KYC360 Assignment

1. Environment SetUp

2. Coding

   Technology: ASP.NET

3. Database

   SQL Server

4. Testing

   Swagger UI

5. Future Scope

1.Environment Set-Up:

Download and install Visual Studio 2022 and SQL Server

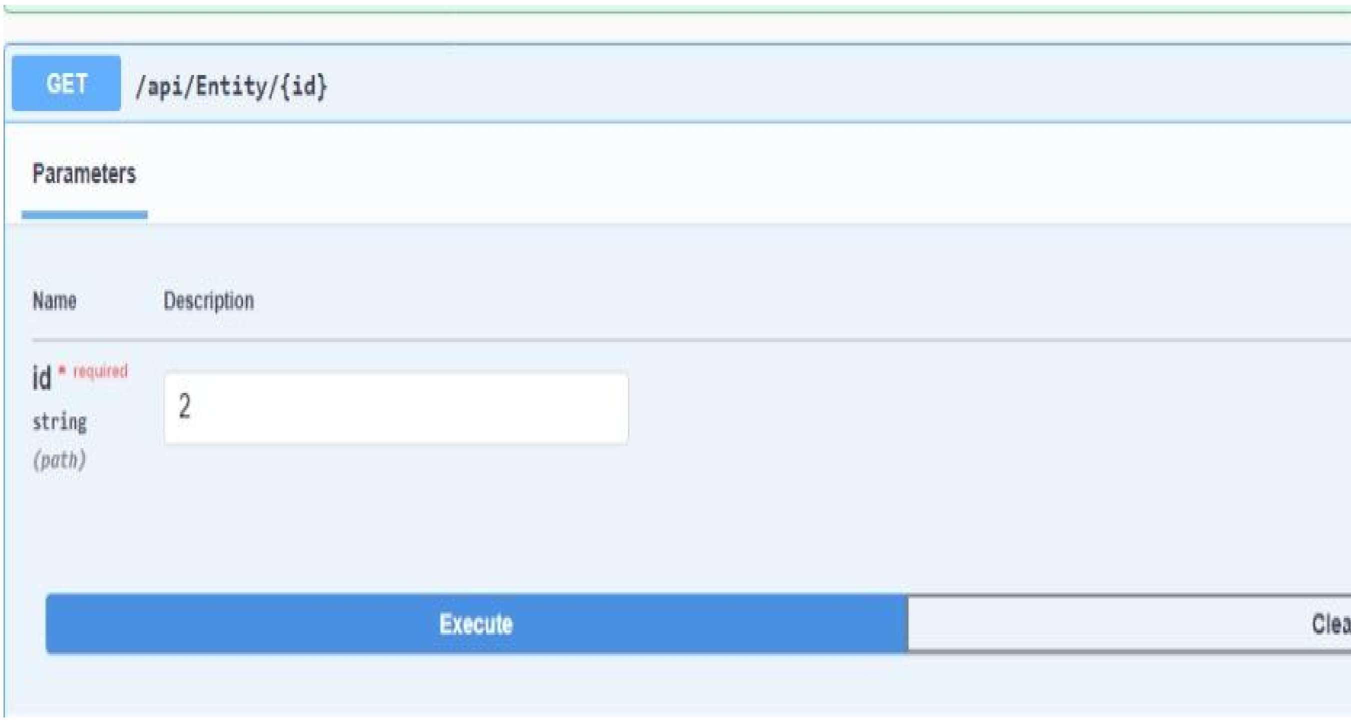
Install Microsoft Entity Framework

Install Polly Library for Retry and Back-off Mechanism

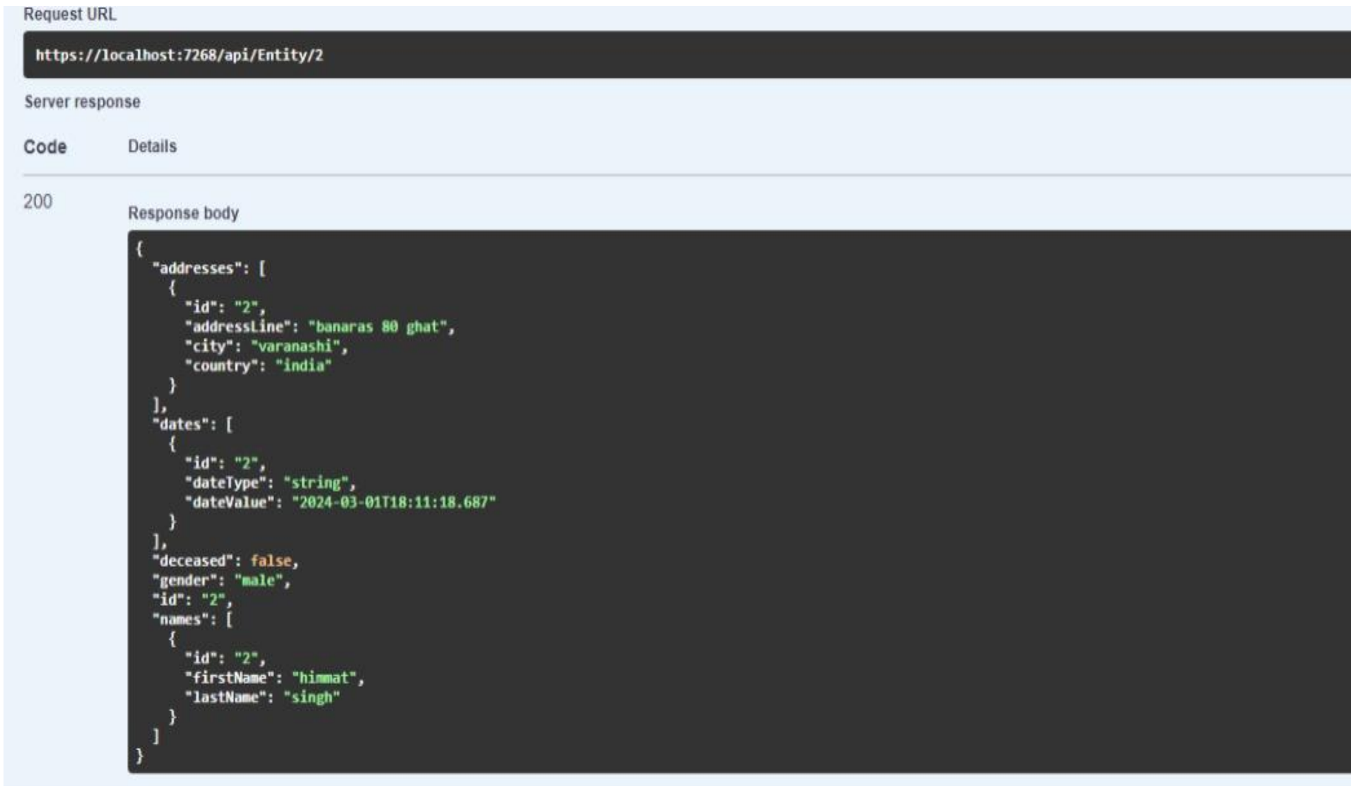
2.Coding:

Built an API using ASP.NET as Back-end technology and implemented CRUD endpoints.

Single entity: Users would be able to retrieve a single entity by id.



In above query, make a request with id equal to 2 and response will be -


Request URL

```
https://localhost:7268/api/Entity/2
```

Server response

Code Details

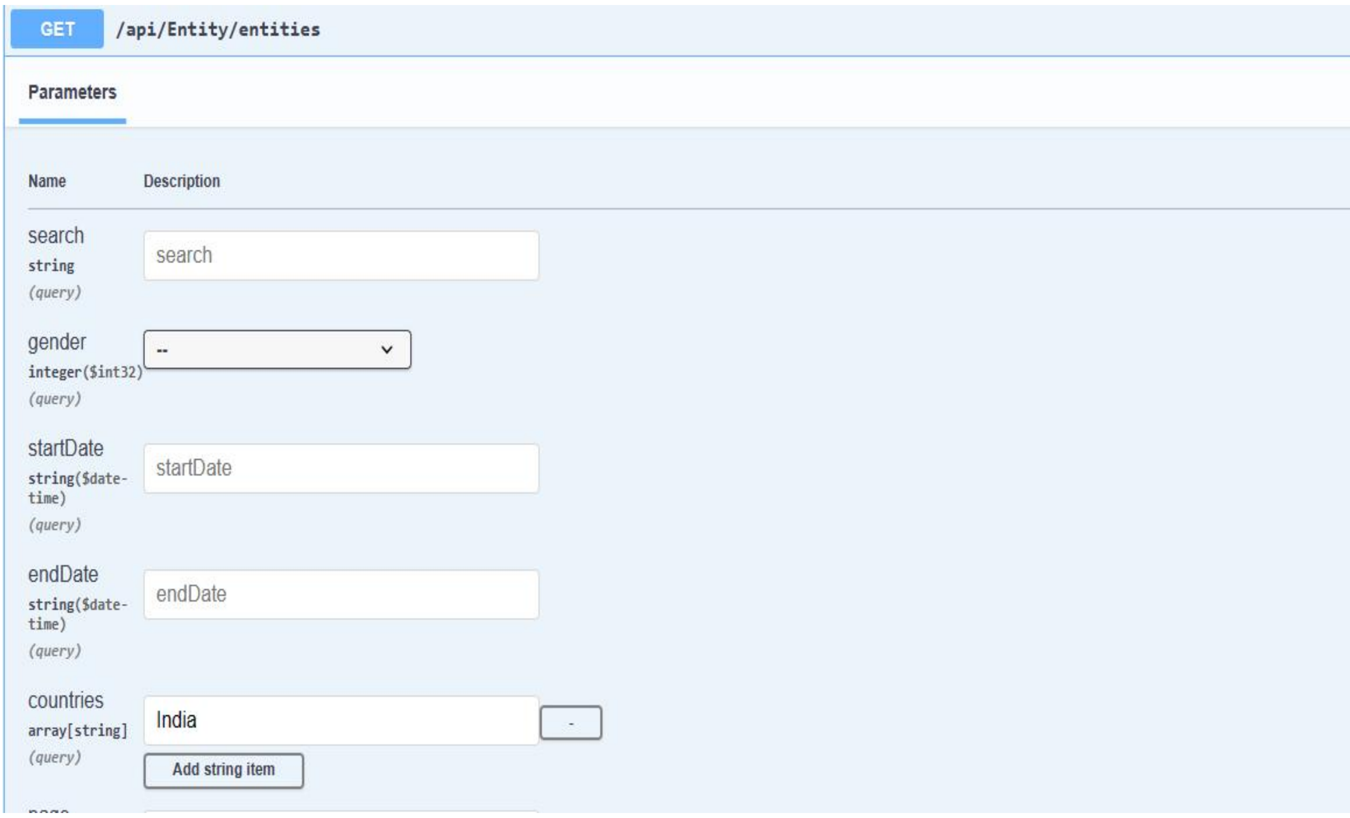
200 Response body

```
{
  "addresses": [
    {
      "id": "2",
      "addressLine": "banaras 80 ghat",
      "city": "varanashi",
      "country": "india"
    }
  ],
  "dates": [
    {
      "id": "2",
      "dateType": "string",
      "dateValue": "2024-03-01T18:11:18.687"
    }
  ],
  "deceased": false,
  "gender": "male",
  "id": "2",
  "names": [
    {
      "id": "2",
      "firstName": "himmat",
      "lastName": "singh"
    }
  ]
}
```

Listing Entities: An endpoint, user can fetch a list of entities.

Searching and Filtering: User can search the data based on country name, first name or last name. And user can filter the data based on gender, start date and/or end date.

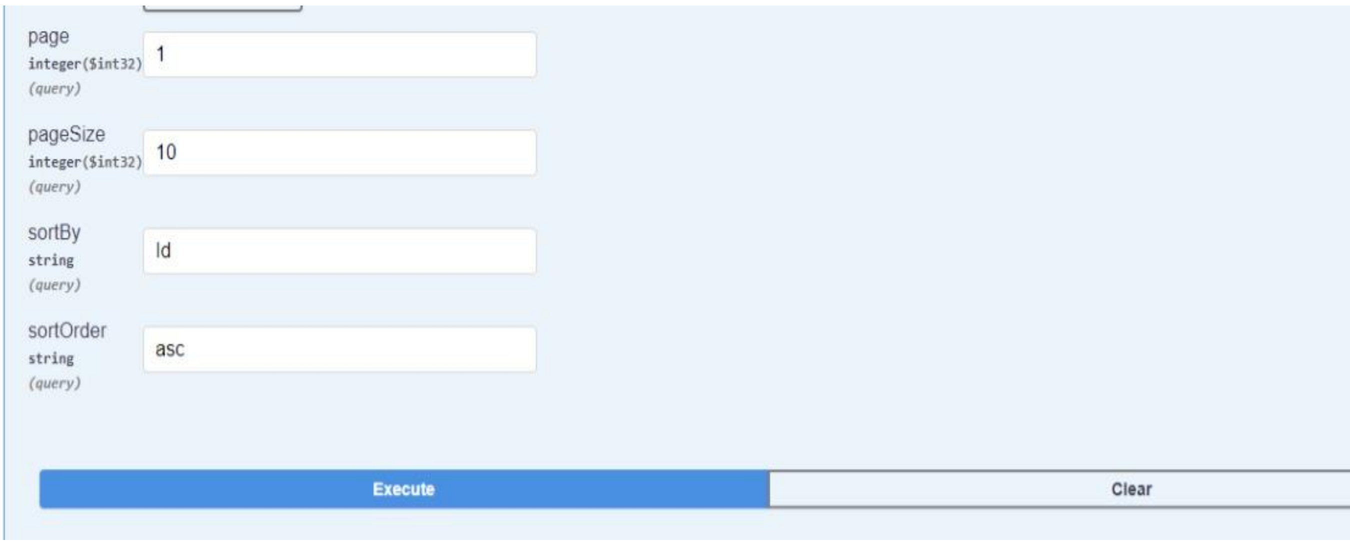
Support for pagination and sorting: User can send number of entries each page to show and page number to display, and will get the required output.



GET /api/Entity/entities

Parameters

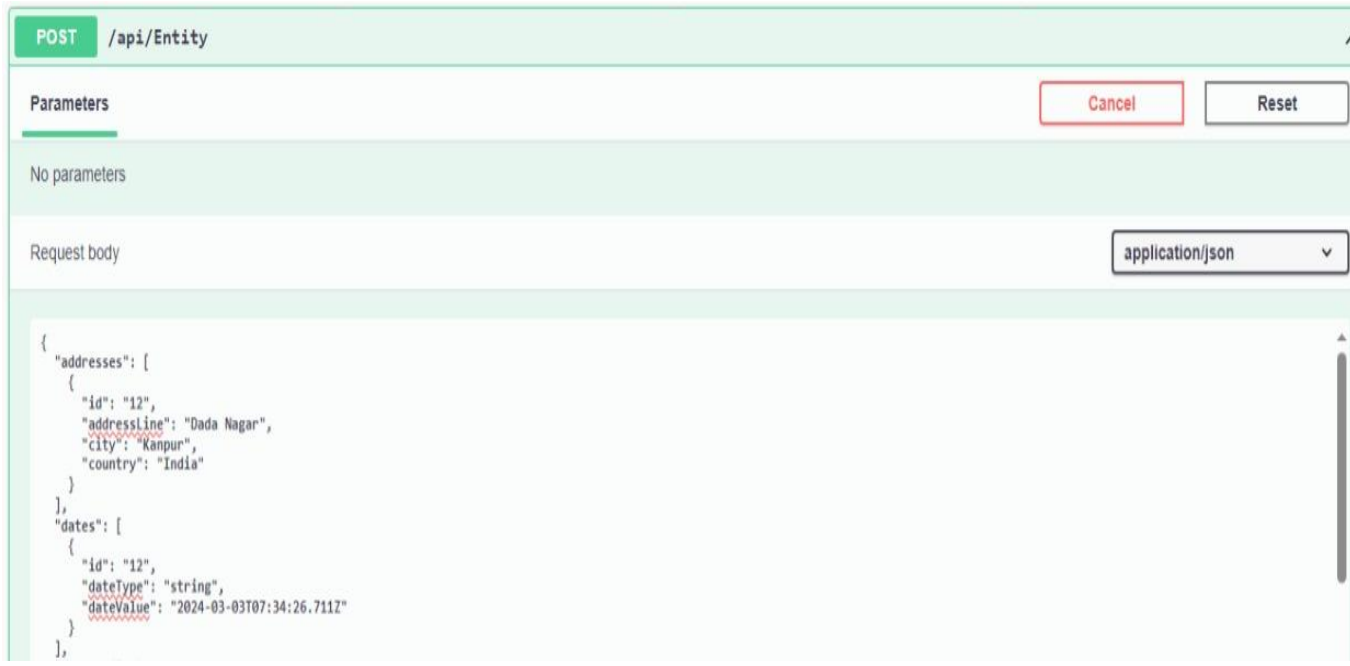
Name	Description
search string *(query)*	search
gender integer($int32) *(query)*	--
startDate string($date-time) *(query)*	startDate
endDate string($date-time) *(query)*	endDate
countries array[string] *(query)*	India



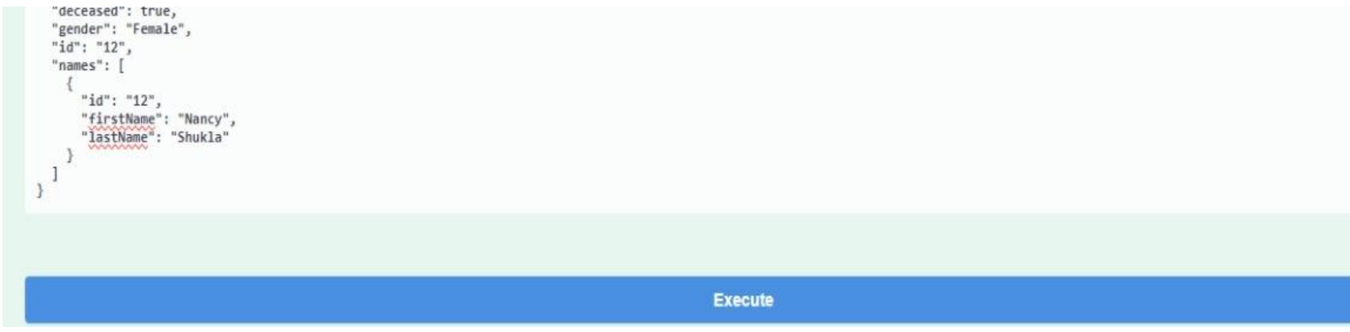
Name	Description
page integer($int32) *(query)*	1
pageSize integer($int32) *(query)*	10
sortBy string *(query)*	Id
sortOrder string *(query)*	asc

By applying searching, filtering, pagination and sorting, data can be fetched accordingly. Page denotes page number that user wants to display and pageSize denotes the number of entries in each page.

Writing Data(Create):



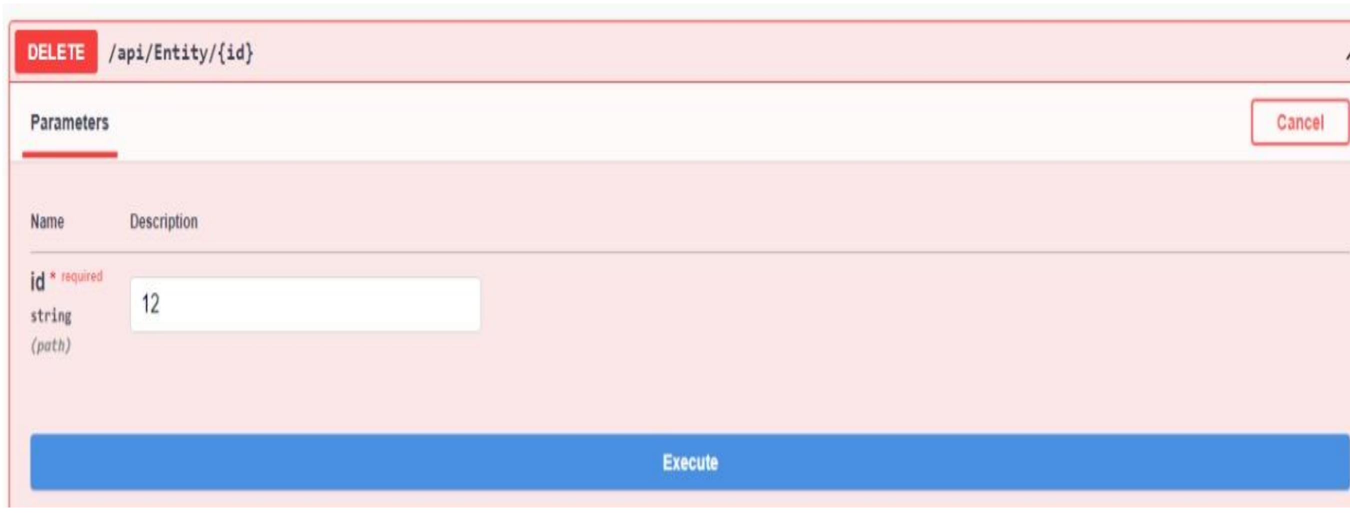
```
{
  "addresses": [
    {
      "id": "12",
      "addressLine": "Dada Nagar",
      "city": "Kanpur",
      "country": "India"
    }
  ],
  "dates": [
    {
      "id": "12",
      "dateType": "string",
      "dateValue": "2024-03-03T07:34:26.711Z"
    }
  ],
```



```
  "deceased": true,
  "gender": "Female",
  "id": "12",
  "names": [
    {
      "id": "12",
      "firstName": "Nancy",
      "lastName": "Shukla"
    }
  ]
}
```

Implemented Retry and Back-off Mechanism:
For writing operation, implemented a retry mechanism with exponential back-off.
It utilizes the **AsyncRetryPolicy** from the Polly library.
The **WaitAndRetryAsync** method specifies to handle exceptions of type **Exception** and retries the operation up to 3 times with an increasing delay defined by exponential back-off.
A randomized simulation of failure to trigger the retry mechanism through Swagger API.

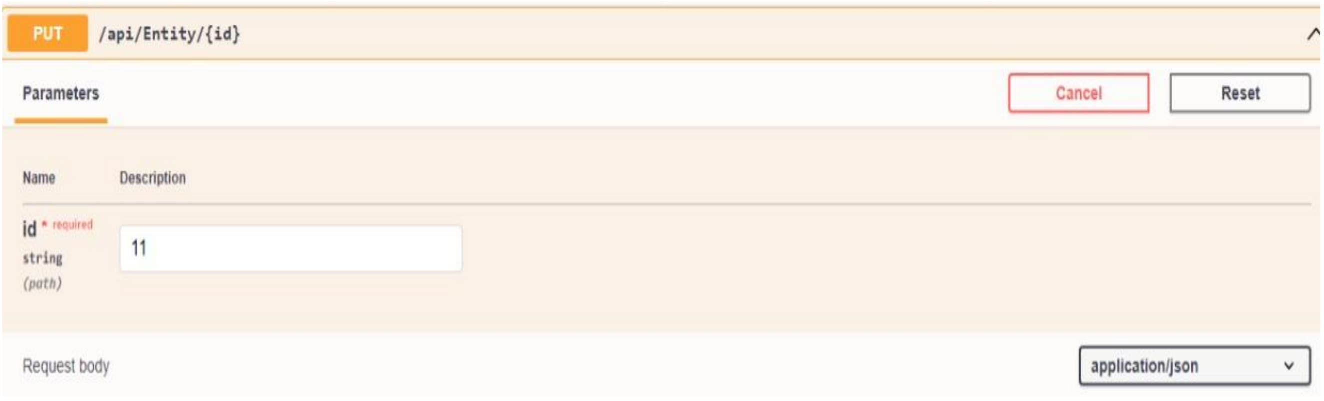
Delete:



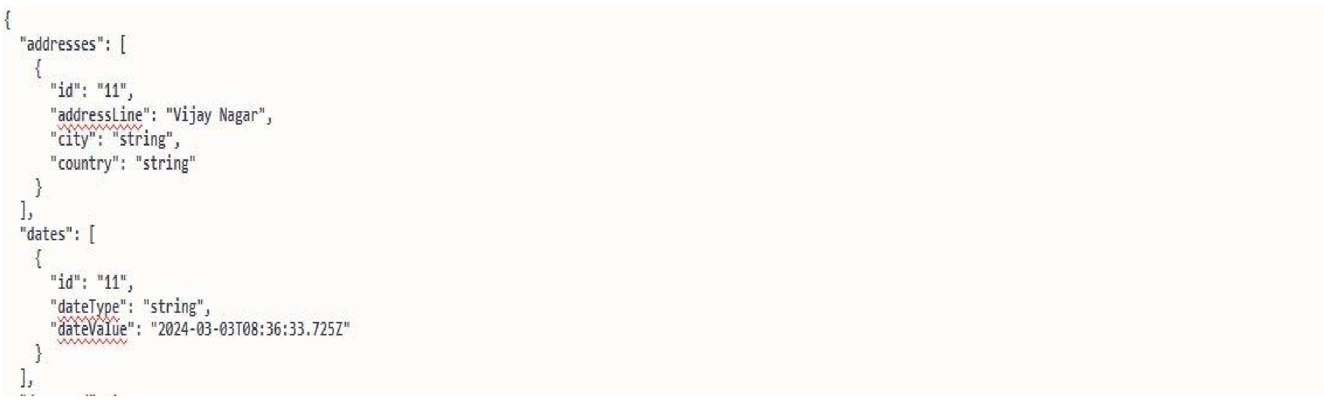
id: 12

Data can be deleted with the help of id.

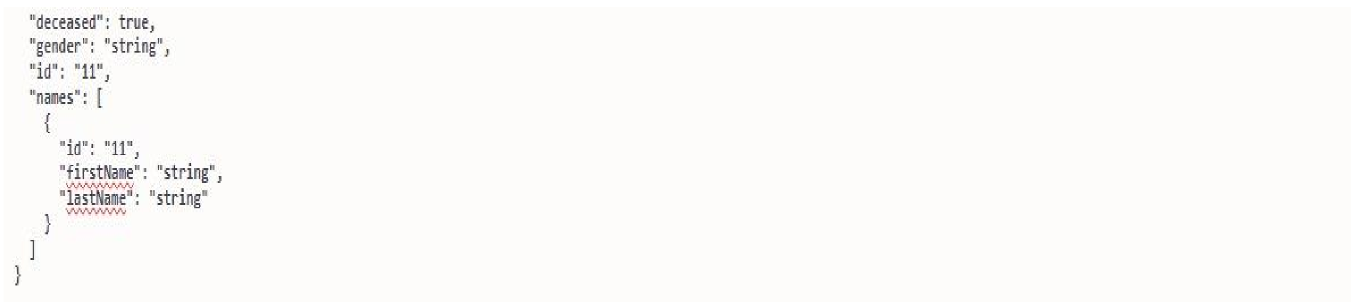
Update:



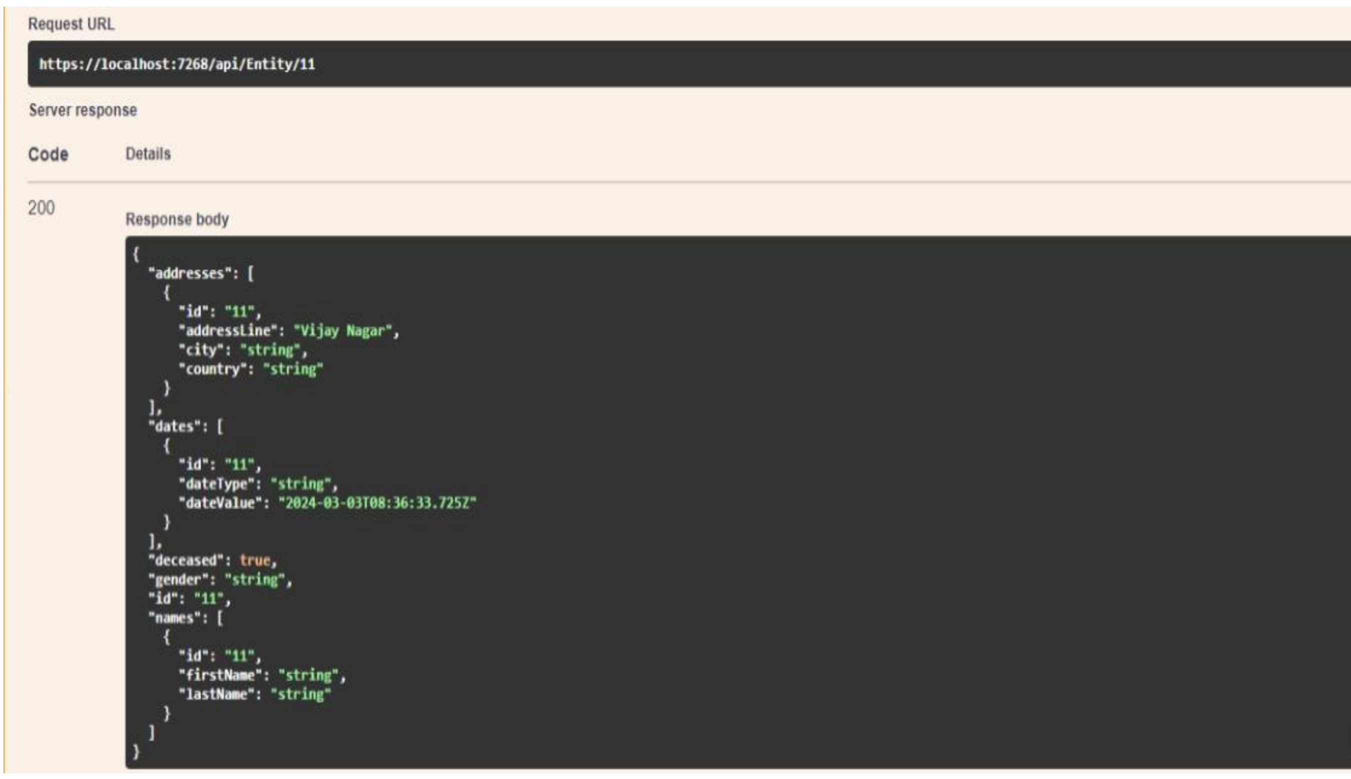
id: 11



```
{
  "addresses": [
    {
      "id": "11",
      "addressLine": "Vijay Nagar",
      "city": "string",
      "country": "string"
    }
  ],
  "dates": [
    {
      "id": "11",
      "dateType": "string",
      "dateValue": "2024-03-03T08:36:33.725Z"
    }
  ],
```



```
  "deceased": true,
  "gender": "string",
  "id": "11",
  "names": [
    {
      "id": "11",
      "firstName": "string",
      "lastName": "string"
    }
  ]
}
```



Request URL

```
https://localhost:7268/api/Entity/11
```

Server response

Code	Details
200	Response body

```
{
  "addresses": [
    {
      "id": "11",
      "addressLine": "Vijay Nagar",
      "city": "string",
      "country": "string"
    }
  ],
  "dates": [
    {
      "id": "11",
      "dateType": "string",
      "dateValue": "2024-03-03T08:36:33.725Z"
    }
  ],
  "deceased": true,
  "gender": "string",
  "id": "11",
  "names": [
    {
      "id": "11",
      "firstName": "string",
      "lastName": "string"
    }
  ]
}
```

3.Database:

Used SQL server for database management. There are 4 tables in the database.

Address(ID Primary key, AddressLine, City, Country, EntityId),

Date(Id, DateType, DateValue, EntityId),

Name(Id, FirstName, LastName, EntityId),

Entities(Id, Deceased, gender)

For now, ID column in address, ID column in Date, ID column in Name and ID column in Entities have same values and ID is given along with other details when data is inserted.

Note: So, this is simple implementation to check working of all the CRUD endpoints. It can be done in better way by reducing Redundancy.

4.Testing:
A randomized simulation of failure to trigger the retry mechanism through Swagger API.

5.Future Scope:
In future scope of this assignment, we'll implement database in more effective way.(Not providing ID manually, data redundancy, data privacy etc.)