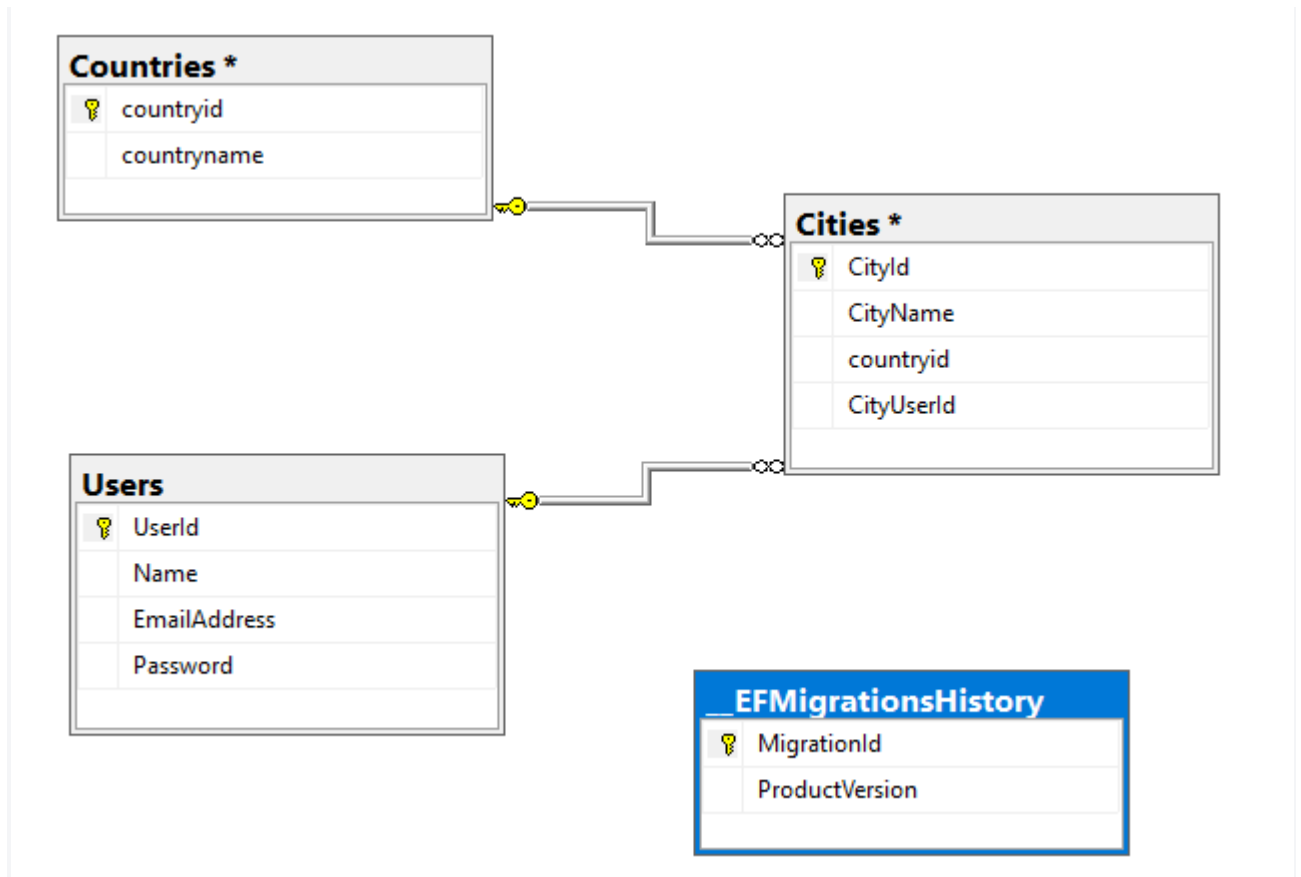


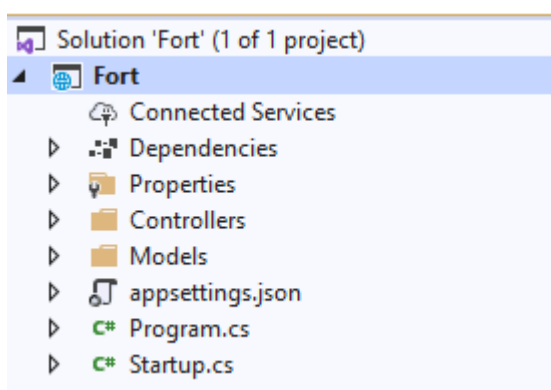
Exercise-1 EXPLANATIONS

Data Base: FortCode

Created 3 tables – Users, Countries, Cities



Solution: Created new empty ap.net core API Project – **Fort**



1. Added below **nuGet** packages.

- ✓ Microsoft.EntityFrameworkCore.SqlServer
- ✓ Microsoft.EntityFrameworkCore.Tools
- ✓ Microsoft.EntityFrameworkCore.Design

- ✓ Microsoft.AspNetCore.Identity
- ✓ Microsoft.AspNetCore.Authentication.JwtBearer
- ✓ Microsoft.AspNetCore.Mvc.NewtonsoftJson

2. Using Scaffold-DbContext command(E.F) Added DbContext class and models.
3. Added DB connection string in **appsettings.json** file

```
"AllowedHosts": "*",
"ConnectionStrings": {
  "FortDB": "Server=VALI-PC\\SQLEXPRESS;Database=FortCode;UID=sa;PWD=123;"
},
```

4. "Controllers" folder created 2 new controllers and "Model " folder 2 interface classes and 2 classes.

1. UserController
2. CityController
1. IAuthRepository
2. ICityRepository
3. AuthRepository
4. CityRepository

=====*****=====

UserController:

```
[Route("api/[controller]")]
1. [HttpPost("Register")]
2. [HttpPost("Login")]
```

Register:

- 1) Required fields and format validation
- 2) Checking User Email address exists or not
- 2) Security purpose password will be encrypting.
Ex: Test@12345 ---> VGVzdEAxMjM0

Login:

- 1) Required fields and format validation
- 2) Checking User Id(Email address) exists or not
- 3) Password validation (Encrypt)
Ex: Test@12345 ---> VGVzdEAxMjM0==DB Password
- 4) Create **JwtSecurityToken** for **Authorize** user. That will be expire one day.

Note: For API testing we used "[POSTMAN](#)" App.

Testing: [POSTMAN](#)

1. Register

Post: <https://localhost:44362/api/User/Register>

Body: { "Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"" }

- Password=null

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for '+ New', 'Import', 'Runner', and 'My Workspace'. The main area is titled 'Untitled Request' and shows a POST request to 'https://localhost:44362/api/User/Register'. The request body is set to JSON and contains the following data: { "Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"" }. The status bar at the bottom indicates a '400 Bad Request' error with a response time of 23 ms and a size of 402 B. The response body is displayed in the 'Body' tab, showing an error message: { "errors": { "Password": ["Password is required"] }, "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1" }. The 'Follow link' button is visible next to the type field.

Postman

File Edit View Help

+ New Import Runner My Workspace Invite

No Environment

POST https://localhost:44362/api/User/Register

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "Name": "Sheksha",
3   "EmailAddress": "sheksha@gmail.com",
4   "Password": ""
5 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": {
3     "Password": [
4       "Password is required"
5     ]
6   },
7   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
8 }
```

Find and Replace Console Bootcamp

Body: {"Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"Test@1122"}

POST ▼ https://localhost:44362/api/User/Register Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings ●

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1
2 { "Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"Test@1122" }
```

Body Cookies Headers (5) Test Results 🌐 Status: 200 OK Time: 2.92 s Size: 215 B

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1 {
2   "data": 10,
3   "success": true,
4   "message": null
5 }
```

DB:

select * from users

90 % ▼

Results Messages

	UserId	Name	EmailAddress	Password
1	9	Vali	vali@gmail.com	VGvzdEAxMjM0
2	10	Sheksha	sheksha@gmail.com	VGvzdEAxMTly

Body: { "Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"Test@1122"}

- EmailAddress =Already register email address

POST ▼ https://localhost:44362/api/User/Register Send ▼

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings ●

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1
2 { "Name":"Sheksha", "EmailAddress":"sheksha@gmail.com", "Password":"Test@1122" }
```

Body Cookies Headers (5) Test Results 🌐 Status: 400 Bad Request Time: 128 ms Size: 242 B Save

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1 {
2   "data": 0,
3   "success": false,
4   "message": "User already exists."
5 }
```

2. Login

Post: <https://localhost:44362/api/User/Login>

Body: { "EmailAddress":"sheksha","Password":"" }

- EmailAddress =Invalid email format
- Passwords=null

POST <https://localhost:44362/api/User/Login> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings ●

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1 {
2   "EmailAddress": "sheksha",
3   "Password": ""
4 }
```

Body Cookies Headers (5) Test Results Status: 400 Bad Request Time: 24 ms Size: 475 B Sav

Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "errors": {
3     "Password": [
4       "Password is required"
5     ],
6     "EmailAddress": [
7       "The EmailAddress field is not a valid e-mail address."
8     ]
9   },
10 }
```

Body: { "EmailAddress":"sheksha@gmail.com", "Password":"aaaTest@1122" }

- Password is wrong

POST <https://localhost:44362/api/User/Login> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings ●

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1 { "EmailAddress": "sheksha@gmail.com", "Password": "aaaTest@1122" }
```

Body Cookies Headers (5) Test Results Status: 400 Bad Request Time: 51 ms Size: 239 B Sav

Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "data": null,
3   "success": false,
4   "message": "Wrong password"
5 }
```

- User Id not there

Body: { "EmailAddress": "sheksha@gmail.com", "Password": "Test@1122" }

- Valid Credentials then generate Authorize key (Using JWT).

===== End - UserController =====

CityController:

```
[Route("api/[controller]")]
```

```
[Authorize]
```

1. [HttpPost("AddCountry")]
2. [HttpGet("GetCityList")]
3. [HttpGet("GetCity")]
4. [HttpPost("AddCity")]
5. [HttpPost("UpdateCity")]
6. [HttpPost("DeleteCity")]

Note: In this controller we added `[Authorize]`. That's why we have to pass Key and Value in header.
Without add key value json request will show **401-Unauthorized** error.

Header:

Key: Authorization

Value: Bearer + JWT Key

ex: Bearer

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJGbzJ0U2VydmljZUFjY2Vzc1Rva2VuliwianRpljoiZDYyYTJmNmUtOTZkZS00MGMxLWFhNmQtNGUwZTAzOTBkMjgwiwiWFOljoimS8xOS8yMDIxlDEwOjI4OjA2IEFNliwiSWQiOiIxMCIslk5hbWUiOiJTaGVrc2hhliwiRW1haWwiOiJzaGVrc2hhQGdtYWlsLmNvbSIsImV4cCI6MTYxMTEzODQ4NiwiXNzljoiRm9ydFNlcilzImF1ZCI6IklZvcnRTZXJ2aWNIUG9zdG1hbktNsaWVudCJ9.RtyZGHCMjFztBmXJeF5Q73g7AVDLn6XHA6nJdmxcWis

AddCountry:

- 1) **Verify** JSON Web Token ID
- 2) Required fields validation
- 3) Checking Country exists or not

AddCity:

- 1) **Verify** JSON Web Token ID
- 2) Required fields validation
- 3) Checking User Favorite City exists or not

GetCity:

- 1) **Verify** JSON Web Token ID
- 2) City Id Validation
- 3). Check CityUserId == LoginUseelD

GetCityList:

- 1) **Verify** JSON Web Token ID
- 2) Check CityUserId == LoginUseelD

UpdateCity:

- 1) **Verify** JSON Web Token ID
- 2) Required fields validation
- 3) Check City exist or not and CityUserId == LoginUseelD
- 4) Show response City Updated or not

DeleteCity:

- 1) **Verify** JSON Web Token ID
- 2) City Id Validation
- 3) Check CityUserId == LoginUseelD
- 4) Show response City Deleted or not

1. Add Country

Post: <https://localhost:44362/api/City/addCountry>

JSON Header: Adding JWT key, value (In this controller we have to add JWT id to all API requests)

POST

https://localhost:44362/api/City/addCountry

Send

ParamsAuthorizationHeaders (9)Body ●Pre-request ScriptTestsSettings ●

Headers8 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiJ...	JWT ID		

Body: { "CountryName": "Srilanka" }

POST

https://localhost:44362/api/City/addCountry

Send

ParamsAuthorizationHeaders (9)Body ●Pre-request ScriptTestsSettings ●

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQLJSON ▼

1 { "CountryName": "Srilanka" }

BodyCookiesHeaders (5)Test Results

● Status: 200 OKTime: 1863 msSize: 238 B

PrettyRawPreviewVisualizeJSON ▼

1 {
2 "data": 0,
3 "success": true,
4 "message": "Country Sucessfully Added."
5 }

Body: { "CountryName":""}

- CountryName =null

POST

https://localhost:44362/api/City/addCountry

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

Set as variable

...

www-form-urlencoded

raw

binary

GraphQL

JSON

1

{ "CountryName":""}

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"errors": {

3

"Countryname": [

4

"Country Name is required"

5

]

}

Body: { "CountryName":" Srilanka "}

- CountryName =Already exists

POST

https://localhost:44362/api/City/addCountry

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{ "CountryName":"Srilanka"}

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"data": 0,

3

"success": false,

4

"message": "Country already exist."

5

}

2. AddCity

Post: <https://localhost:44362/api/City/addCity>

Header: // Same as above

Body: { "CityName": "Jaipur", "countryid": "1" }

The screenshot shows a REST client interface with a POST request to `https://localhost:44362/api/City/addCity`. The request body is a JSON object: `{ "CityName": "Jaipur", "countryid": "1" }`. The response status is 200 OK, with a time of 138 ms and a size of 250 B. The response body is displayed in a pretty-printed JSON format:

```
1 {
2   "data": 0,
3   "success": true,
4   "message": "Your favourite City Sucessfully Added."
5 }
```

Body: { "CityName": "Jaipur", "countryid": "1" }

The screenshot shows the same REST client interface, but the response status is 400 Bad Request, with a time of 79 ms and a size of 256 B. The response body is displayed in a pretty-printed JSON format:

```
1 {
2   "data": 0,
3   "success": false,
4   "message": "Your favourite City already exist."
5 }
```

Body: { "CityName": "", "countryid": "1" }

POST

https://localhost:44362/api/City/addCity

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{ "CityName": "", "countryid": "1" }

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"errors": {

3

"CityName": [

4

"City Name is required"

5

]

6

},

7

"type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",

8

"title": "One or more validation errors occurred.",

9

"status": 400,

3. GetCity

Get: https://localhost:44362/api/City/GetCity?cityId=6

Header: // Same as above

GET

https://localhost:44362/api/City/GetCity?cityId=6

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"cityId": 6,

3

"cityName": "JaipurNew",

4

"countryid": 1,

5

"countryName": "India",

6

"userId": 10,

7

"userName": "Sheksha"

8

}

DB:

```
select * from Cities where CityUserId=10
```

	CityId	CityName	countryid	CityUserId
1	6	Jaipur	1	10
2	7	Pune	1	10

Get: <https://localhost:44362/api/City/GetCity?cityId=455>

- User Favorite city not there

GET <https://localhost:44362/api/City/GetCity?cityId=455> Send

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

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

1

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 1646 ms Size: 323 B Save Resp

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",
3   "title": "Not Found",
4   "status": 404,
5   "traceId": "|2582e5ad-47d4a5ffe613fc3f."
6 }
```

4. GetCityList

Get: <https://localhost:44362/api/City/GetCityList>

GET <https://localhost:44362/api/City/GetCityList> Send

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

Body Cookies Headers (5) Test Results Status: 200 OK Time: 56 ms Size: 378 B Sa

Pretty Raw Preview Visualize JSON ▼

```
1 [
2   {
3     "cityId": 6,
4     "cityName": "Jaipur",
5     "countryid": 1,
6     "countryName": "India",
7     "userId": 10,
8     "userName": "Sheksha"
9   },
10  {
11    "cityId": 7,
12    "cityName": "Pune",
13    "countryid": 1,
14    "countryName": "India",
15    "userId": 10,
16    "userName": "Sheksha"
17  }
18 ]
```

5. UpdateCity

Post: <https://localhost:44362/api/city/UpdateCity>

Header: // Same as above

Body: { "CityUserId":"10","CityId":"8","CityName":"Jaipur","countryid":"1"}

The screenshot shows a REST client interface with a POST request to `https://localhost:44362/api/City/UpdateCity`. The request body is a JSON object: `{ "CityUserId": "10", "CityId": "8", "CityName": "Jaipur", "countryid": "1" }`. The response status is `400 Bad Request` with a time of `7.00 s` and size of `252 B`. The response body is a JSON object: `{ "data": 0, "success": false, "message": "City Not Updated or Not Found." }`.

```
POST https://localhost:44362/api/City/UpdateCity
```

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 { "CityUserId": "10", "CityId": "8", "CityName": "Jaipur", "countryid": "1" }
```

Body Cookies Headers (5) Test Results Status: 400 Bad Request Time: 7.00 s Size: 252 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": 0,
3   "success": false,
4   "message": "City Not Updated or Not Found."
5 }
```

Body { "CityUserId":"10","CityId":"6", "CityName":"JaipurNew","countryid":"1"}

The screenshot shows a REST client interface with a POST request to `https://localhost:44362/api/City/UpdateCity`. The request body is a JSON object: `{ "CityUserId": "10", "CityId": "6", "CityName": "JaipurNew", "countryid": "1" }`. The response status is `200 OK` with a time of `1813 ms` and size of `237 B`. The response body is a JSON object: `{ "data": 0, "success": true, "message": "City Updated Sucessfully." }`.

```
POST https://localhost:44362/api/City/UpdateCity
```

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON Beau

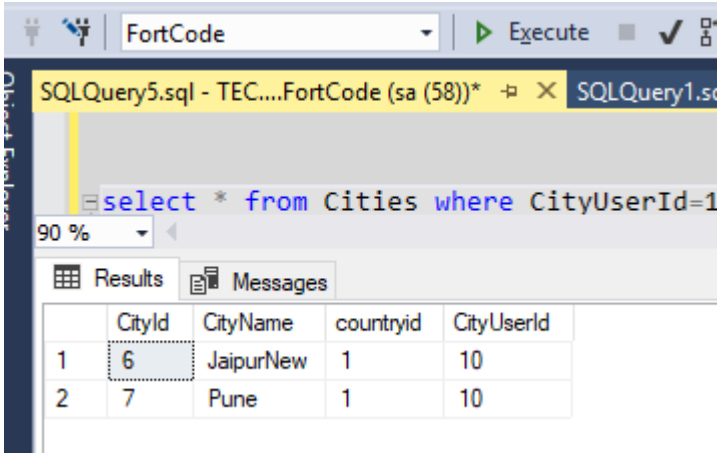
```
1 { "CityUserId": "10", "CityId": "6", "CityName": "JaipurNew", "countryid": "1" }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1813 ms Size: 237 B Save Respons

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": 0,
3   "success": true,
4   "message": "City Updated Sucessfully."
5 }
```

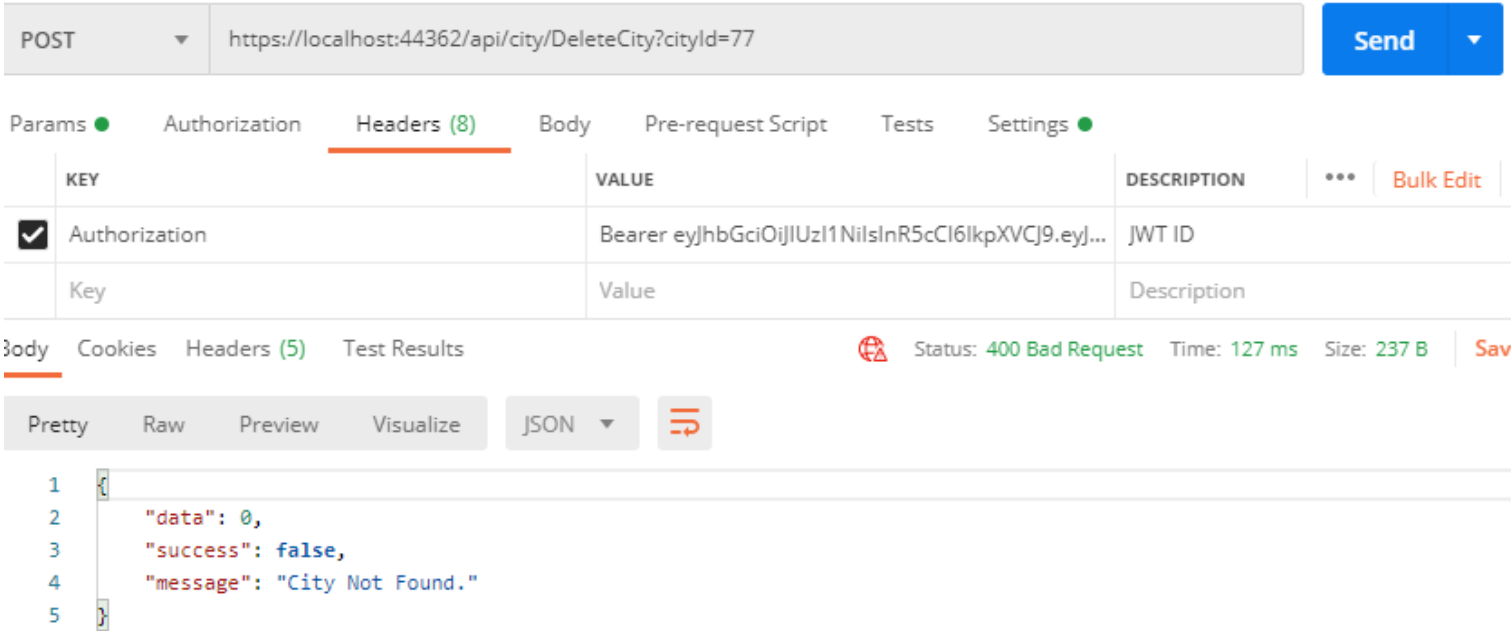
DB:



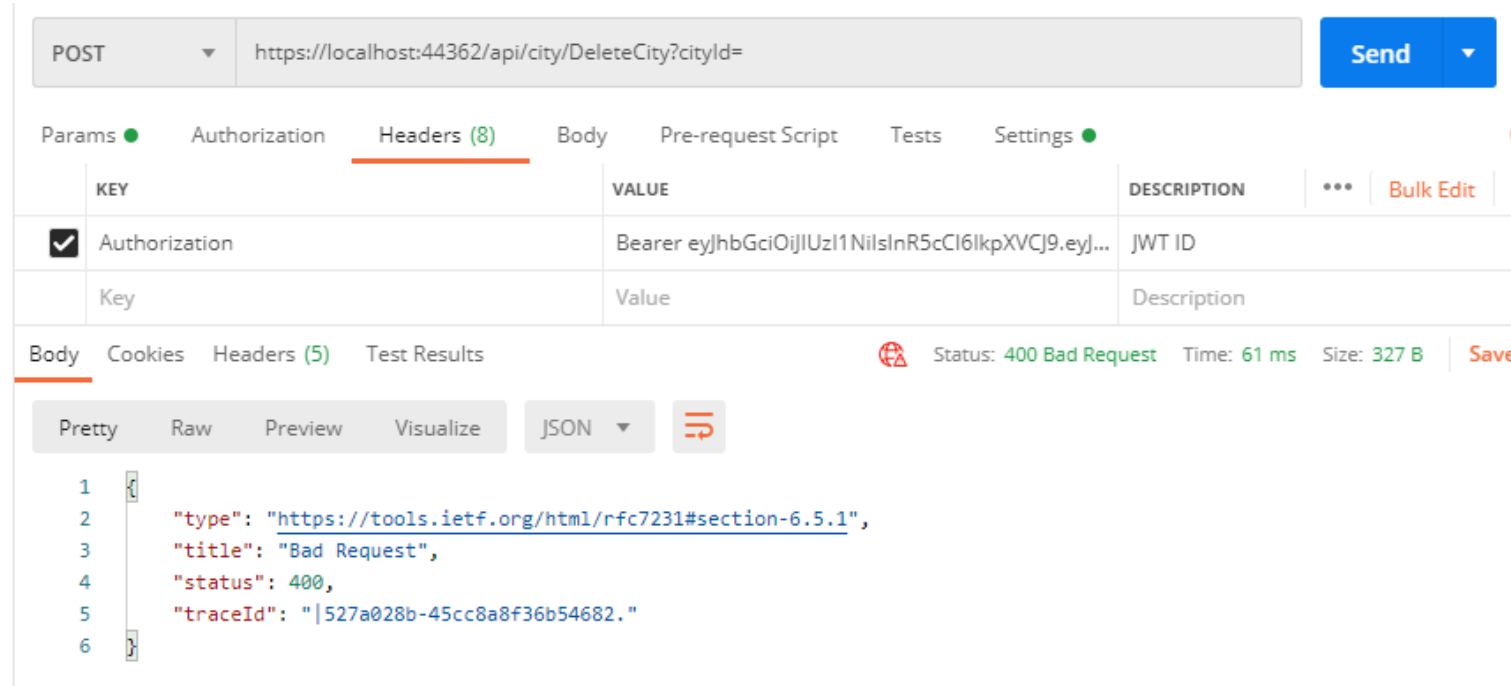
6. DeleteCity

Post: <https://localhost:44362/api/city/DeleteCity?cityId=77>

Header: // Same as above



Post: <https://localhost:44362/api/city/DeleteCity?cityId=>



DB:

SQL query: `select * from Cities where CityUserId=10`

90 %

Results Messages

	CityId	CityName	countryid	CityUserld
1	6	JaipurNew	1	10
2	7	Pune	1	10

Post: <https://localhost:44362/api/city/DeleteCity?cityId=7>

POST

https://localhost:44362/api/city/DeleteCity?cityId=7

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...	JWT ID		
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OK Time: 115 ms Size: 237 B Sav

Pretty

Raw

Preview

Visualize

JSON

```

1 {
2   "data": 0,
3   "success": true,
4   "message": "City Deleted Sucessfully."
5 }
```

DB:

SQL Query: `select * from Cities where CityUserId=10`

10 %

Results Messages

	CityId	CityName	countryid	CityUserId
1	6	JaipurNew	1	10

===== End - CityController =====