# SWAGGER

TABLE OF CONTENTS

Introduction 3

API Specification 3

SwashBuckle 4

Swagger Implementation 4

Installing the required packages 4

Configuring the Services 5

Configurations of Middleware 6

Configuring the security for Response 8

Services for Authorization 9

Adding Document Definitions in Endpoints 10

XML Documentation Comments 15

Configuring comments in Swagger 17

Conclusion 18

S.AMIRTHA LAKSHMI

# Introduction

Swagger

OPENAPI Specification

SwashBuckle

Swagger is an open-source set of guidelines, requirements, and resources which is used to create and describe RESTful APIs. Developers can write interactive, machine and human-readable API documentation using the Swagger framework, which helps in documentation of the HTTP Requests which is helpful for the users to visualize the response. It’s comprehensible for developers and non-developers provides the friendly UI which is designed as API. The major swagger tools includes,

* [Swagger Editor](https://editor.swagger.io/?_ga=2.210342934.2091967461.1669550672-149303343.1667975198) – browser-based editor where you can write OpenAPI definitions.
* [Swagger UI](https://github.com/swagger-api/swagger-ui) – renders OpenAPI definitions as interactive documentation.
* [Swagger Codegen](https://github.com/swagger-api/swagger-codegen) – generates server stubs and client libraries from an OpenAPI definition.
* [Swagger Editor Next (beta)](https://editor-next.swagger.io/?_ga=2.210342934.2091967461.1669550672-149303343.1667975198) – browser-based editor where you can write and review OpenAPI and AsyncAPI definitions.
* [Swagger Core](https://github.com/swagger-api/swagger-core)– Java-related libraries for creating, consuming, and working with OpenAPI definitions.
* [Swagger Parser](https://github.com/swagger-api/swagger-parser) – standalone library for parsing OpenAPI definitions
* [Swagger APIDom](https://github.com/swagger-api/apidom) – provides a single, unifying structure for describing APIs across various description languages and serialization formats.

Features

* Maintains constant synchronization between the API specification for both the client and server.
* Allows the user to generate REST API and verify the requests dynamically
* The responses are displayed in the format of JSON/Yaml

# API Specification

Open API is a collection of guidelines that describes how to display an API request. Without access to source code, additional documentation, or network traffic analysis, humans and computers can both discover and comprehend a service's capabilities by OpenAPI Specification (OAS). This specifies a standard, programming language-independent interface description for HTTP APIs. A consumer can comprehend and interact with a remote service when it is correctly specified using OpenAPI and only requires a small amount of implementation logic.

The XML and attribute annotations included in the controllers and models are the foundation of the document. It serves as the central component of the OpenAPI flow and powers tools like Swagger UI. It has defined as openapi.json as default format which is shown as below, which provides the path, information about the requests.



The swagger.json files explain about how to describe the Restful API’s in accordance with the specification.It contains about the path of the URL’s, request body, content, schemas, parameters, responses.

SwashBuckle

A nuget package called Swashbuckle contains Swagger tools for describing APIs created on the Microsoft.NET platform.

Diagram

Description automatically generated

Swagger Implementation

A screenshot of a computer

Description automatically generated

While creating a new project in Visual Studio as ASP.NET Core Web API the application itself provides the OpenAPI Support,so the application itself provides the Swagger as inbuilt feature along with the middleware,and services required to execute the swagger.For other Web application projects we need to do the further process,

Installing the required Packages

If the chosen project other than Web API we need to add and install swagger to get the swagger UI to display the restful API.

Graphical user interface, text, application, website

Description automatically generated

The below image shows the folder structure of Web App (Model-View-Controller) which has separate folders created for controllers,models,views

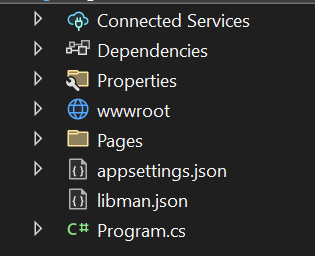
Text

Description automatically generated

Graphical user interface, website

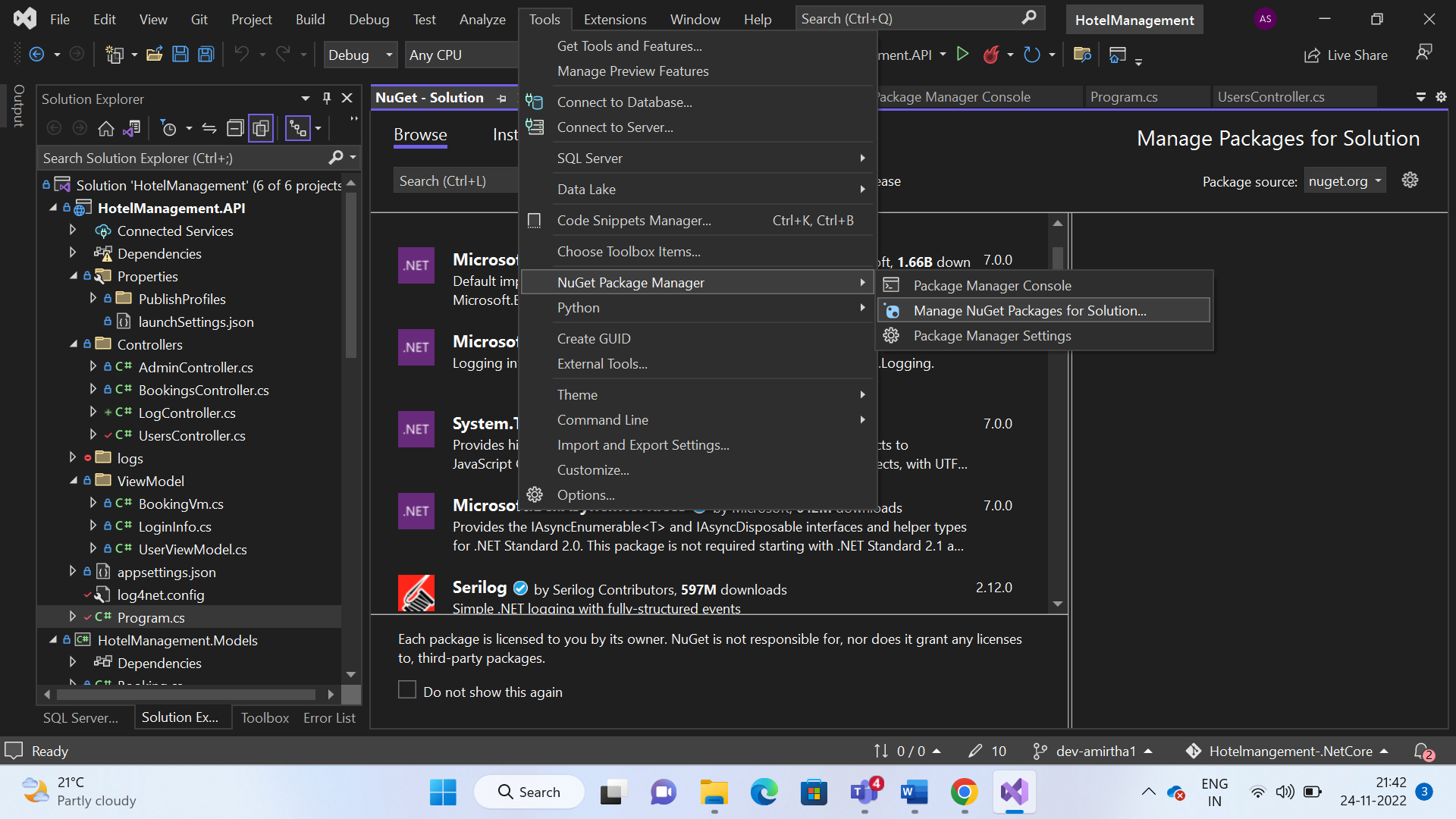
Description automatically generated

The below images is the folder structure of WebApp which doesn’t have the controllers so we need to create the controllers to act on swagger



**STEP-1:** Installation of SwashBuckle

* To work in the swagger, swashbuckle is the essential extension which is needed to be present in .NET platform.
* To install it go to Tools->NuGetPackage Manager->Manage NuGetPackagesForSolution
* Search for the necessary package and install it.



Text

Description automatically generated with medium confidence

Configuration of Services

**STEP-2:** Configuring Swagger

The services which are required for configuring swagger should be added before building the solution ,

AddEndpointaApiExplorer is used for API explorer support (ie) gets the API methods without any codes.

SwaggerGen is used to builds the swagger document objects from routes, controller, models.

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

Configurations of Middleware

**Step-3:** Configuring the Swagger UI Middleware

In order to start working in the swagger we need to create a Middleware in the program.cs file, where middleware is a software which are built inside a pipeline used to handle the response and requests.

Swagger is used to add swagger middleware.

SwaggerUI helps to create Application and HTML page for the API’s.

if (app.Environment.IsDevelopment())

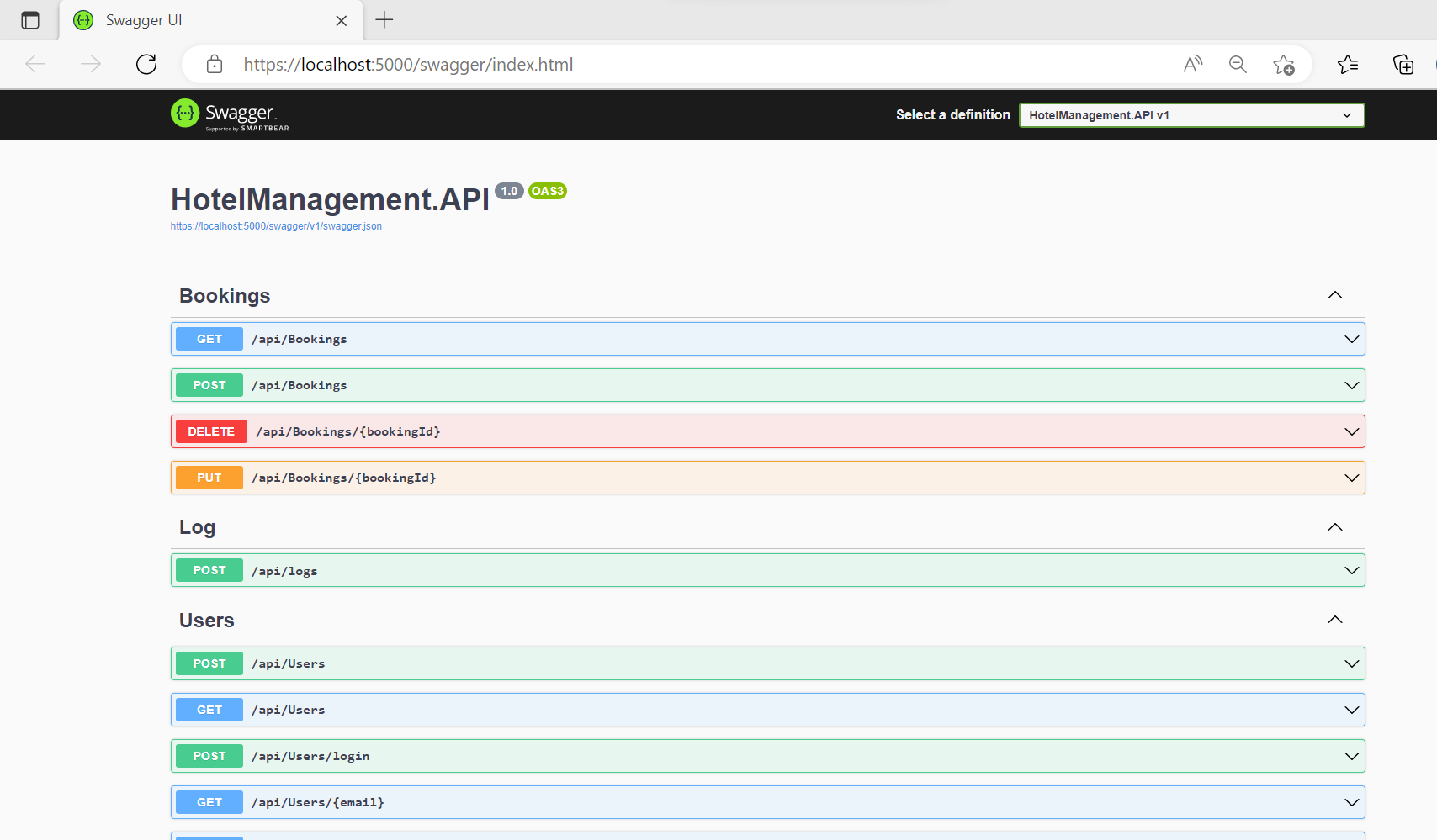
{

app.UseSwagger();

app.UseSwaggerUI();

}

Swagger UI



After adding the requirements in program.cs file the implementation displays the HTML page of Swagger with the documented UI along with the controller and services we added based on the Http requests the user given

A picture containing application

Description automatically generated

While expanding each request the user need to click the try it out button and should add the necessary input asked by the swagger and then while clicking the execute button it provides the server response by the status code based on the response along with the details of output which is data from the server database.

Background pattern

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Configuring the Security of Response-JWT

After implementation of basic CRUD operations Swagger also checks the authorized requests. Authorization is a process by which a server determines if the client has permission to use a resource or access a file. This can be done by adding upon BEARER token in required requests and verify the request and provide the response.Here the below mentioned code was to include JWT Token, whereas the authorization can be done by other methods like OAuth2

Services for Authorization

builder.Services.AddSwaggerGen(option =>

{

option.SwaggerDoc("v1", new OpenApiInfo { Title = "Demo API", Version = "v1" });

option.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Please enter a valid token",

Name = "Authorization",

Type = SecuritySchemeType.Http,

BearerFormat = "JWT",

Scheme = "Bearer"

});

option.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type=ReferenceType.SecurityScheme,

Id="Bearer"

}

},

new string[]{}

}

});

});

Adding the services of Authorization then,

**Step-1:** Requests will be added with a symbol of lock which defines the authentication of the requests.

Background pattern

Description automatically generated with medium confidence

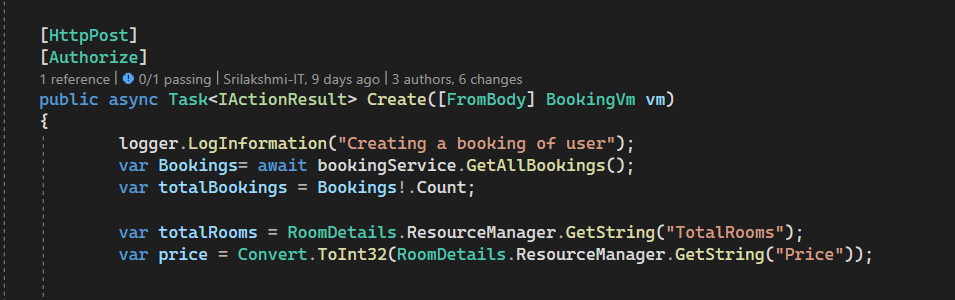
**Step-2:** The user should add the Token which is produced by the JWT from the server



Graphical user interface, text, application, chat or text message

Description automatically generated

**Step-3:** Authorize keyword should be mentioned above each request which are needed to be authenticated.



**Step-4:** After Authorization with JWT Token the request will be authenticated automatically so the users should provide the token and need to execute the request as the same mentioned above

Graphical user interface, application

Description automatically generated

Adding Document Definitions in Endpoints

These details can be optional, but it would be preferably better to give the details about the tasks which will display underneath of the Json link

builder.Services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new OpenApiInfo

{

Version = "v1",

Title = "First API",

Description = "An ASP.NET Core Web API for managing ToDo items",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "Sample test",

Url = new Uri("https://example.com/contact")

},

License = new OpenApiLicense

{

Name = "Sample View",

Url = new Uri("https://example.com/license")

}

});

});

Graphical user interface, text, application

Description automatically generated

XML Documentation Comments

A screenshot of a computer

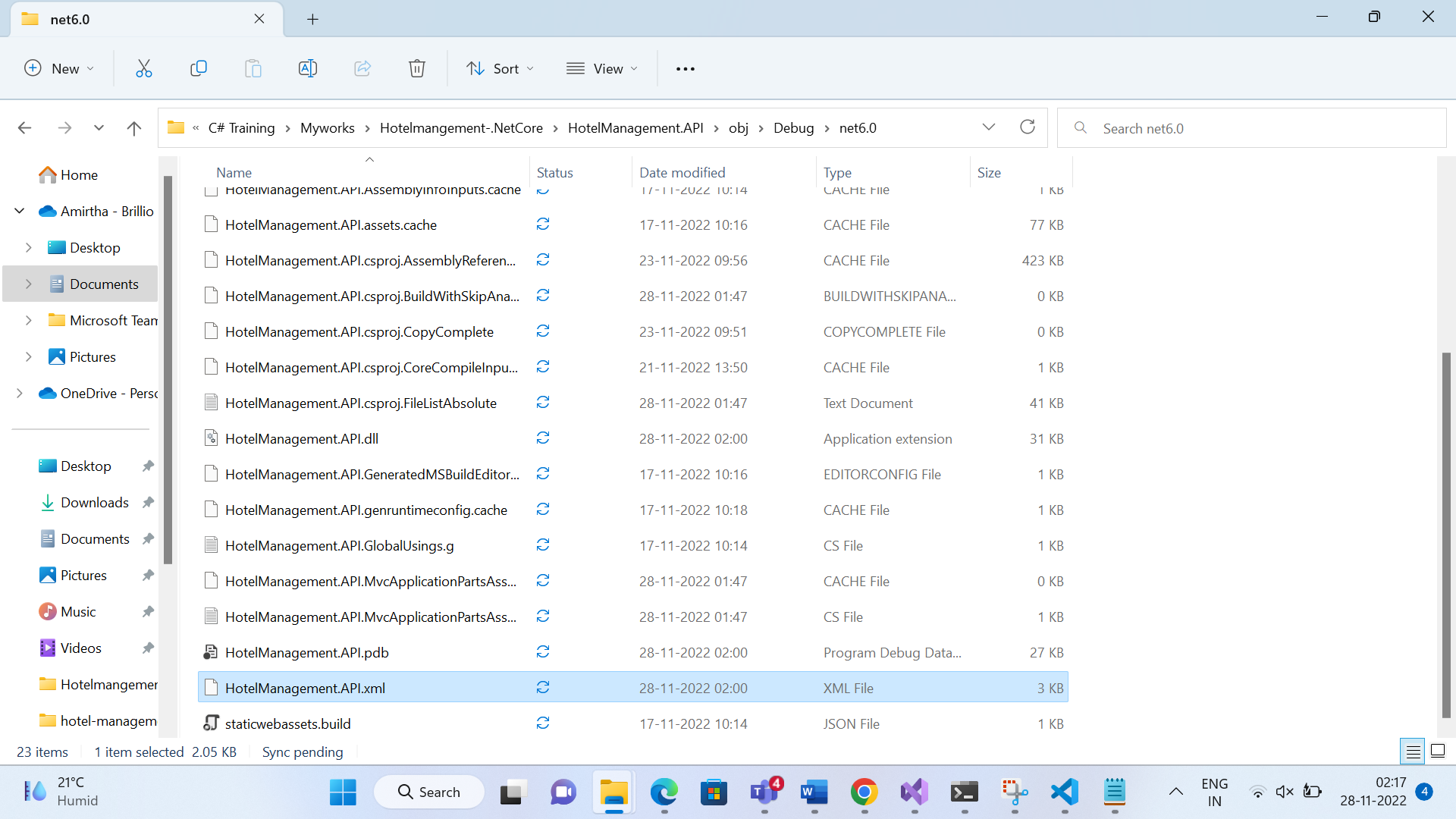
Description automatically generated with medium confidence

**Step-1:** Right Click the Project -> Properties -> to get the properties of our application

Text

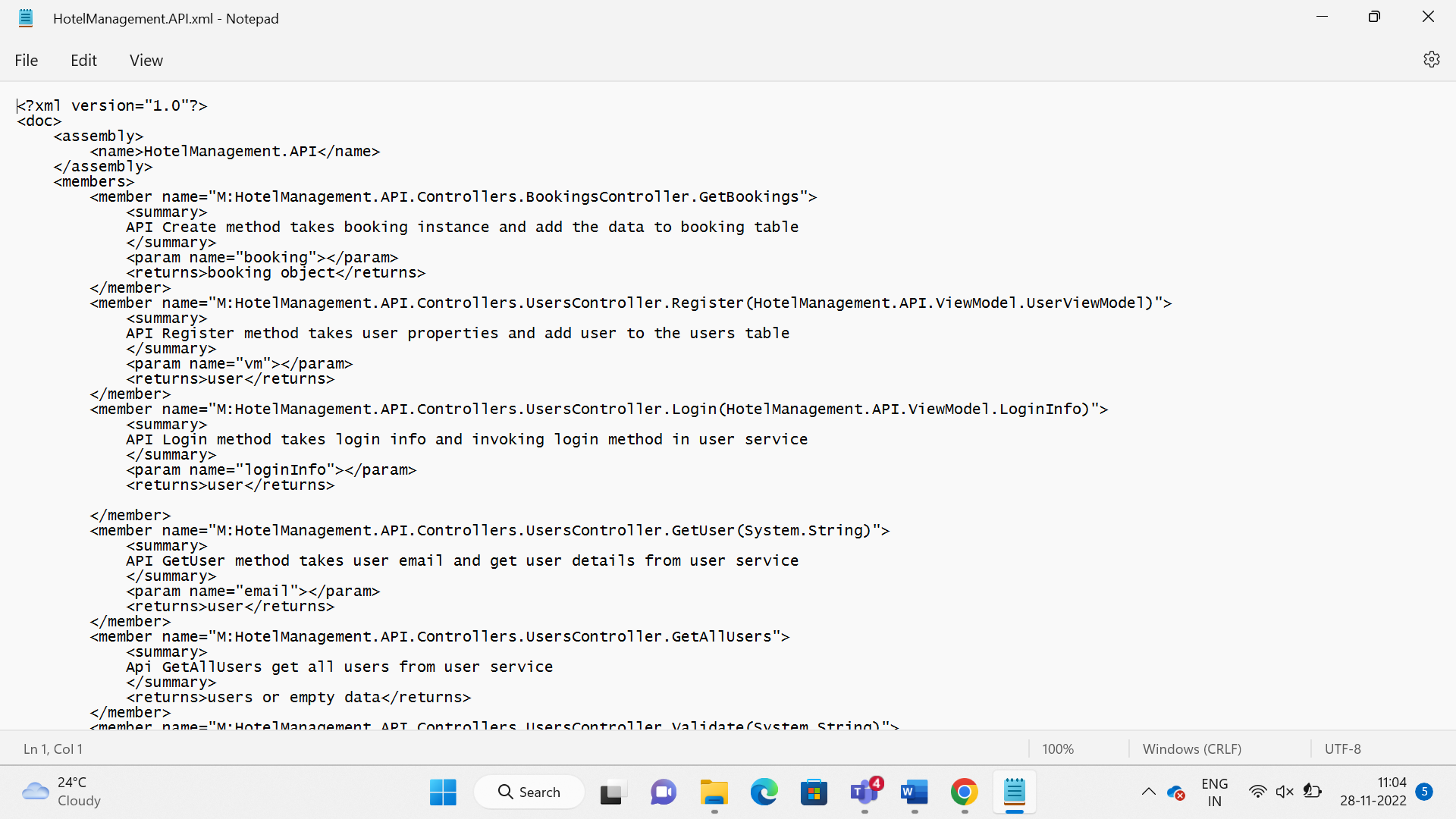
Description automatically generated

**Step-2:** Select Build -> output -> Check XML document file with the path which is shown as below

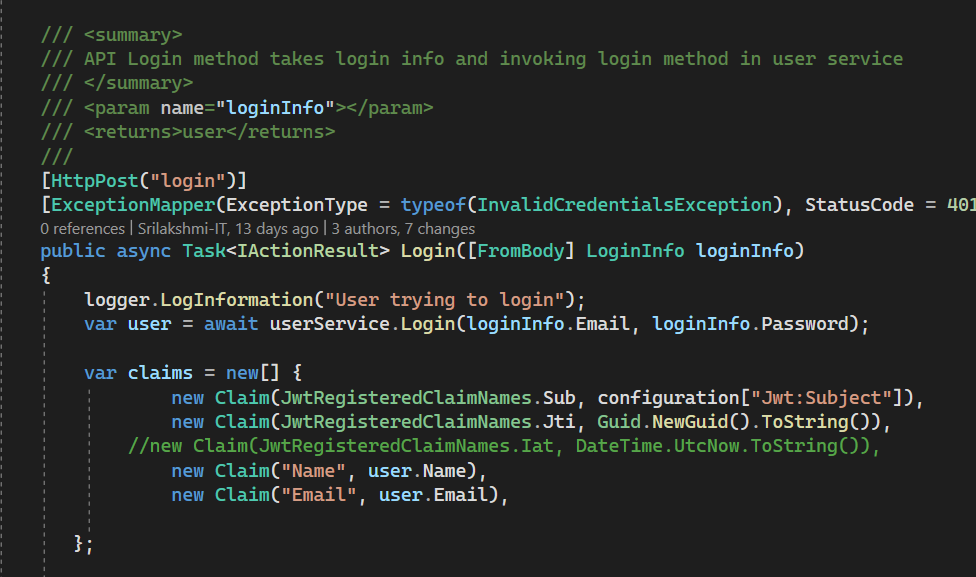


The XML file consists of all the comments that we have been added in our project file.

Output of XML File



This xml shows the comments of each method which are written in the xml summary headers, which can be added in the Swagger Documentation



The xml summary headers should be added as shown above. If we press the BackSlash(/) three times the summary header will be automatically created along with the param, return value of the particular specific method.

Configuring Comments in Swagger

builder.Services.AddSwaggerGen(options =>

{

var xmlfile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";

var xmlpath=Path.Combine(AppContext.BaseDirectory,xmlfile);

options.IncludeXmlComments(xmlpath);

});

The xmlpath need to be added in the AddSwaggerGen services to get the summary header in the Swagger Documentation

Output of Swagger

After performing the steps mentioned earlier the swagger will display the xml summary headers to the respective rest API’s

Text, application, email

Description automatically generated

Conclusion

In this documentation, it is explained about the Swagger API documentation, OpenAPI Specification. It provides a interface or the request and response handlers. Here the overview of how to get the request from the server and execute it using swagger tool is discussed along with the part of security configurations and XML Comments to be added in a step-by-step process.