**27. Building Web API**

1. **Understanding HTTP Verbs**

* HTTP Verbs tell the server what to do with data identified by URL.
* The HTTP method is supplied in the request line and specifies the operation that the client has requested.
* The primary or most commonly used HTTP Verbs are POST, GET, PUT and DELETE, These correspond to Create, Read, Update and Delete (CRUD) operations.
* There are number of other verbs too, but they are utilized less-frequent methods.
* The HTTP Verbs comprise a major portion of our “uniform interface” constraint and provide us the action counterpart to the noun-based resources.

1. HTTP GET

* This verb should be used only to get information or data from database or other source.
* This method is used to read (Retrieve) a representation of a source.
* This method not to modify resource in any way, As GET requests do not change the state of the resource, these are said safe methods.
* GET APIs should be idempotent, which means that making multiple identical requests must produce the same result every time until another API has changed the state of the resource on the server.

1. HTTP POST

* This verb should be used only to create new entry with information or database or other source.
* The POST verb is most often utilized to create new resource.
* These methods are used to send the data to the API server to create a resources and the data sent to the server is stored in the request body of the HTTP request.
* This request is non-idempotent, It mutates data on backend server (By creating a resource) , as opposed to a GET request, Which does not change any data.

1. HTTP PUT

* This verb should be used only to update the existing entry with information or data to database or other source.
* Similar to POST, PUT requests are used to send data to the API to update or create resources.
* The difference is that PUT requests are idempotent that is calling the same PUT requests multiple times will always produce the same result, In contrast, calling a POST request repeatedly make have side effects of creating the same resource multiple times.

1. HTTP DELETE

* This verb should be used only to delete existing entry with database or other resources.
* This method is one of the most common in Web APIs.
* If a new user is created with a POST request, and it can be retrieved with a GET requests , then making a DELETE request will completely remove that user.

1. **Implement GET, POST, PUT, DELETE**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace mytrial.Controllers

{

public class ValuesController : ApiController

{

static List<string> languages = new List<string>()

{

"c#", "ASP.NET", "MVC"

};

// GET api/values

public IEnumerable<string> Get()

{

return languages;

}

// GET api/values/5

public string Get(int id)

{

return languages[id];

}

// POST api/values

public void Post([FromBody] string value)

{

languages.Add(value);

}

// PUT api/values/5

public void Put(int id, [FromBody] string value)

{

languages[id] = value;

}

// DELETE api/values/5

public void Delete(int id)

{

languages.RemoveAt(id);

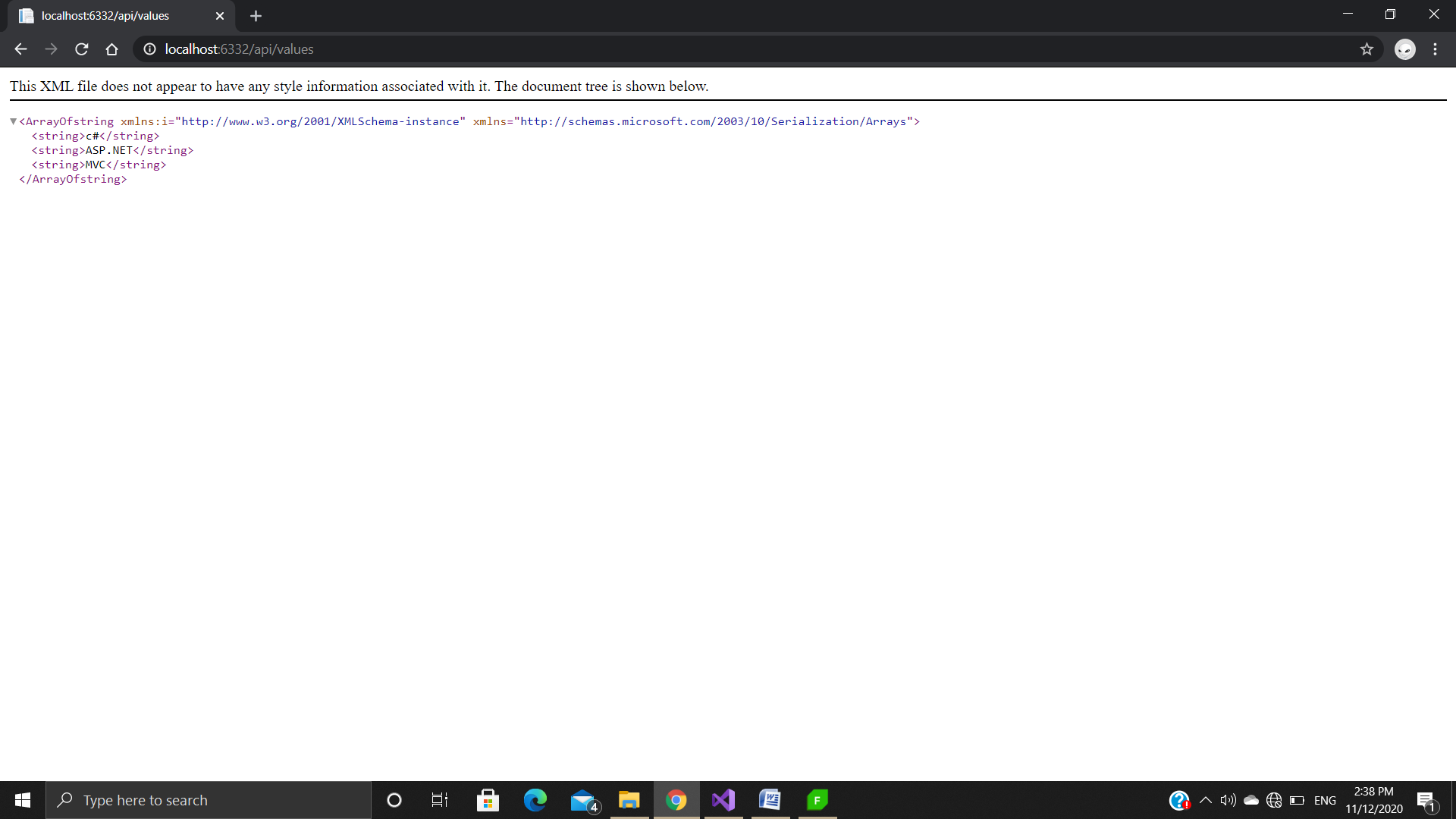
}

}

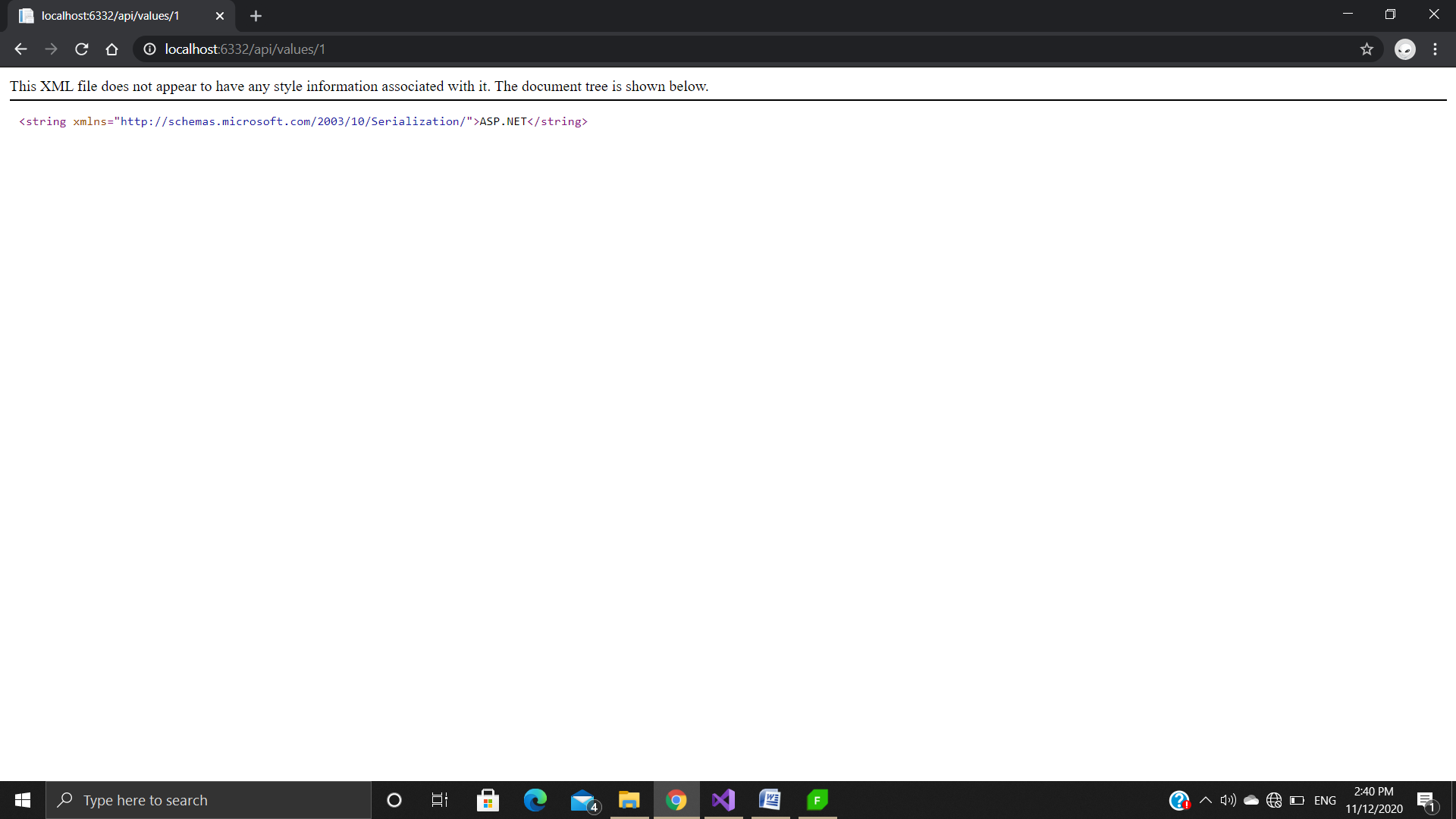
}

**OUTPUT**

* Build the project solution and run it on Google chrome
* Using URL <http://localhost:6332/api/values>

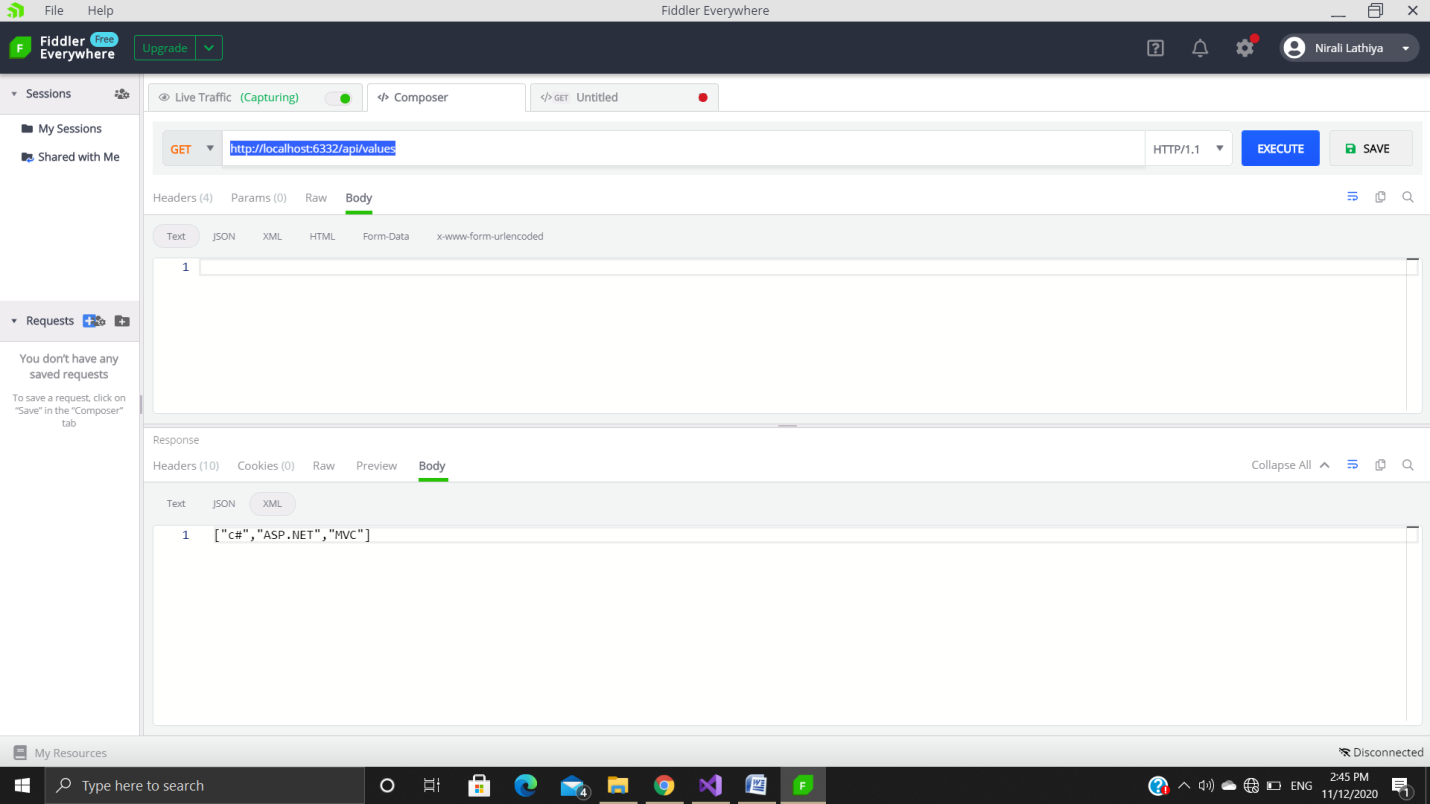


* Using URL <http://localhost:6332/api/values/1>

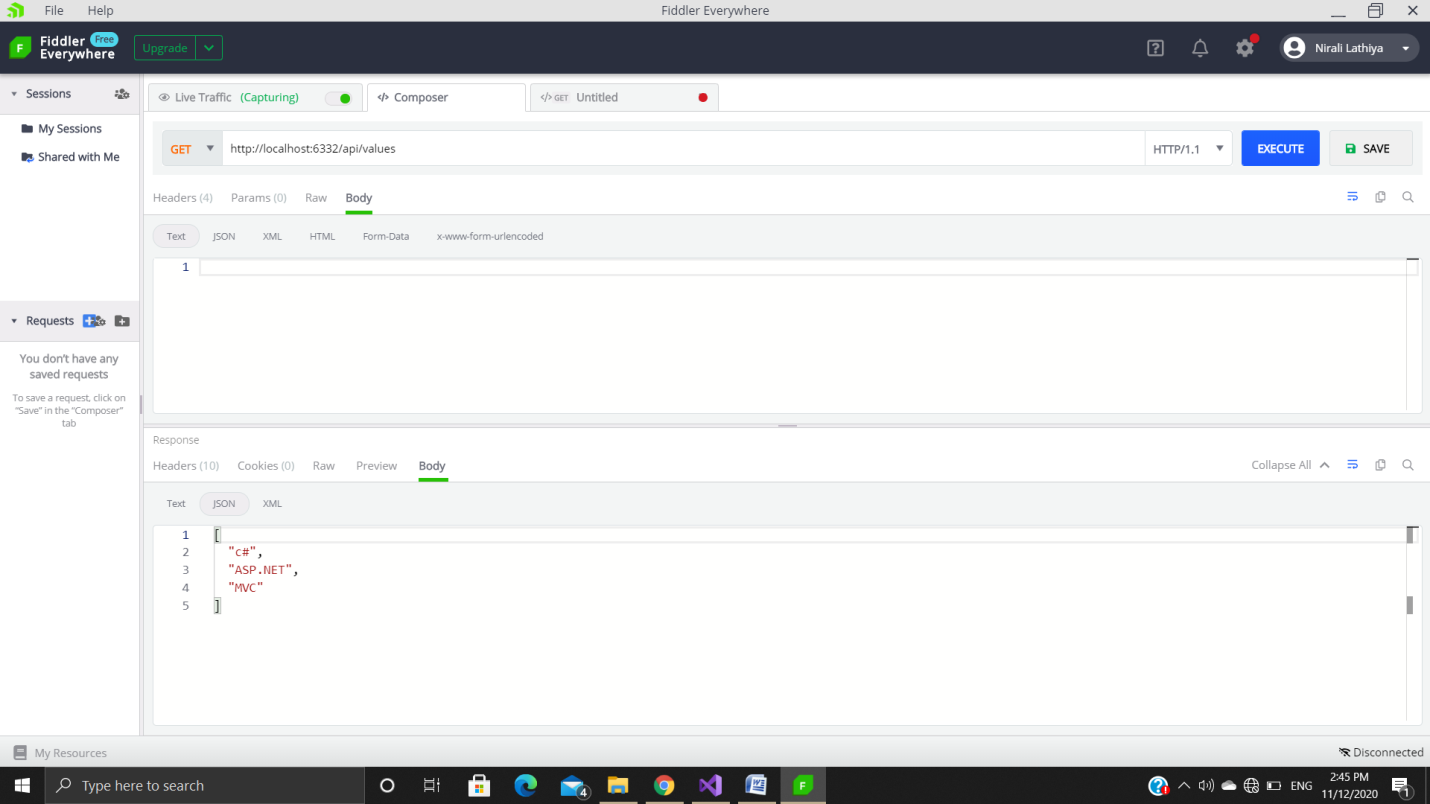


* Using Fiddler for GET, POST, PUT, DELETE action in Web API.
* **GET (** [**http://localhost:6332/api/values**](http://localhost:6332/api/values)**)**

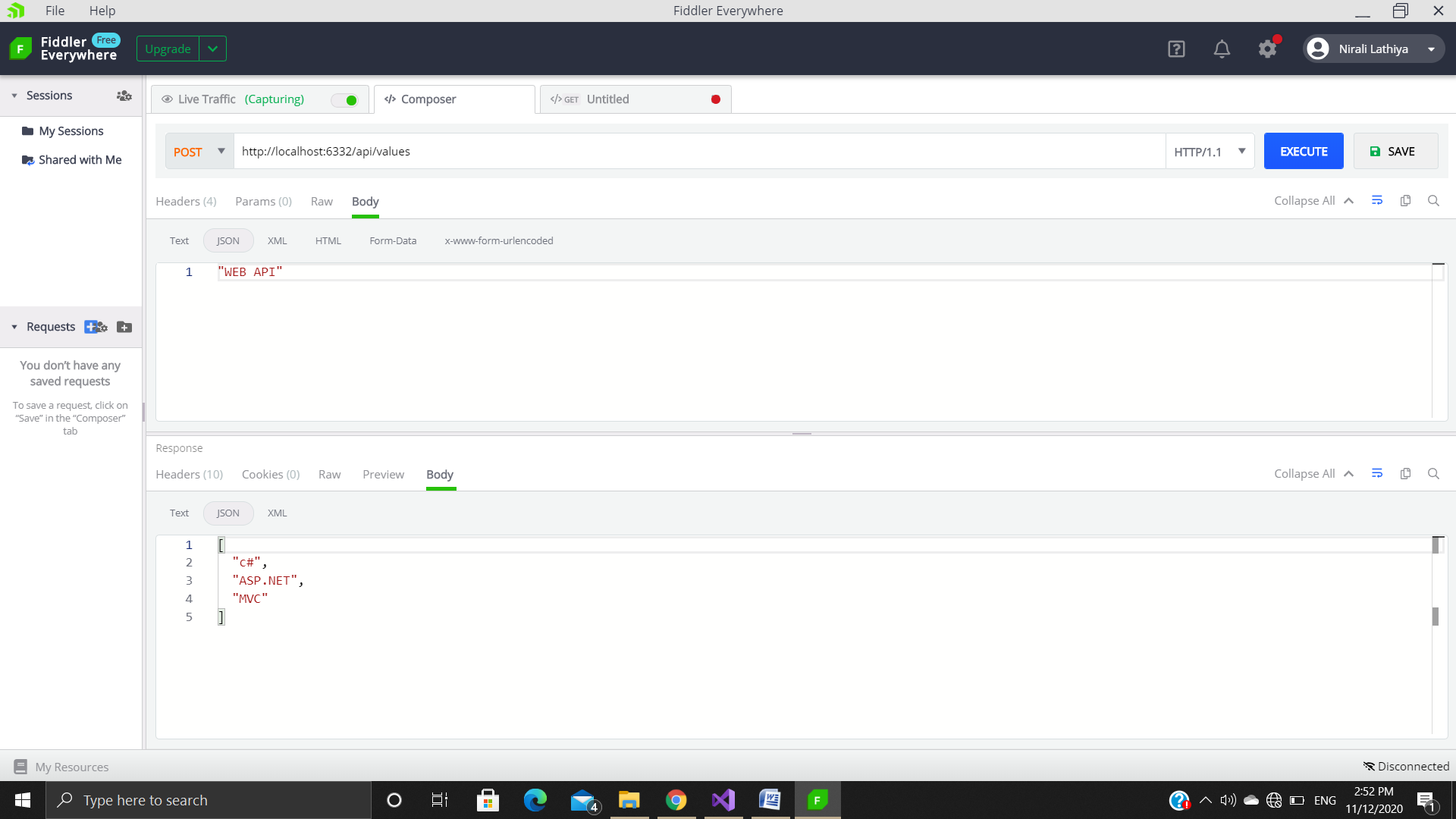
**In XML**



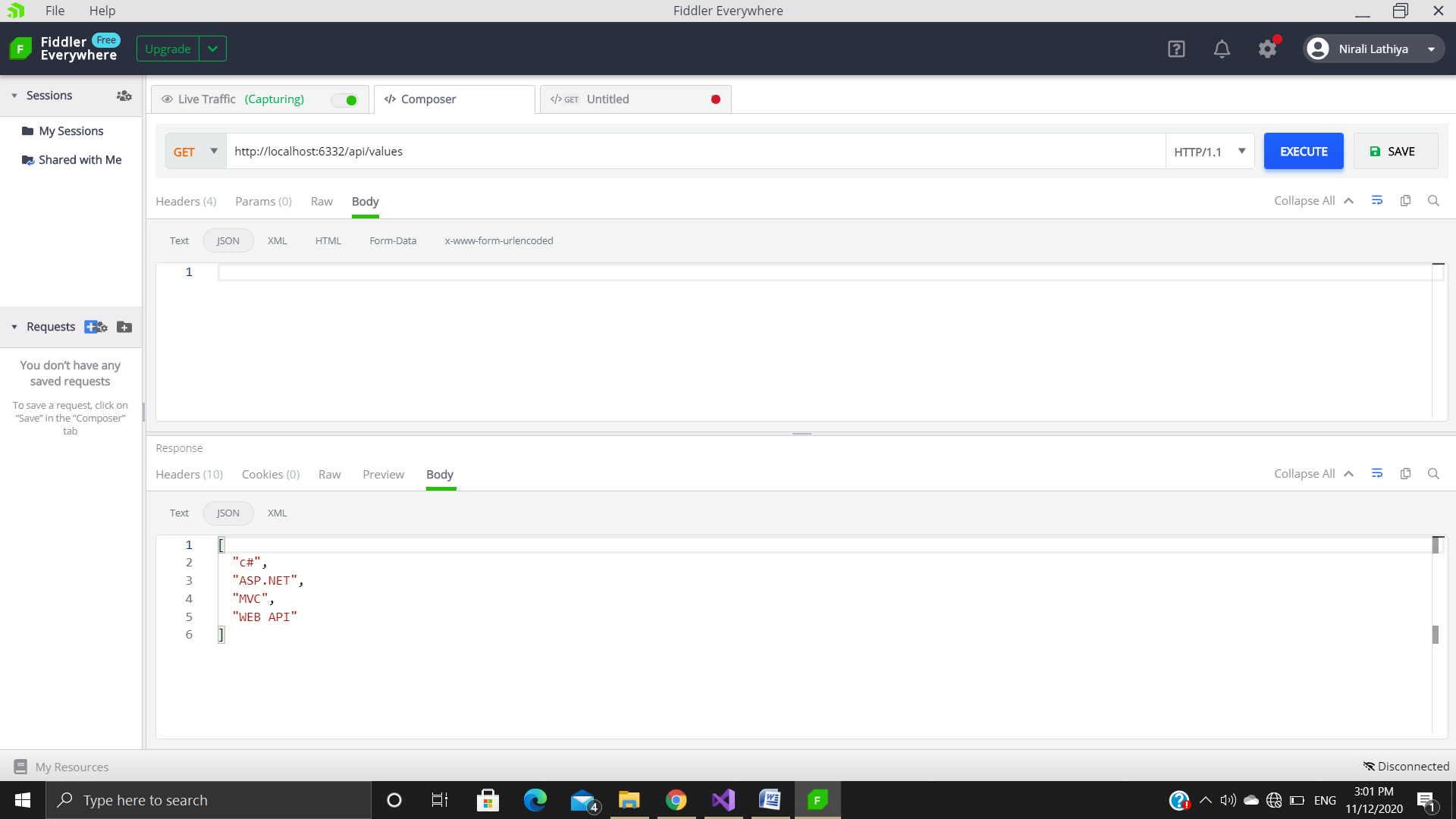
**In JSON**

****

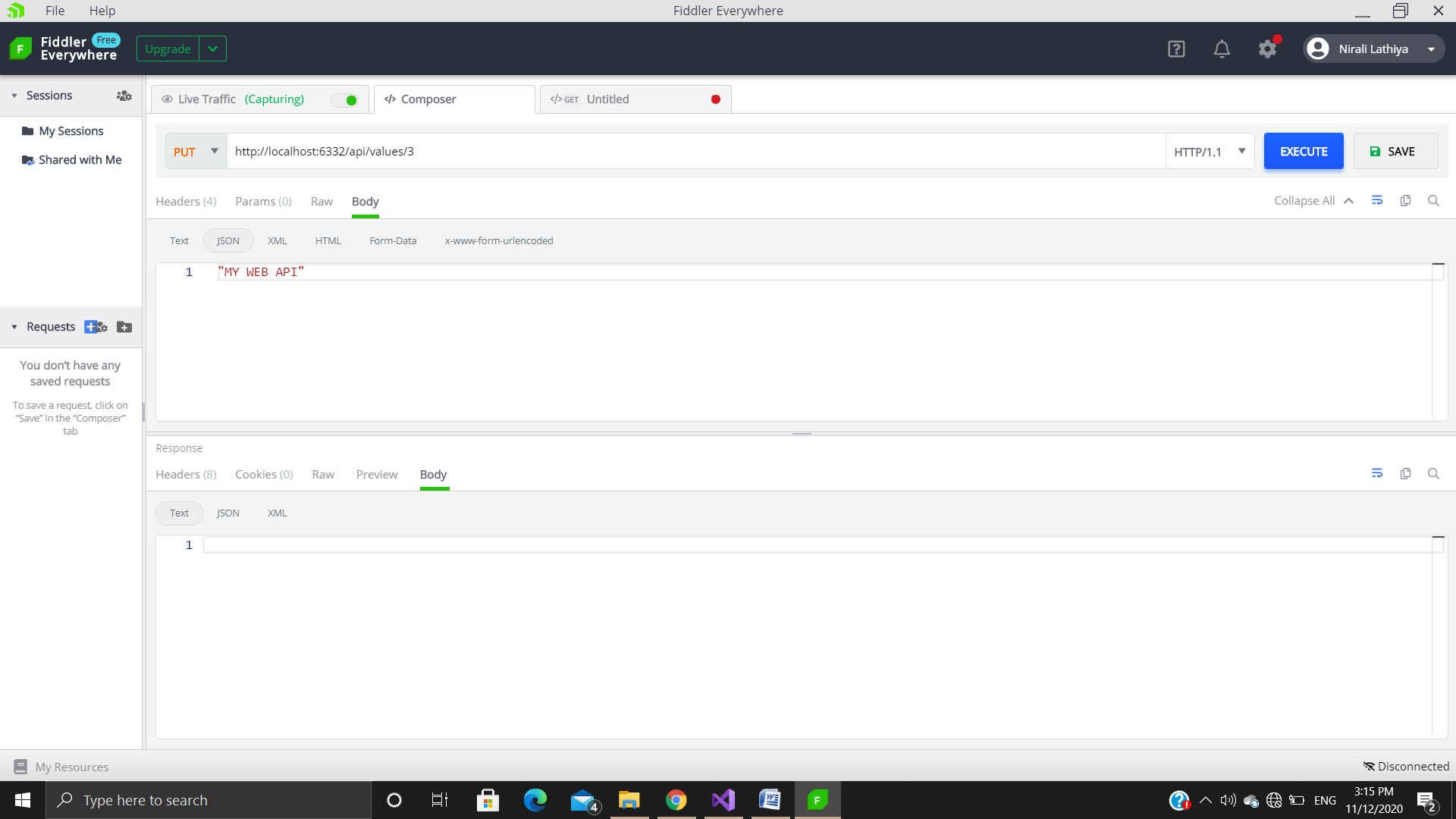
* **POST (** [**http://localhost:6332/api/values**](http://localhost:6332/api/values)**)**
* Now select the POST method and add new string or value in Request Body.
* We are adding the value “WEB API”



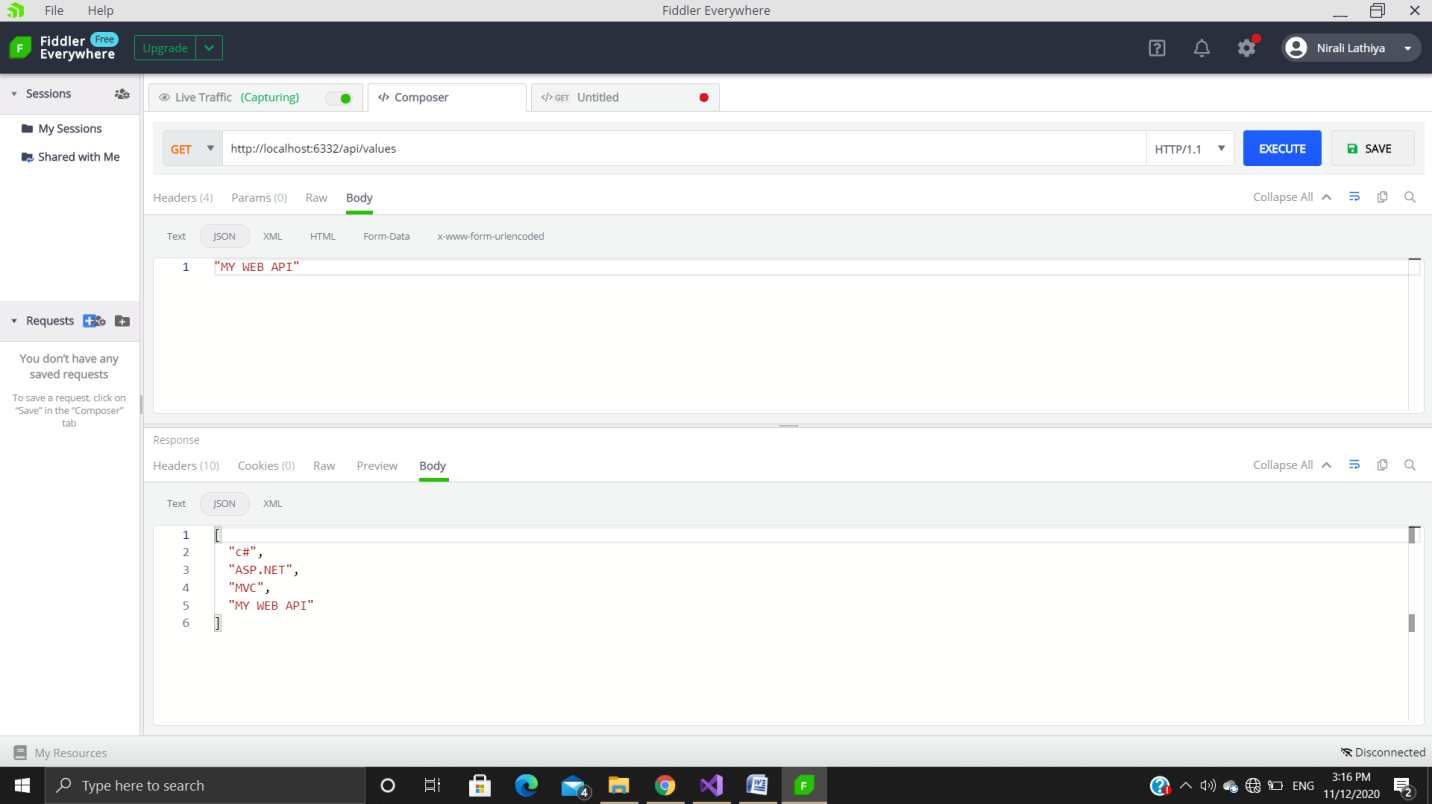
* Then Execute it.
* Now we can see as a response is no-cache in header because our return type is void
* Now we can check whether the string is added or not by using GET request
* Now we can see that new item is added in the list.



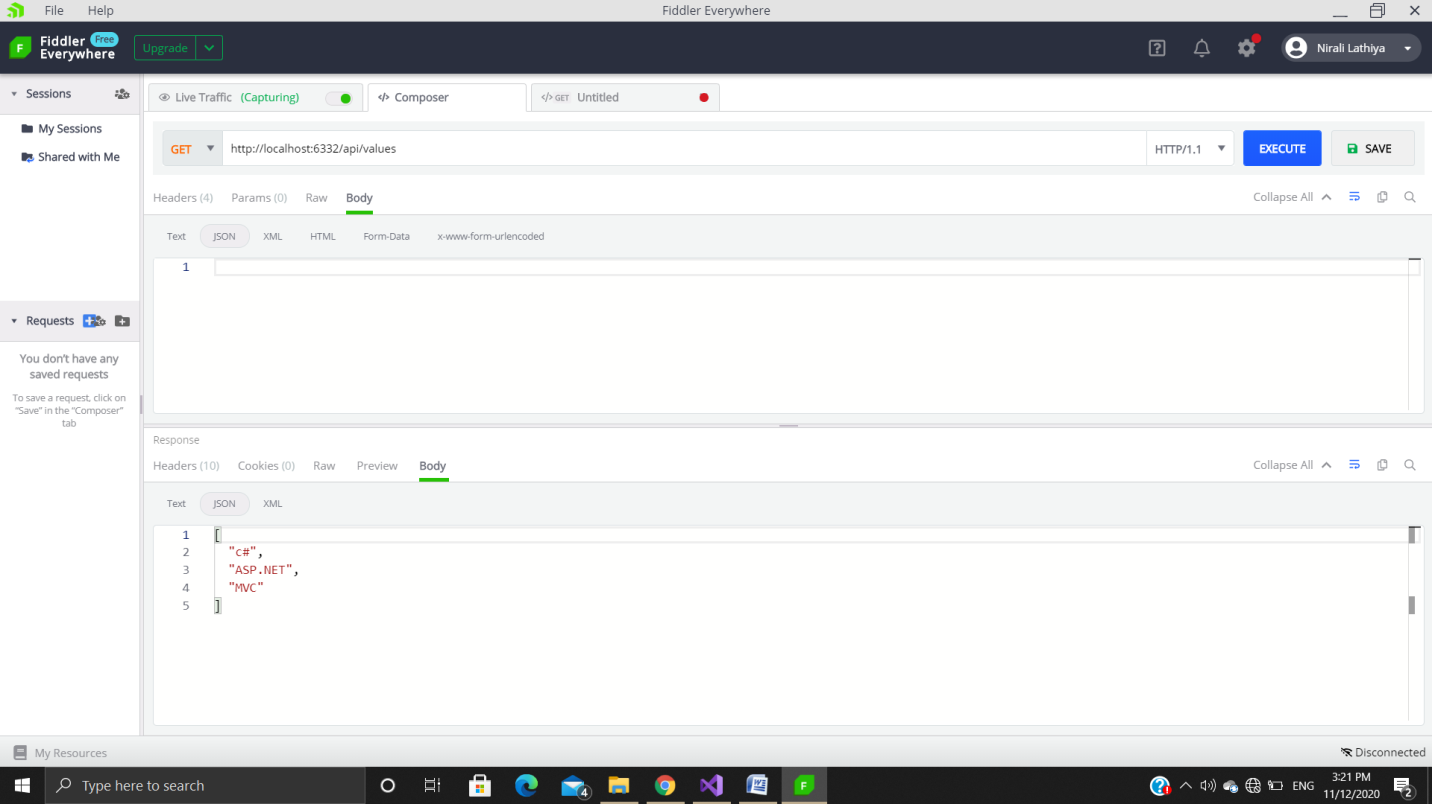
* **PUT (** [**http://localhost:6332/api/values/3**](http://localhost:6332/api/values/3)**)**
* Now we will update the string using PUT HTTP Verb.
* Select PUT method and type Where we want a new string to update the old string, followed by passing the index value to update the string in the URL and type the string in request body.
* Then click to execute button.
* WE are goanna to update at index 3 from “WEB API” to “MY WEB API”



* Now we can check whether the string is updated or not by using GET request
* Now we can see that new item is updated in the list.



* **DELETE (** [**http://localhost:6332/api/values/3**](http://localhost:6332/api/values/3)**)**
* We can delete the string from static variable.
* Now select the DELETE method and pass an index value which we want to delete
* Then click on execute.
* Then check by GET request whether the value is deleted or not.



1. **Understanding JSON structure**

* JSON states for JavaScript Object Notation
* It is a light weighted data interchange format.
* It is easy for humans to read and write, and also easy for machines to parse and generate.
* IT is a Data Representation format like XML and commonly used for APIs and Configs.
* It is light weighted and easy to read and write compare to XML and much cleaner because it is not including open and close tags.
* It can integrated easily with most languages.
* JSON supports two widely used data structures

1. Collection of name/value pairs

* Different programming languages supports this data structure in different names like object, record, struct, dictionary, hash table, keyed list, or associative array.

1. An ordered list of values

* In various programming languages, it is called as array, vector, list or sequence.
* JSON supports two widely used data structures
* Since data supported by Jason is also supported by most of the modern programming languages, it makes JSON a very useful data-interchange format.