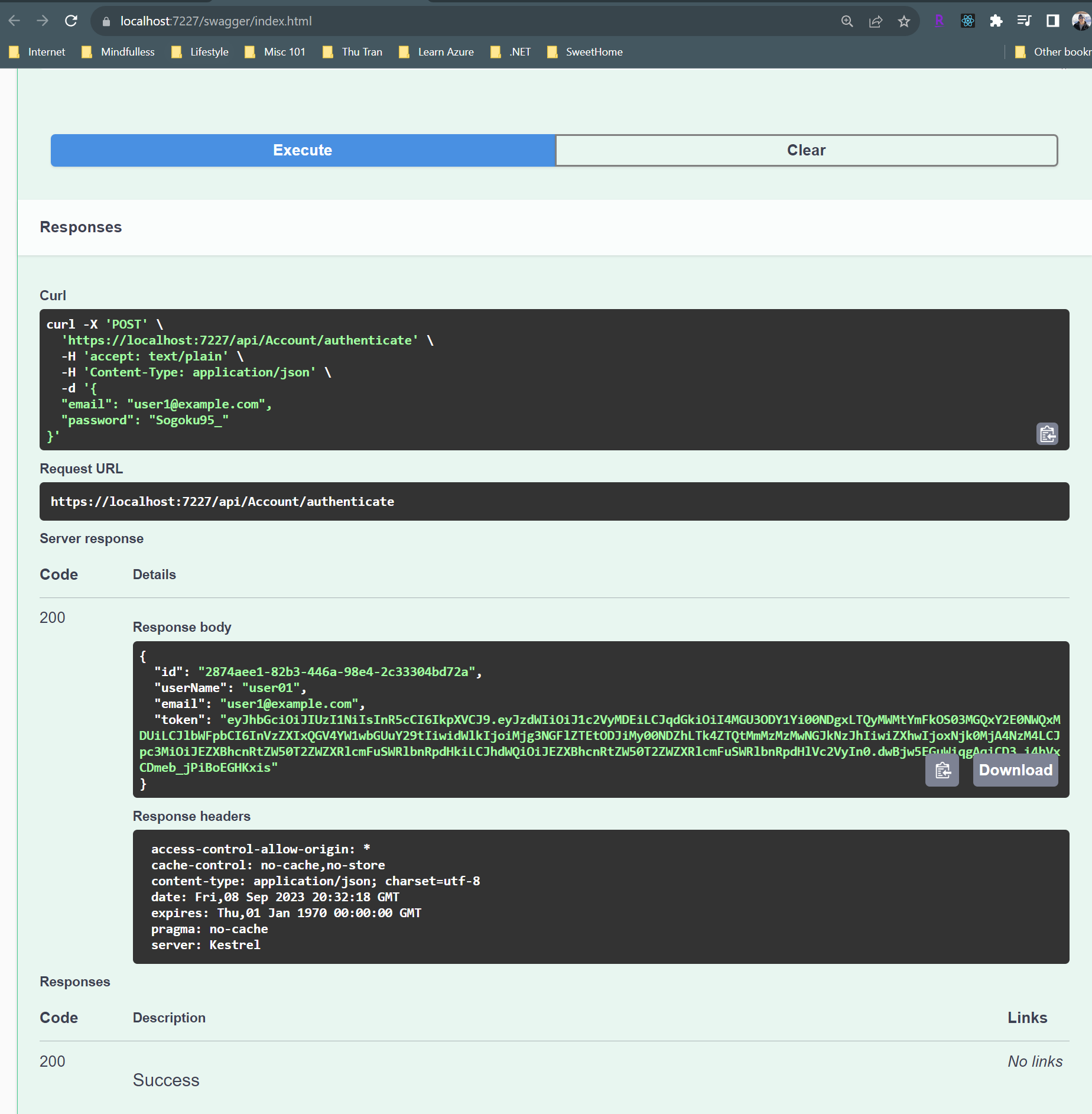
APIs can be tested via Swagger or Postman before release to another team to implement UI.

**Swagger**

Token Generation: First, a token needs to be generated. This token is typically obtained through an authentication process, such as logging in with valid user credentials.

Calling a Sub-API: Once you have the token, you can use it to access a specific sub-API related to user demographics. A sub-API is typically a part of a larger API that deals with a specific subset of data or functionality.

Bearer Token Authentication: When making API calls to the User Demographic sub-API, you must include the bearer token in the request headers. Bearer token authentication is a common method of ensuring that only authorized users can access the API. The API server will verify the token to ensure the request is legitimate

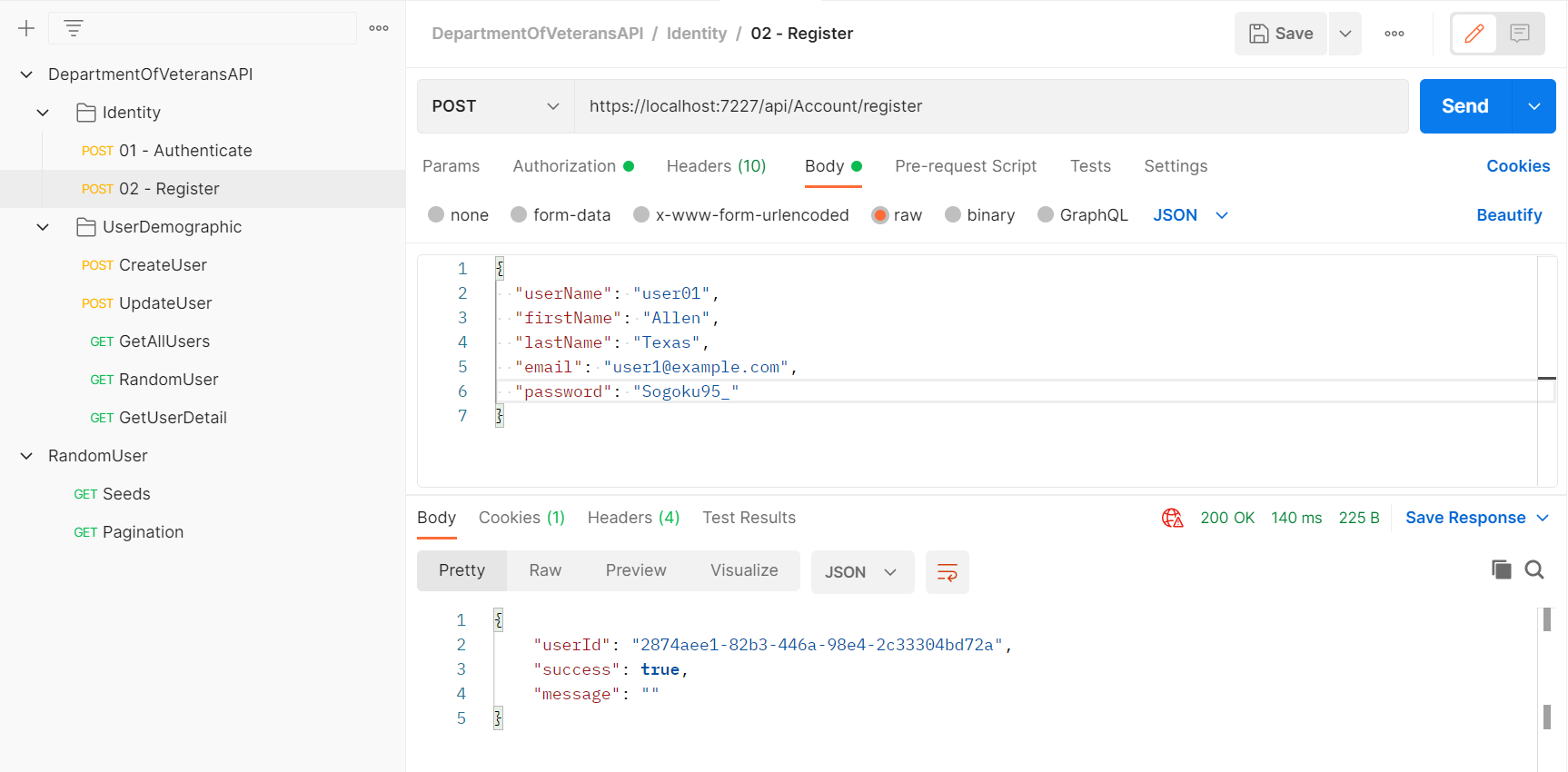


**Postman**

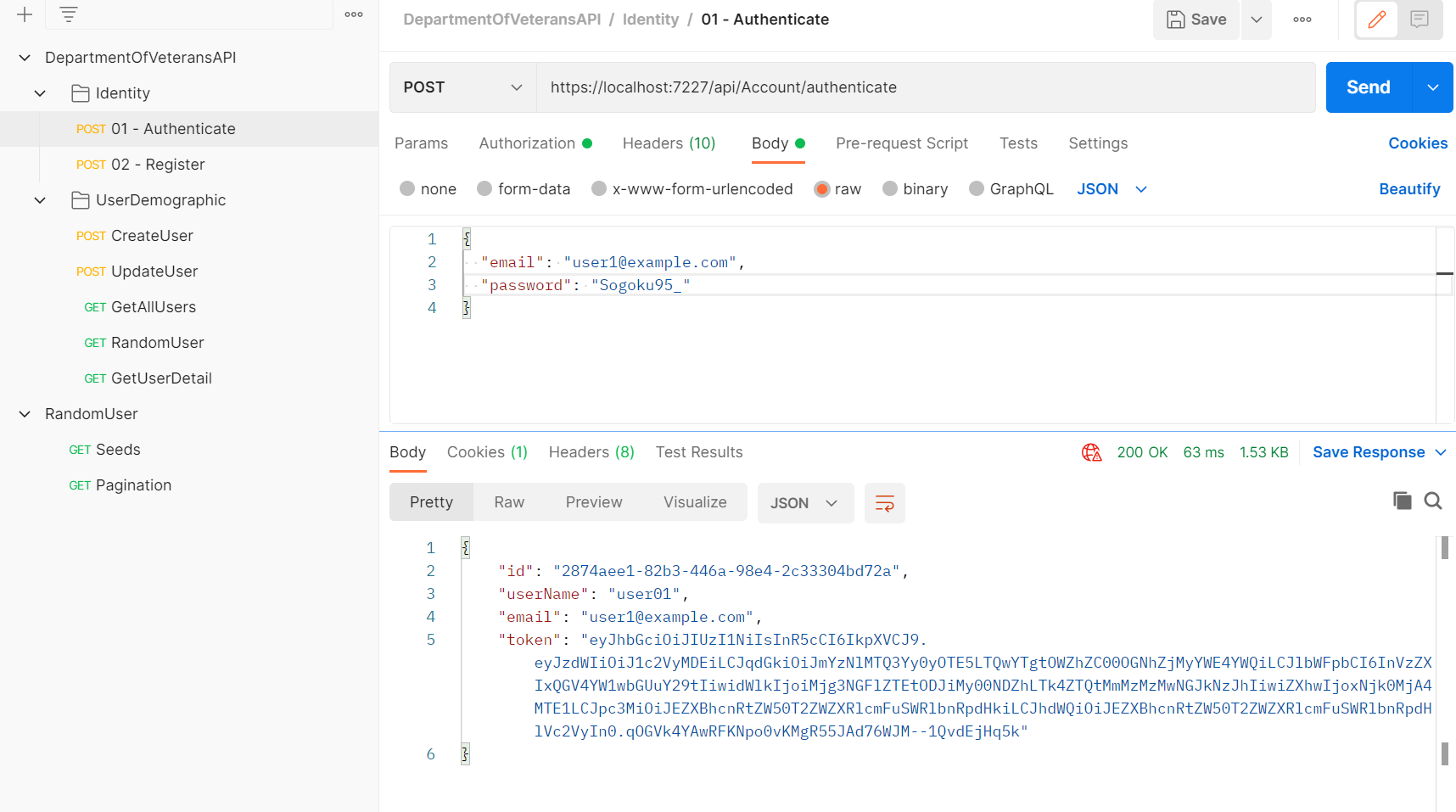
You can download the collection and verify all api method beside Swagger UI. Follow the screenshot for

sample. Don’t forget to create **{{token}}** as variable to make the call easier.

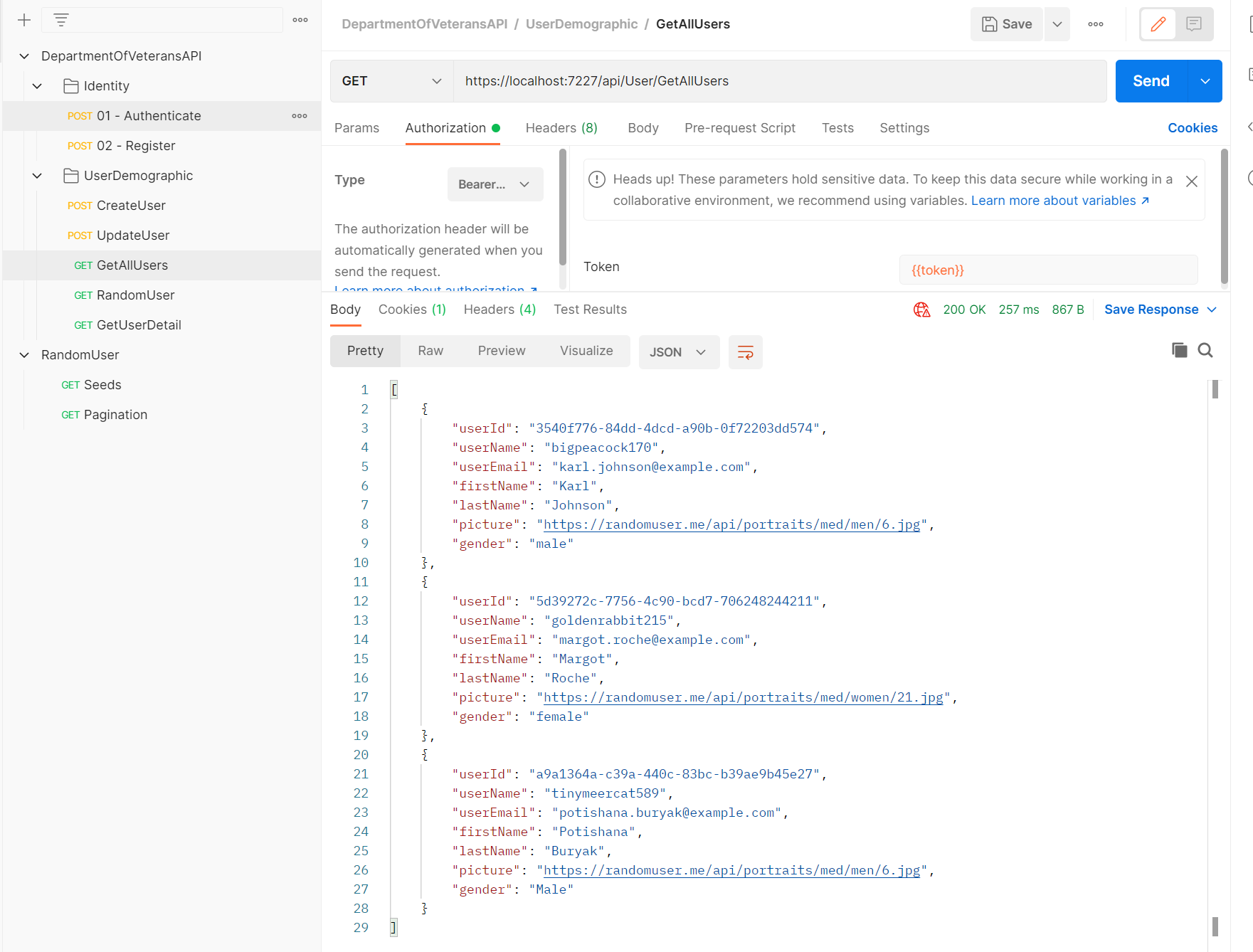
Register login account



Authenticate api



Assign token to {{token}} variable



Conclusion

* The application can be enhanced to better serve end users by incorporating additional features
  + Apply new modern layout (sample <https://themeforest.net/item/sliced-aspnet-mvc-5-tailwind-css-admin-dashboard-template/47450107>)
  + Add forgot password, user list, role base or enable F2A
  + Support upload file from client, add more user property
  + Export, email function should be function for next sprint
  + App may have some bugs (since doer could not be a good tester 😊)

Thank you and please comment and suggest to make it works better!!!