

Nethan Developer guide

[Document subtitle]



May 2, 2024

Neurogaint Systems Pvt ltd

**1. Introduction**

**1.1** **Purpose**

This document has been updated to describe the expanded capabilities of the Nethan Solution Accelerator, which supports detailed generation for business logic, ORM entities, repositories, as specified in distinct JSON configuration files.

**1.2 Scope**

The Nethan Solution Accelerator now provides tools to generate a comprehensive back-end system architecture including business logic services, ORM layers, all aimed at reducing development time and enforcing consistency across large-scale projects.

**2. System Overview**

The extended capabilities of James allow for the generation of:

a. Service Layer (NethanService.JSON)  
b. ORM Entities (NethanORMModel.JSON)  
c. Nethan Controllers (Nethan.JSON)

d.DbContext(NethanDbContext.JSON)

These components are vital for modular, scalable, and maintainable back-end development in enterprise applications.

**3. Architecture**

**3.1 High-Level Architecture**

* **Nethan Solution Accelerator:** NethanCore application that reads various JSON configurations and generates corresponding .NET code.
* **Generated Components:**
* **Service Classes**
* **ORM Entities**
* **Controllers**

**3.2 Components**

**1. Configuration Loaders**

Parse distinct JSON files for different architectural requirements.

**2. Code Generators**

Generate C# classes for services, entities and Controllers.

**4. Process Flow**

**1. Configuration Specification:** Developers specify the architectural components in corresponding JSON files.

**2. Code Generation:**

* The Nethan Solution Accelerator reads the JSON configurations.
* Generates C# code for specified components, including business logic layers, ORM configurations, and RESTful controllers.

**3. Compilation and Deployment:**

* The generated code is compiled.
* Deployed within an Asp. Netcore application Environment.

**4. Runtime:**

* The Asp .NET application serves the business logic through API endpoints managed by generated controllers, accesses data through database and utilizes service classes for business operations.

**5. Technologies Used**

* **C#:** Primary programming language.
* **Entity Framework Core:** For ORM Implementation.
* **Asp. Netcore MVC/Web API:** For Restful API development.
* **Newtonsoft.Json:** For Json parsing.
* **Visual Studio:** Recommended development environments.

**6. Code generator link:**

Download the code from Git hub using below link.

https://github.com/mounikapotla911/NGS/blob/main/NethanCore.zip

**Prerequisites:**

Visual Studio 2022 (Community version)

.Net 8 SDK

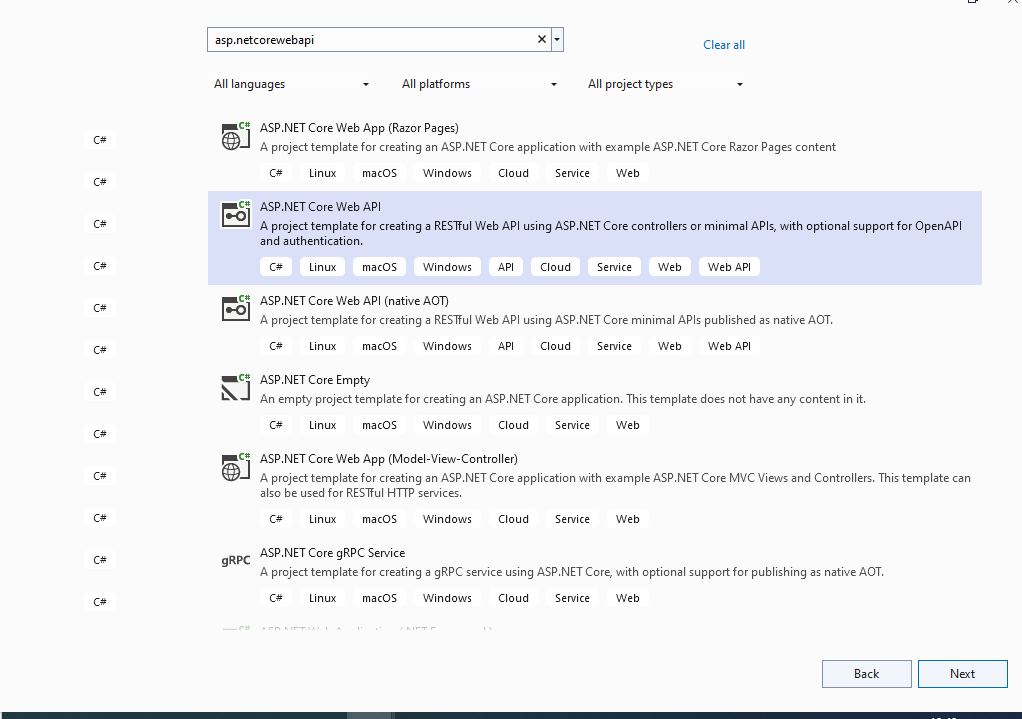
SQL server management Studio 2019

**7. Steps:**

First, Download the Nethancore.zip file.

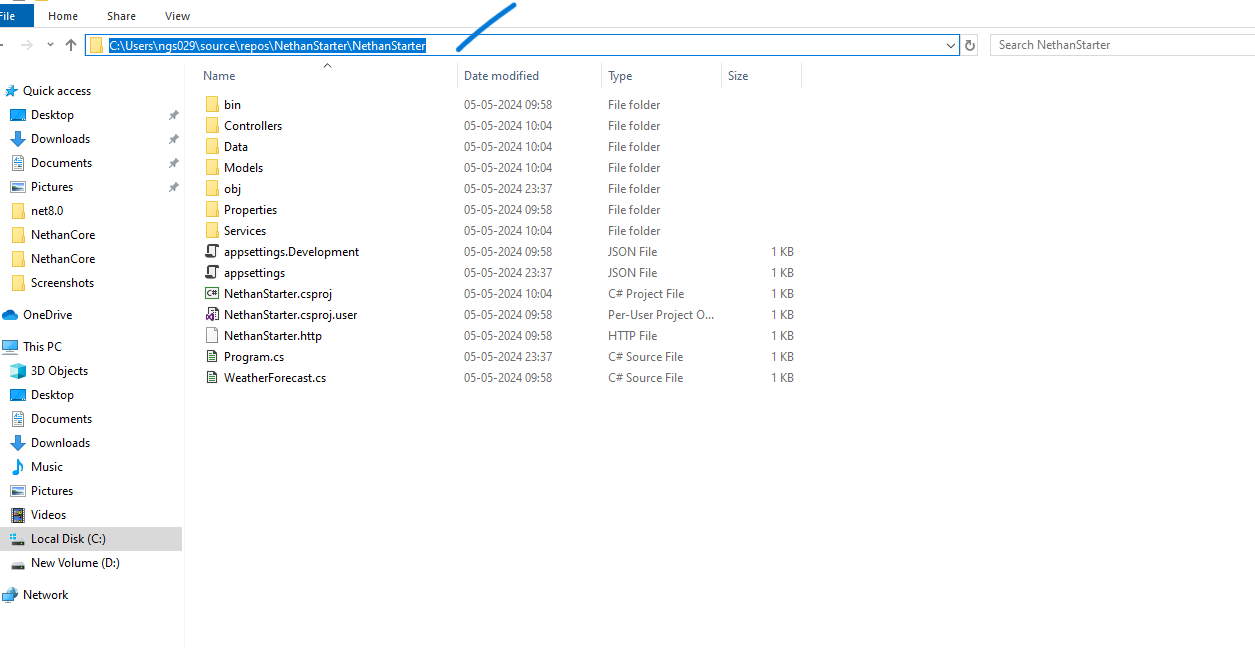
Create the Empty Asp.netcorewebapi application (NethanStarter).

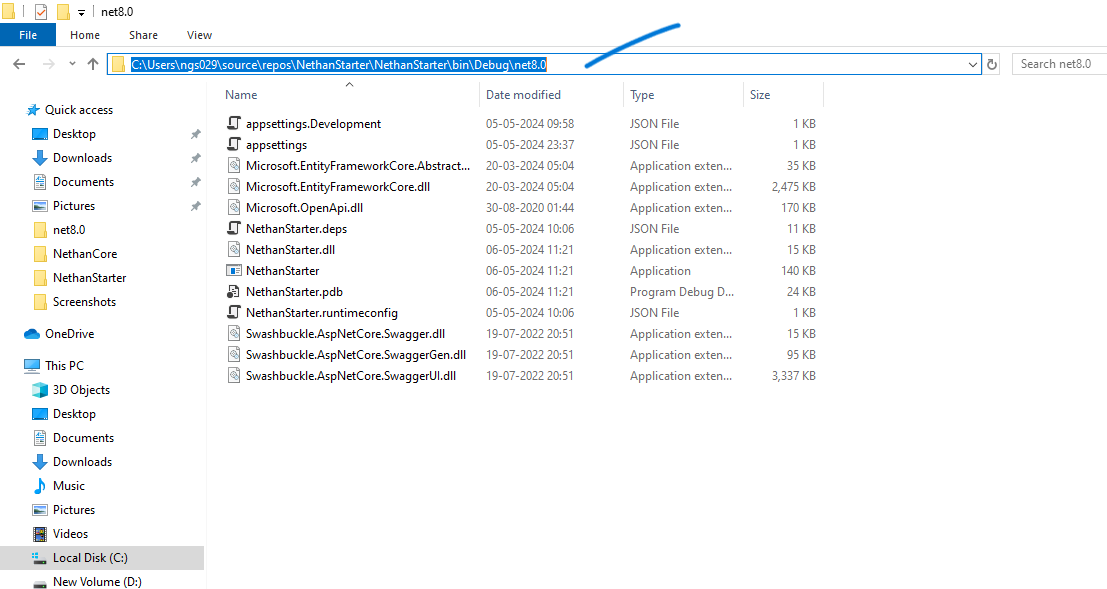
**Creation of Asp.netCoreWebAPI:**



Created the Asp. Netcore web Api application.

Named this application as Nethan Starter.

Above fig is the path of NethanStarter application.

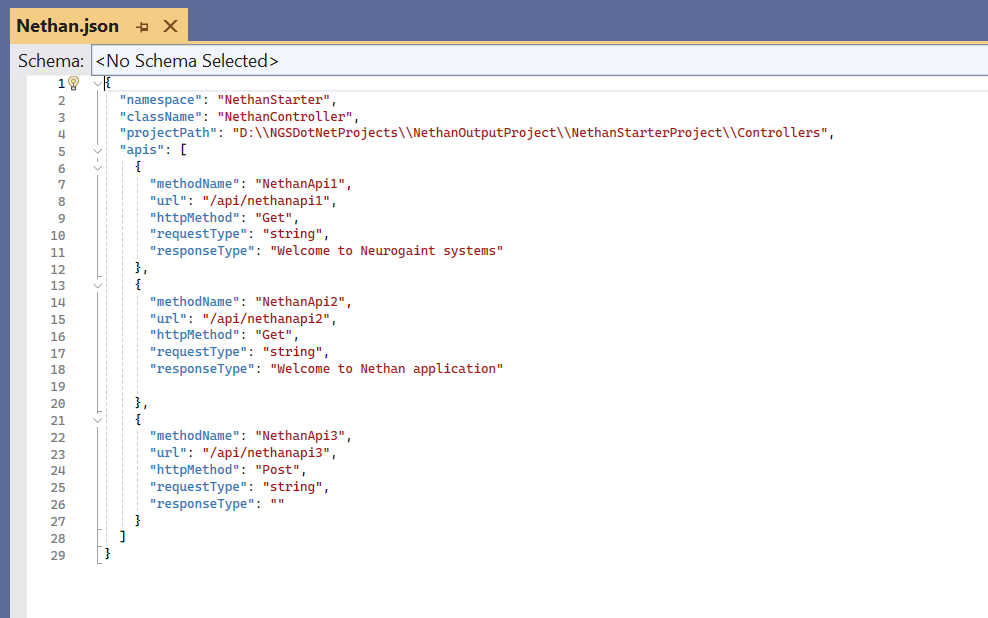


This above fig shows where the Json files are located.

In below fig the Nethanstarter application path has given in Nethancore app for all Json files.

Then We need to mention the path file in Nethancore Application for all Json files

And provide the properties values for the below as mentioned to generate CONTROLLER stubs



Namespace: Provide the namespace name

className: Name of the controller class name

ProjectPath: Destination path of the target application

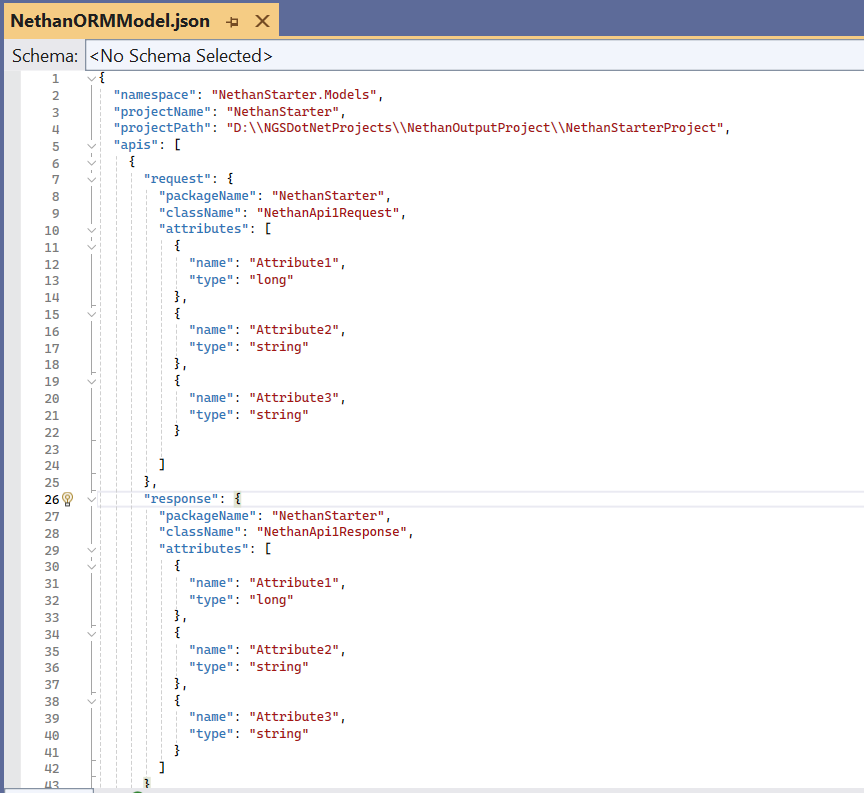
Apis: It consists of List of API properties.

1. methodName: Declare the name of the method.
2. URL: Specify the Context path URL.
3. method:  here we are using Get method.
4. requestClass: Provide the RequestClass(Data type) based on your requirement

responseClass: provide the ResponseClass (Data type), based on your requirement.

By default, you can give it as null.

2.To Generate “NethanORMModelGenerator” stubs please provide the properties values for the below as mentioned.



namespace: Provide the name space of the target project.

projectPath: Mention the Location and name of the Target Project.

apis: It consists of List of API properties.

packageName: Declare the Package Name.

className: Declare the Class Name.

c. attributes: Take the list of properties

d. name: provide the name of attribute.

e. type: provide the Data Type.

Repeat the same if you have multiple request/responses.

To Generate “NethanServiceGenerator” stubs please provide the properties values for the below as mentioned



namespace: Mention the namespace of the Target Project.

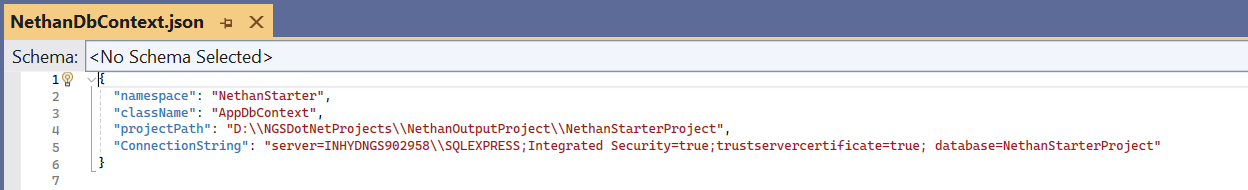
projectPath: Mention the Location of the Target Project.

Apis: It consists of List of BusinessLogicAPI properties.

b.className: Declare the Class Name.

c.methodName: declare the Method Name.

4. To Generate “NethanDbContext” stubs please provide the database details as mentioned below



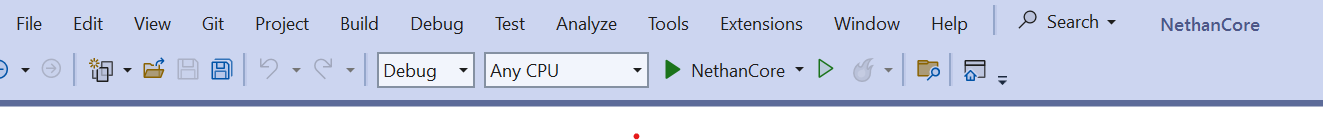
Namespace: provide target project name

Class name: provide the dB context class name

Project Path: The location of the target project

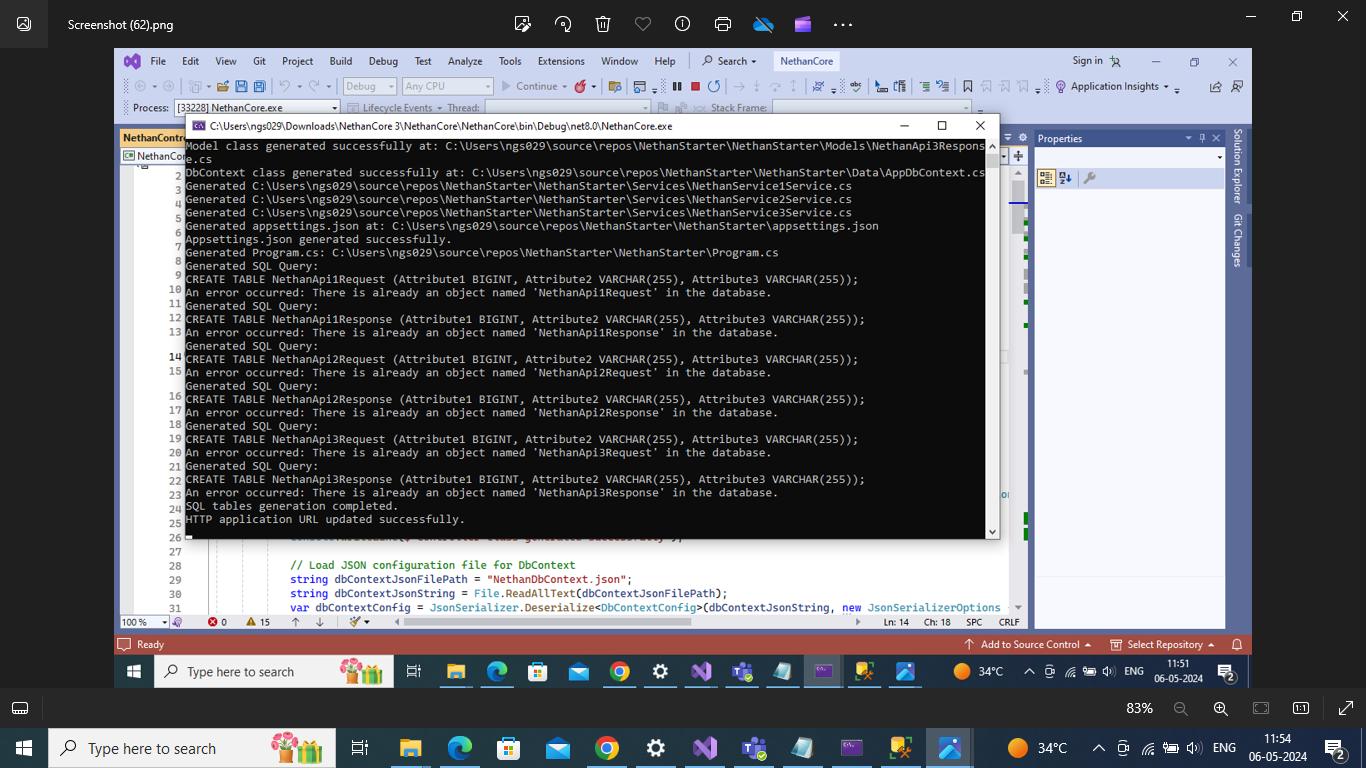
Connection String: Provide the db server name and respective database name

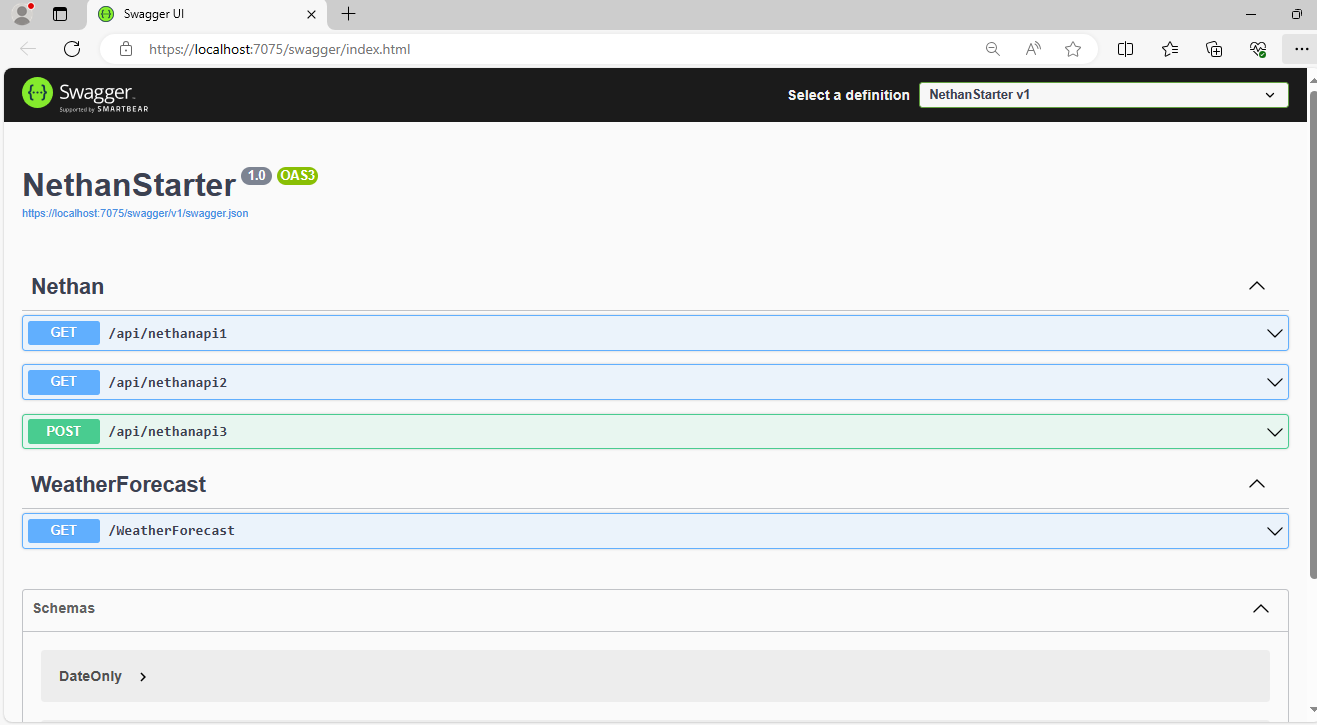
Run the NethanCore application once all properties are configured.



Click NethonCore button or F5 to run the application.

After a successful process, the stubs will be generated in the target application (NethanStarter). In Below fig we have the output of stubs creation.

Then Run the NethanStarterApplication.

Above fig is the output of target application.