

## CSC Assignment 1 Documentation

### Task 1

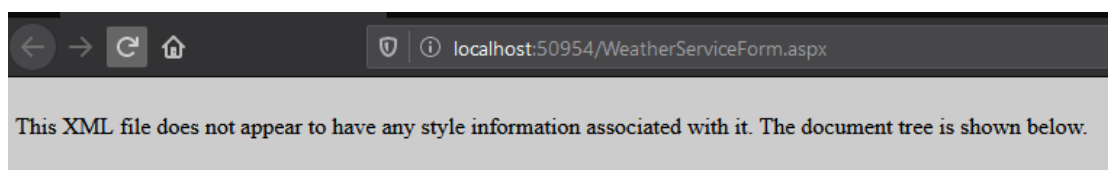
#### Browser (aspx)

*Located in WeatherService Folder*

Assign the country, number of days and insert your key in WeatherServiceForm.aspx.cs from <https://www.worldweatheronline.com/developer/>.

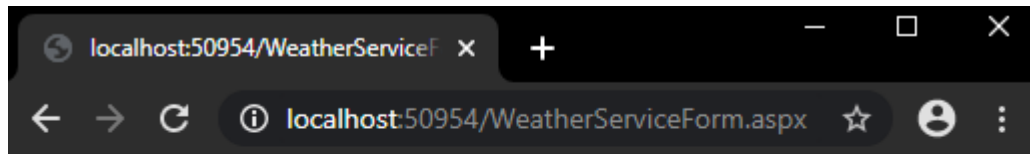
```
//Assign values for query
var country = "china";
var numOfDays = "5";
var key = "*****";
```

Debug run the application and you will see the following page on load.



```
<data>
  <request>
    <type>City</type>
    <query>Beijing, China</query>
  </request>
  <current_condition>
    <observation_time>05:03 AM</observation_time>
    <temp_C>6</temp_C>
    <temp_F>43</temp_F>
    <weatherCode>113</weatherCode>
    <weatherIconUrl>
      http://cdn.worldweatheronline.net/images/wsymbols01_png_64/wsymb01_0001_sunny.png
    </weatherIconUrl>
    <weatherDesc>Sunny</weatherDesc>
    <windspeedMiles>4</windspeedMiles>
    <windspeedKmph>7</windspeedKmph>
    <winddirDegree>320</winddirDegree>
    <winddir16Point>NW</winddir16Point>
    <precipMM>0.0</precipMM>
    <precipInches>0.0</precipInches>
    <humidity>26</humidity>
    <visibility>10</visibility>
    <visibilityMiles>6</visibilityMiles>
    <pressure>1029</pressure>
    <pressureInches>31</pressureInches>
    <cloudcover>0</cloudcover>
    <FeelsLikeC>5</FeelsLikeC>
    <FeelsLikeF>40</FeelsLikeF>
    <uvIndex>3</uvIndex>
  </current_condition>
```

In the case of error, the web service will ask u to verify your API key as well as your condition inputs.



## Error accessing web service

Reconnecting in 5s...



Please check the following:

- Network connection
- API key : XXXXXXXXXXXXXXXXXXXXXXXXXX
- Country selected : china
- Number of days selected : 5

Waiting for localhost...

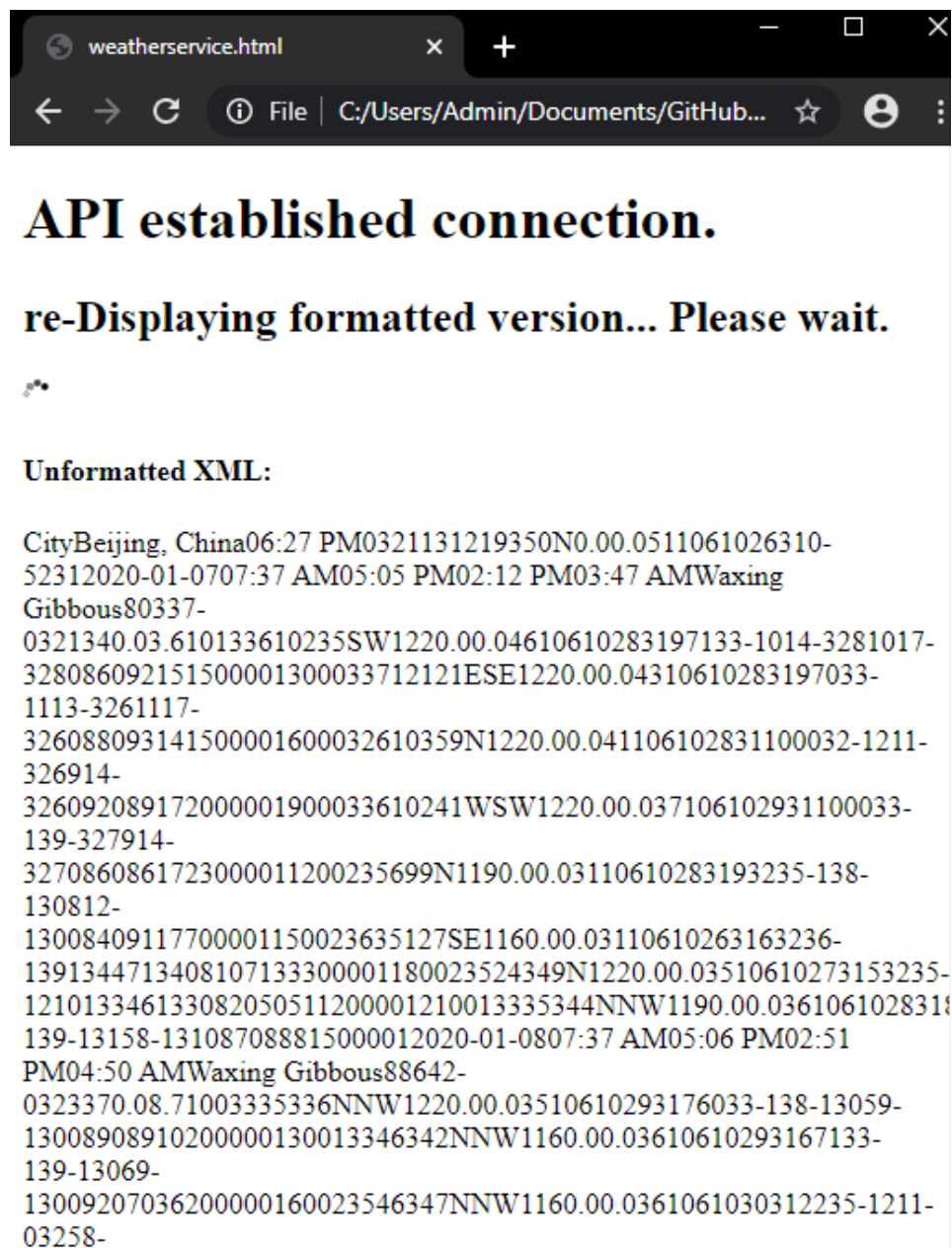
## Browser (JQuery)

Located in WeatherService\_js Folder

Assign the country, number of days and insert your key in script.js from <https://www.worldweatheronline.com/developer/> .

```
var country = "china";  
var numOfDays = "5";  
var key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
```

Debug run the application and you will see the following page on API successful connection.



The screenshot shows a web browser window with the title 'weatherservice.html'. The address bar shows the file path 'C:/Users/Admin/Documents/GitHub...'. The main content area displays the message 'API established connection.' followed by 're-Displaying formatted version... Please wait.' and a loading spinner. Below this, the text 'Unformatted XML:' is followed by a large block of unformatted XML data, which appears to be a list of weather forecasts for various locations in China, including Beijing, Wuxi, and others, with details like date, time, and weather conditions.

**API established connection.**

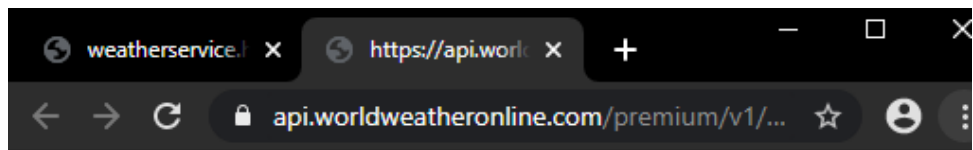
**re-Displaying formatted version... Please wait.**

•••

**Unformatted XML:**

CityBeijing, China06:27 PM0321131219350N0.00.0511061026310-52312020-01-0707:37 AM05:05 PM02:12 PM03:47 AMWaxing Gibbous80337-0321340.03.610133610235SW1220.00.04610610283197133-1014-3281017-328086092151500001300033712121ESE1220.00.04310610283197033-1113-3261117-326088093141500001600032610359N1220.00.041106102831100032-1211-326914-326092089172000001900033610241WSW1220.00.037106102931100033-139-327914-3270860861723000011200235699N1190.00.03110610283193235-138-130812-13008409117700001150023635127SE1160.00.03110610263163236-1391344713408107133300001180023524349N1220.00.03510610273153235-121013346133082050511200001210013335344NNW1190.00.036106102831139-13158-131087088815000012020-01-0807:37 AM05:06 PM02:51 PM04:50 AMWaxing Gibbous88642-0323370.08.71003335336NNW1220.00.03510610293176033-138-13059-13008908910200000130013346342NNW1160.00.03610610293167133-139-13069-13009207036200000160023546347NNW1160.00.0361061030312235-1211-03258-

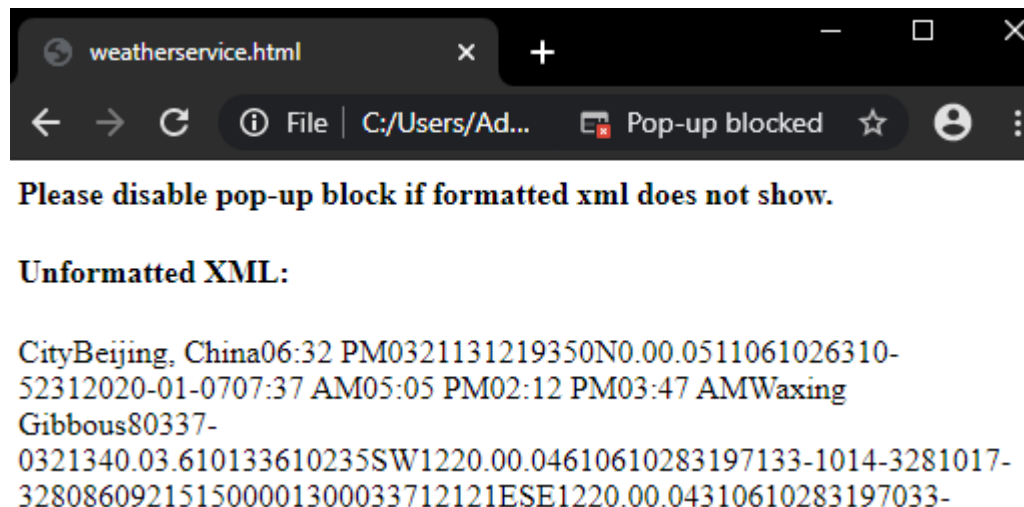
A formatted XML will be pop up in a new window. (Direct XML file from world weather)



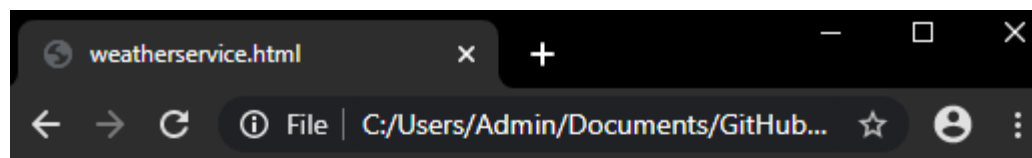
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<data>
  <request>
    <type>City</type>
    <query>Beijing, China</query>
  </request>
  <current_condition>
    <observation_time>06:29 PM</observation_time>
    <temp_C>0</temp_C>
    <temp_F>32</temp_F>
    <weatherCode>113</weatherCode>
    <weatherIconUrl>
      <![CDATA[
        http://cdn.worldweatheronline.net/images/wsymbols01_png_64/ws
      ]]>
    </weatherIconUrl>
    <weatherDesc>
      <![CDATA[ Clear ]]>
    </weatherDesc>
    <windspeedMiles>12</windspeedMiles>
    <windspeedKmph>19</windspeedKmph>
    <winddirDegree>350</winddirDegree>
    <winddir16Point>N</winddir16Point>
    <precipMM>0.0</precipMM>
    <precipInches>0.0</precipInches>
    <humidity>51</humidity>
    <visibility>10</visibility>
    <visibilityMiles>6</visibilityMiles>
    <pressure>1026</pressure>
    <pressureInches>31</pressureInches>
    <cloudcover>0</cloudcover>
    <FeelsLikeC>-5</FeelsLikeC>
    <FeelsLikeF>23</FeelsLikeF>
    <uvIndex>0</uvIndex>
  </current_condition>
```

In the case of formatted XML not popping up, enable or allow pop-up window.



In the case of error, client will reconnect and prompt to check common user input mistakes.



## API connection failed.

### Reconnecting in 5s...

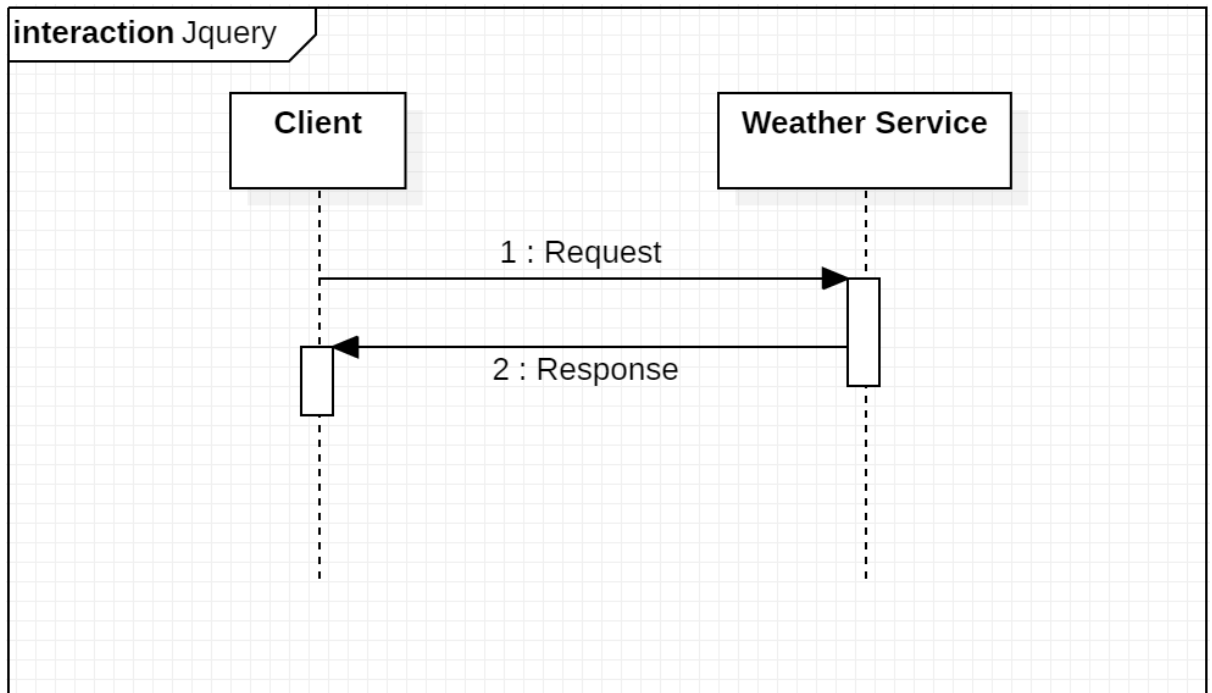


Please check the following:

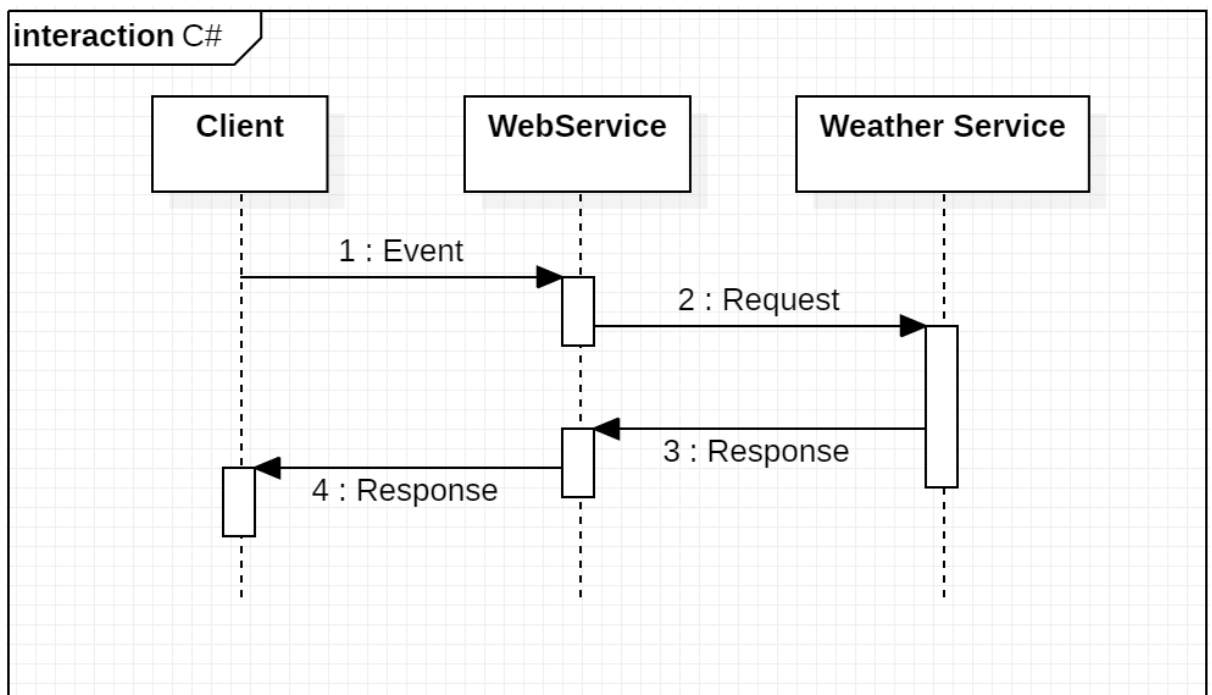
- Network connection
- API key
- Country selected
- Number of days selected

## Sequence Diagrams

Jquery:



aspx:

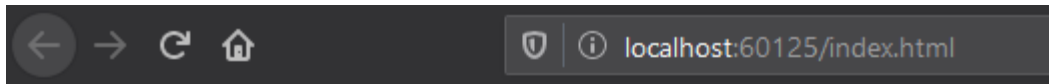


## Task 2

### Browser

Debug run the application and you will see the following page on load.

*(There will be a simulated loading)*



### All Products

- Tomato Soup: \$1
- Yo-yo: \$3.75
- Hammer: \$16.99

### Search by ID

Enter a number between 1 to 3 (ID for the products).

### Search by ID

Tomato Soup: \$1

In the case of error, client will reconnect and prompt to check.

### All Products

❌ Fail to get products. Reloading... (Check your Uri in the project)

### Search by ID

## Postman

Localhost:

<http://localhost:60125/>

APIs:

Version	Type	API Route
V1	GET	api/v1/products/version
V1	GET	api/v1/products/message?name1=<name>&name2=<name>&name3=<name>
V1	GET	api/v1/products
V1	GET	api/v1/products/{id}
V2	GET	api/v2/products
V2	GET	api/v2/products/{id}
V2	GET	api/v2/products?Category=<category>
V2	POST	api/v2/products
V2	PUT	api/v2/products/{id}
V2	DELETE	api/v2/products/{id}
V3	GET	api/v3/products
V3	GET	api/v3/products/{id}
V3	GET	api/v3/products?Category=<category>
V3	POST	api/v3/products
V3	PUT	api/v3/products/{id}
V3	DELETE	api/v3/products/{id}



GET: api/v1/products/version

Link: <http://localhost:25157/api/v1/products/version>

Get API version.

GET

▼

http://localhost:60125/api/v1/products/version

Params

Authorization

Headers (9)

Body ●

Pre-request S

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON ▼

1

2

3

4

5

[

"hello",

"version 1",

"1"

]

GET: api/v1/products/message?name1=<name>&name2=<name>&name3=<name>

Link: <http://localhost:60125/api/v1/products/message/name1=ji&name2=jii1&name3=ji3>

Get method with user defined parameters.

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `http://localhost:60125/api/v1/products/message?name1=ji&na...`, Send button.
- Params Tab:** Query Params table with columns KEY, VALUE, and DESCRIPTION.
- Response Bar:** Status: 200 OK, Time: 9ms, Size: 491 B, Save Response button.
- Body Tab:** Pretty view of the response body.

	KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/>	name1	ji		
<input checked="" type="checkbox"/>	name2	jii1		
<input checked="" type="checkbox"/>	name3	ji3		
	Key	Value	Description	

```
1 <html>
2
3 <body>Hello World name1 =ji name2= jii1 name3=ji3 is provided in path parameter</body>
4
5 </html>
```

GET: api/v1/products

Link: <http://localhost:60125/api/v1/products>

Get all products.

GET

http://localhost:60125/api/v1/products

Params

Authorization

Headers (9)

Body ●

Pre-request S

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON

```
1  [
2    {
3      "Id": 1,
4      "Name": "Tomato Soup",
5      "Category": "Groceries",
6      "Price": 1.0
7    },
8    {
9      "Id": 2,
10     "Name": "Yo-yo",
11     "Category": "Toys",
12     "Price": 3.75
13   },
14   {
15     "Id": 3,
16     "Name": "Hammer",
17     "Category": "Hardware",
18     "Price": 16.99
19   }
20 ]
```

GET: api/v1/products/{id}

Link: <http://localhost:60125/api/v1/products/1>

Get product based on id. (Available Id between 1 to 3)

GET ▼ http://localhost:60125/api/v1/products/1

Params Authorization Headers (9) Body ● Pre-request S

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (10) Test Results Status: 200 OK

Pretty Raw Preview Visualize BETA JSON ▼

```
1 {
2   "Id": 1,
3   "Name": "Tomato Soup",
4   "Category": "Groceries",
5   "Price": 1.0
6 }
```

Providing invalid id will lead to 404 Not Found.

GET ▼ http://localhost:60125/api/v1/products/5

Params Authorization Headers (9) Body ● Pre-request S

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (9) Test Results Status: 404 Not Found

Pretty Raw Preview Visualize BETA Text ▼

```
1 
```

GET: api/v2/products

Link: <http://localhost:60125/api/v2/products>

Get all products.

GET

http://localhost:60125/api/v2/products

Params

Authorization

Headers (9)

Body ●

Pre-request S

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON

```
1  [
2    {
3      "Id": 1,
4      "Name": "Tomato soup",
5      "Category": "Groceries",
6      "Price": 1.39
7    },
8    {
9      "Id": 2,
10     "Name": "Yo-yo",
11     "Category": "Toys",
12     "Price": 3.75
13   },
14   {
15     "Id": 3,
16     "Name": "Hammer",
17     "Category": "Hardware",
18     "Price": 16.99
19   }
20 ]
```

GET: api/v2/products/{id}

Link: <http://localhost:60125/api/V2/products>

Get product based on id. (Available Id between 1 to 3)

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:60125/api/v2/products/1
- Params:** Authorization, Headers (9), Body ●, Pre-request
- Query Params:** A table with 2 columns: KEY, VALUE. The first row contains 'Key' and 'Value'.
- Body:** Pretty, Raw, Preview, Visualize BETA, JSON ▼, and a refresh icon.
- Status:** 200 OK
- Response Body:**

```
1 {  
2   "Id": 1,  
3   "Name": "Tomato soup",  
4   "Category": "Groceries",  
5   "Price": 1.39  
6 }
```

Providing invalid id will lead to 400 Bad Request

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:60125/api/v2/products/5
- Params:** Authorization, Headers (9), Body ●, Pre-request S
- Query Params:** A table with 2 columns: KEY, VALUE. The first row contains 'Key' and 'Value'.
- Body:** Pretty, Raw, Preview, Visualize BETA, Text ▼, and a refresh icon.
- Status:** 400 Bad Request
- Response Body:**

```
1 No Product Id Found: 5
```

GET: api/v2/products?Category=<category>

Link: <http://localhost:60125/api/v2/products?Category=Toys>

Get products by Category as parameter.

GET ▼ http://localhost:60125/api/v2/products?Category=Toys

Params ● Authorization Headers (9) Body ● Pre-request

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	Category	Toys
	Key	Value

Body Cookies Headers (10) Test Results Status: 200 OK T

Pretty Raw Preview Visualize BETA JSON ▼

```
1 [
2   {
3     "Id": 2,
4     "Name": "Yo-yo",
5     "Category": "Toys",
6     "Price": 3.75
7   }
8 ]
```

POST: api/v2/products

Link: <http://localhost:60125/api/v2/products>

Create product using Post.

The screenshot displays a REST client interface with the following components:

- Request Section:**
  - Method: POST
  - URL: http://localhost:60125/api/v2/products
  - Body Type: raw (selected)
  - Body Content:

```
1 {  
2   "Name": "Potato soup",  
3   "Category": "Groceries",  
4   "Price": 2.00  
5 }
```
- Response Section:**
  - Status: 201 Created
  - Body Type: Pretty (selected)
  - Body Content:

```
1 {  
2   "Id": 4,  
3   "Name": "Potato soup",  
4   "Category": "Groceries",  
5   "Price": 2.00  
6 }
```



PUT: api/v2/products/{id}

Link: <http://localhost:60125/api/v2/products/2>

Change product details based on id using Put. (Available Id between 1 to 3)

The screenshot displays a REST client interface with the following components:

- Request Section:**
  - Method: PUT
  - URL: http://localhost:60125/api/v2/products/2
  - Body Type: raw (selected)
  - Body Content:

```
1 {  
2   "Name": "Car",  
3   "Category": "Toy",  
4   "Price": 100.00  
5 }
```
- Response Section:**
  - Status: 200 OK
  - Body Type: JSON (selected)
  - Body Content (Pretty):

```
1 {  
2   "Id": 2,  
3   "Name": "Car",  
4   "Category": "Toy",  
5   "Price": 100.00  
6 }
```

Providing invalid id will lead to 400 Bad Request

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:60125/api/v2/products/5
- Body:** A JSON object with the following properties:

```
{  "Name": "Car",  "Category": "Toy",  "Price": 100.00}
```
- Status:** 400 Bad Request
- Response Body:** No Product Id Found: 5

DELETE: api/v2/products/{id}

Link: <http://localhost:60125/api/v2/products/2>

Remove product based on id using Delete. (Available Id between 1 to 3)

DELETE

http://localhost:60125/api/v2/products/2

Params

Authorization

Headers (9)

Body ●

Pre-request

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON

```
1  [
2    {
3      "Id": 1,
4      "Name": "Tomato soup",
5      "Category": "Groceries",
6      "Price": 1.39
7    },
8    {
9      "Id": 3,
10     "Name": "Hammer",
11     "Category": "Hardware",
12     "Price": 16.99
13   }
14 ]
```

Providing invalid id will lead to 400 Bad Request

DELETE

http://localhost:60125/api/v2/products/5

Params

Authorization

Headers (9)

Body ●

Pre-request S

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 400 Bad Request

Pretty

Raw

Preview

Visualize BETA

Text

1

No Product Id Found: 5

GET: api/v3/products

Link: <http://localhost:60125/api/v3/products>

Get all products.

GET

http://localhost:60125/api/v3/products

Params

Authorization

Headers (9)

Body ●

Pre-request

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON

1

[

2

{

3

"Id": 1,

4

"Name": "Tomato soup",

5

"Category": "Groceries",

6

"Price": 1.39

7

},

8

{

9

"Id": 2,

10

"Name": "Yo-yo",

11

"Category": "Toys",

12

"Price": 3.75

13

},

14

{

15

"Id": 3,

16

"Name": "Hammer",

17

"Category": "Hardware",

18

"Price": 16.99

19

}

20

]

GET: api/v3/products/{id}

Link: <http://localhost:60125/api/v3/products/3>

Get product based on id. (Available Id between 1 to 3)

GET <http://localhost:60125/api/v3/products/3>

Params Authorization Headers (9) Body ● Pre-request

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (10) Test Results Status: 200 OK

Pretty Raw Preview Visualize BETA JSON ↺

```
1 {
2   "Id": 3,
3   "Name": "Hammer",
4   "Category": "Hardware",
5   "Price": 16.99
6 }
```

Providing invalid id will lead to 400 Bad Request

GET <http://localhost:60125/api/v3/products/5>

Params Authorization Headers (9) Body ● Pre-request

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (10) Test Results Status: 400 Bad Request

Pretty Raw Preview Visualize BETA Text ↺

```
1 No Product Id Found: 5
```

GET: api/v3/products?Category=<category>

Link: <http://localhost:60125/api/v3/products?Category=Groceries>

Get products by Category as parameter.

GET

http://localhost:60125/api/v3/products?Category=Groceries

Params

Authorization

Headers (9)

Body

Pre-request Script

Query Params

	KEY	VALUE	DE
<input checked="" type="checkbox"/>	Category	Groceries	
	Key	Value	D

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Time: 25ms

Pretty

Raw

Preview

Visualize BETA

JSON

```
1  [
2    {
3      "Id": 1,
4      "Name": "Tomato soup",
5      "Category": "Groceries",
6      "Price": 1.39
7    }
8  ]
```

POST: api/v3/products

Link: <http://localhost:60125/api/v3/products>

Create product using Post.

The screenshot displays a REST client interface with the following components:

- Request Section:**
  - Method: POST
  - URL: http://localhost:60125/api/v3/products
  - Body tab selected, showing raw JSON: 

```
{  "Name": "ABC",  "Category": "Toys",  "Price": 12}
```
- Response Section:**
  - Status: 201 Created
  - Body tab selected, showing pretty-printed JSON: 

```
{  "Id": 4,  "Name": "ABC",  "Category": "Toys",  "Price": 12.0}
```



“Under-posting” will lead to 400 Bad Request. (Required: Name, Price\*)

*\*Price must be between 0 to 100*

POST http://localhost:60125/api/v3/products

Params Authorization Headers (9) Body ● Pre-request Script Test

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● G

```
1 {
2   "Category": "Toys",
3   "Price": -12
4 }
```

Body Cookies Headers (10) Test Results Status: 400 Bad Request Time: 14ms

Pretty Raw Preview Visualize BETA JSON ▼

```
1 {
2   "Message": "The request is invalid.",
3   "ModelState": {
4     "item.Name": [
5       "The Name field is required."
6     ],
7     "item.Price": [
8       "The field Price must be between 0 and 100."
9     ]
10  }
11 }
```

“Over-posting” will lead to 201 Created. (“Over-post” values are excluded in creation)

The screenshot shows a REST client interface. At the top, a POST request is configured for the URL `http://localhost:60125/api/v3/products`. The 'Body' tab is selected, showing a JSON payload: 

```
{  "Name": "ABC",  "Category": "Toys",  "Price": 12,  "Discount": 5}
```

. Below the request, the response is displayed with a status of '201 Created'. The response body is shown in 'Pretty' JSON format: 

```
{  "Id": 5,  "Name": "ABC",  "Category": "Toys",  "Price": 12.0}
```

POST `http://localhost:60125/api/v3/products`

Params Authorization Headers (9) **Body** Pre-request S

● none ● form-data ● x-www-form-urlencoded ● **raw** ● b

```
1 {
2   "Name": "ABC",
3   "Category": "Toys",
4   "Price": 12,
5   "Discount": 5
6 }
```

Body Cookies Headers (11) Test Results Status: 201 Created

Pretty Raw Preview Visualize BETA JSON ↺

```
1 {
2   "Id": 5,
3   "Name": "ABC",
4   "Category": "Toys",
5   "Price": 12.0
6 }
```

^ “Discount” is excluded during creation

PUT: api/v3/products/{id}

Link: <http://localhost:60125/api/v3/products/3>

Change product details based on id using Put. (Available Id between 1 to 3)

The screenshot displays a REST client interface with a PUT request to `http://localhost:60125/api/v3/products/3`. The request body is a JSON object: `{ "Name": "Drill", "Category": "Hardware", "Price": 25.00 }`. The response status is `200 OK`, and the response body is a JSON object: `{ "Id": 3, "Name": "Drill", "Category": "Hardware", "Price": 25.00 }`.

PUT `http://localhost:60125/api/v3/products/3`

Params Authorization Headers (9) **Body** Pre-request

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ text

```
1 {
2   "Name": "Drill",
3   "Category": "Hardware",
4   "Price": 25.00
5 }
```

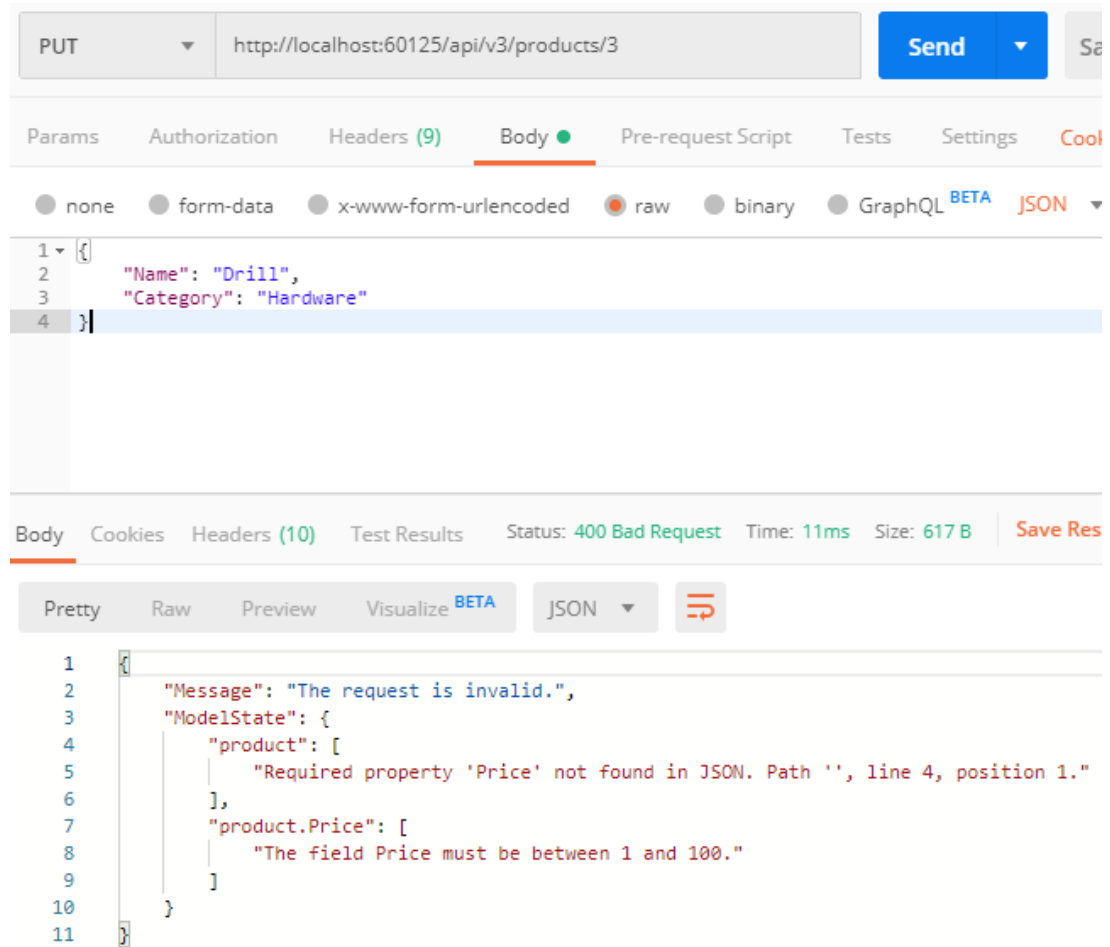
Body Cookies Headers (11) Test Results Status: 200 OK

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "Id": 3,
3   "Name": "Drill",
4   "Category": "Hardware",
5   "Price": 25.00
6 }
```

“Under-posting” will lead to 400 Bad Request. (Required: Name, Price\*)

*\*Price must be between 0 to 100*



PUT http://localhost:60125/api/v3/products/3

Send

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

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

```
1 {
2   "Name": "Drill",
3   "Category": "Hardware"
4 }
```

Body Cookies Headers (10) Test Results Status: 400 Bad Request Time: 11ms Size: 617 B Save Res

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "Message": "The request is invalid.",
3   "ModelState": {
4     "product": [
5       "Required property 'Price' not found in JSON. Path '', line 4, position 1."
6     ],
7     "product.Price": [
8       "The field Price must be between 1 and 100."
9     ]
10  }
11 }
```

“Over-posting” will lead to 200 OK. (“Over-post” values are excluded in creation)

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'PUT' and the URL is 'http://localhost:60125/api/v3/products/3'. Below this, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', and 'Pre-request S'. The 'Body' tab is selected, and the content type is set to 'raw'. The request body is a JSON object: 

```
1 {  
2   "Name": "Drill",  
3   "Category": "Hardware",  
4   "Price": 25.00,  
5   "Discount": 10  
6 }
```

 Below the request, there are tabs for 'Body', 'Cookies', 'Headers (11)', 'Test Results', and 'Status: 200 OK'. The 'Body' tab is selected, and the response is displayed in 'Pretty' format as JSON: 

```
1 {  
2   "Id": 3,  
3   "Name": "Drill",  
4   "Category": "Hardware",  
5   "Price": 25.00  
6 }
```

^ “Discount” is excluded during creation

Providing invalid id will lead to 400 Bad Request

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:60125/api/v3/products/5
- Body:** A JSON object with the following fields:

```
{  "Name": "Drill",  "Category": "Hardware",  "Price": 25.00}
```
- Status:** 400 Bad Request
- Response Body:** No Product Id Found: 5

DELETE: api/v3/products/{id}

Link: <http://localhost:60125/api/v1/products/1>

Remove product based on id using Delete. (Available Id between 1 to 3)

DELETE

http://localhost:60125/api/v3/products/1

Params

Authorization

Headers (9)

Body ●

Pre-request

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize BETA

JSON

1

[

2

{

3

"Id": 2,

4

"Name": "Yo-yo",

5

"Category": "Toys",

6

"Price": 3.75

7

},

8

{

9

"Id": 3,

10

"Name": "Drill",

11

"Category": "Hardware",

12

"Price": 25.00

13

}

14

]

Providing invalid id will lead to 400 Bad Request

DELETE

http://localhost:60125/api/v3/products/5

Params

Authorization

Headers (9)

Body ●

Pre-request S

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (10)

Test Results

Status: 400 Bad Request

Pretty

Raw

Preview

Visualize BETA

Text

1

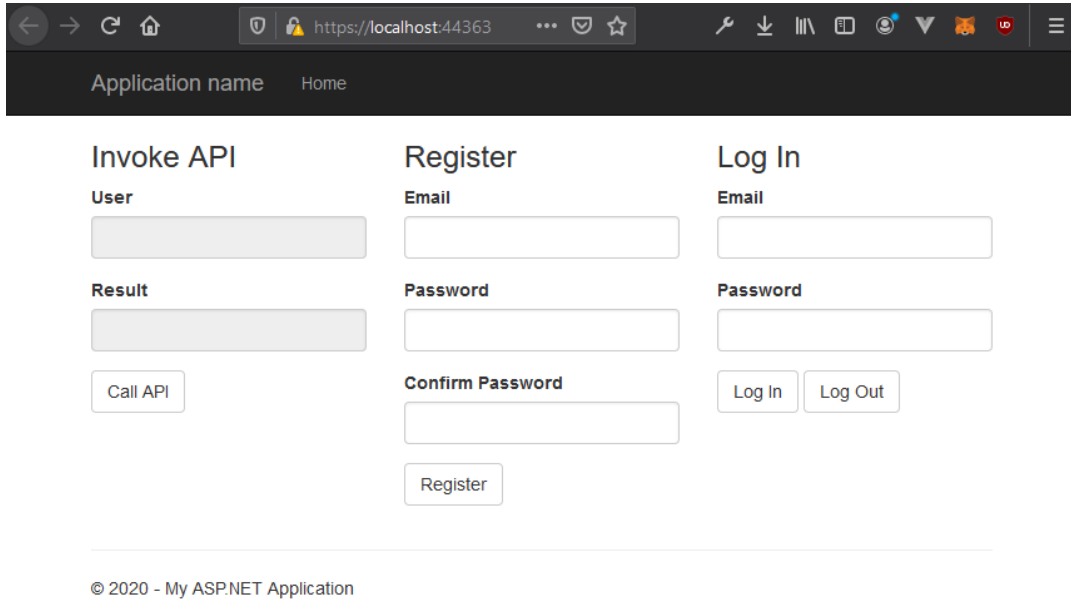
No Product Id Found: 5



## Task 3

### Browser

Debug run the application and you will see the following page on load.



The screenshot shows a web browser window with the address bar displaying `https://localhost:44363`. The browser's address bar includes navigation buttons (back, forward, refresh, home) and a search bar. The page content is divided into three main sections: **Invoke API**, **Register**, and **Log In**.

**Invoke API** section:

- User**: A text input field.
- Result**: A text input field.
- Call API**: A button.

**Register** section:

- Email**: A text input field.
- Password**: A text input field.
- Confirm Password**: A text input field.
- Register**: A button.

**Log In** section:

- Email**: A text input field.
- Password**: A text input field.
- Log In**: A button.
- Log Out**: A button.

At the bottom of the page, there is a footer that reads: © 2020 - My ASP.NET Application.

## Register

Register account by entering email and password twice.

### Invoke API

User

Result

Done!

Call API

### Register

Email

Password

Confirm Password

Register

### Log In

Email

Password

Log In

Log Out

Register can fail if email is already existed in the database.

### Invoke API

User

Result

400: Bad Request

Call API

The request is invalid.

Name admin@mail.com is already taken.

Email 'admin@mail.com' is already taken.

### Register

Email

Password

Confirm Password

Register

### Log In

Email

Password

Log In

Log Out

Register can fail if password fails the validation process.

## Invoke API

User

Result

400: Bad Request

Call API

The request is invalid.

The Password must be at least 6 characters long.

## Register

Email

admin@mail.com

Password

•••••

Confirm Password

•••••

Register

## Log In

Email

Password

Log In

Log Out

## Invoke API

User

Result

400: Bad Request

Call API

The request is invalid.

Passwords must have at least one non letter or digit character.  
Passwords must have at least one lowercase ('a'-'z'). Passwords must have at least one uppercase ('A'-'Z').

## Register

Email

admin@mail.com

Password

•••••••

Confirm Password

•••••••

Register

## Log In

Email

Password

Log In

Log Out

## Log in

Log in to the account by entering email and password. (Bearer token will be created)

### Invoke API

User

admin@mail.com

Result

Call API

### Register

Email

Password

Confirm Password

Register

### Log In

Email

admin@mail.com

Password

••••••••

Log In

Log Out

Log in can fail if email does not exist or password is incorrect.

### Invoke API

User

Result

400: Bad Request

Call API

invalid\_grant

The user name or password is incorrect.

### Register

Email

Password

Confirm Password

Register

### Log In

Email

admin@mail.com

Password

••••••••••••••

Log In

Log Out

## Call API

After logging in, user will be able to call API. (Authorized using bearer token)

### Invoke API

User

admin@mail.com

Result

Hello, admin@mail.com.

Call API

### Register

Email

Password

Confirm Password

Register

### Log In

Email

admin@mail.com

Password

••••••••

Log In

Log Out

API could not be called when user is not log on. (Due to lack of bearer token)

### Invoke API

User

Result

401: Unauthorized

Call API

Authorization has been denied for this request.

### Register

Email

Password

Confirm Password

Register

### Log In

Email

Password

Log In

Log Out

## Postman

Localhost:

<https://localhost:44363/>

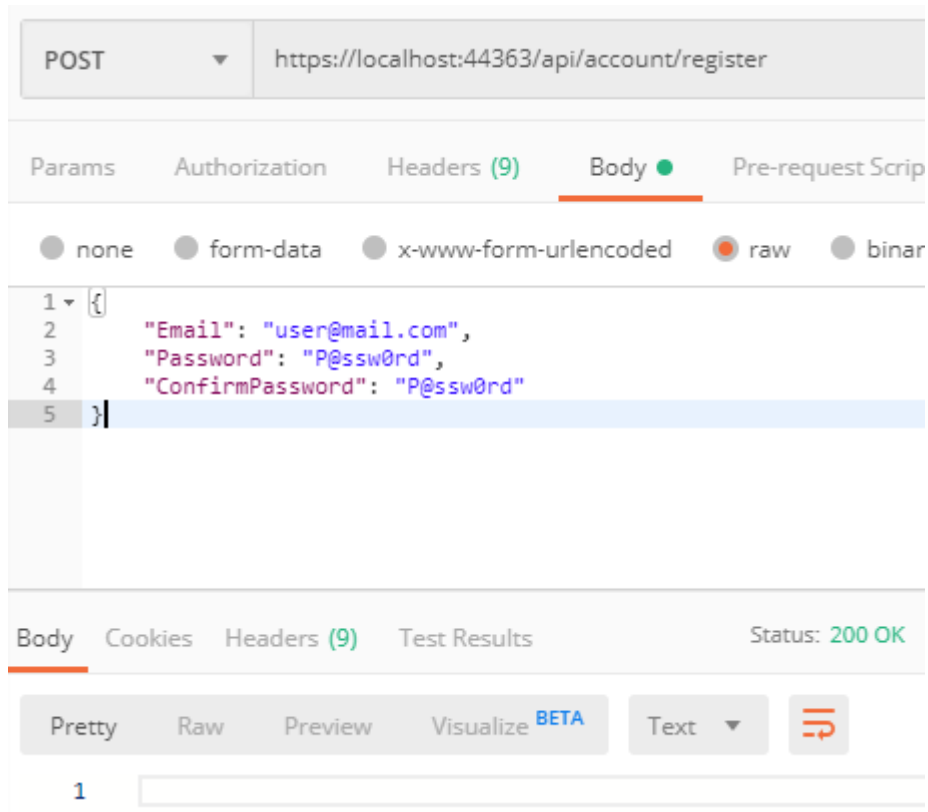
APIs:

Security Level	Type	API Route
Anonymous	POST	api/account/register
Anonymous	POST	token
Authorized	GET	api/values

POST: api/account/register

Link: <https://localhost:44363/api/account/register>

Register account by Post.



Exist email will lead to 400 Bad Request.

The screenshot shows a REST client interface with a POST request to `https://localhost:44363/api/account/register`. The request body is a JSON object with the following fields:

```
{
  "Email": "user@mail.com",
  "Password": "P@ssw0rd",
  "ConfirmPassword": "P@ssw0rd"
}
```

The response status is **400 Bad Request**. The response body is a JSON object indicating an invalid request due to an existing email:

```
{
  "Message": "The request is invalid.",
  "ModelState": {
    "": [
      "Name user@mail.com is already taken.",
      "Email 'user@mail.com' is already taken."
    ]
  }
}
```

The interface includes tabs for Params, Authorization, Headers (9), Body, and Pre-request Script. The Body tab is active, showing the request body. Below the request body, there are tabs for Body, Cookies, Headers (10), Test Results, and Status: 400 Bad Request. The Body tab is active, showing the response body in JSON format. The response body is displayed in a code editor with line numbers 1 through 9.

Fail model validation will lead to 400 Bad Request.

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** https://localhost:44363/api/account/register
- Body Tab:** Selected, showing a JSON request body:

```
1 {  
2   "Email": "user@mail.com",  
3   "Password": "P@ssw0rd",  
4   "ConfirmPassword": "password"  
5 }
```
- Response Tab:** Selected, showing a JSON response body:

```
1 {  
2   "Message": "The request is invalid.",  
3   "ModelState": {  
4     "model.ConfirmPassword": [  
5       "The password and confirmation password do not match."  
6     ]  
7   }  
8 }
```
- Status:** 400 Bad Request
- Time:** 25ms



## POST: token

Link: <https://localhost:44363/token>

Get token for API which need authorization by posting email and password. (Email is set as a ClaimType in the token)

The screenshot shows a REST client interface with a POST request to `https://localhost:44363/token`. The request body is `username=user@mail.com&password=P@ssw0rd&grant_type=password`. The response is a JSON object with the following fields:

```
1 {
2   "access_token":
3     "O7DjgAWaNG-Wi69N014dJzMKVKQtIV0aoD9EtVB1-G1HV4B8qY3ynsMpRL2Pf4J
4     JQrBr_0OGZo7blpAB5HstF_1ykkKf8i7z5GC941XbYEvn0NC4uIYF1pJU49Xjp5y
5     PD4nabbgo1_SFf4T-tsDncYwkf2X0N1VxeFBeRqS_OUQYFtRwpfgI4VvSz5moMMa
6     awjP9l6qPpQaTEi1PqEj5Ac8veRoRXVvJdb9Ajq4Tq1I3sgxN34eV5xLswfUkDs9
7     hLvAIqyEhvgH70gKJxoFv5k9J-fVnVLRccKZuG8WFyH9g2OoLYQk1oe5AUaZ23Sj
8     EgwGIId-DchZ52de9gOaP89AraVyFYNUc2BepfqHgU",
9   "token_type": "bearer",
10  "expires_in": 1209599,
11  "userName": "user@mail.com",
12  ".issued": "Fri, 03 Jan 2020 16:54:56 GMT",
13  ".expires": "Fri, 17 Jan 2020 16:54:56 GMT"
```

Email that does not exist or incorrect password will lead to 400 Bad Request.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `https://localhost:44363/token`
- Body Tab:** Selected, showing a single line of raw text: `username=user@mail.com&password=P@ssw0rd&grant_type=password`
- Body Type:** raw (selected)
- Status:** 400 Bad Request
- Time:** 27ms
- Response Body (JSON):**

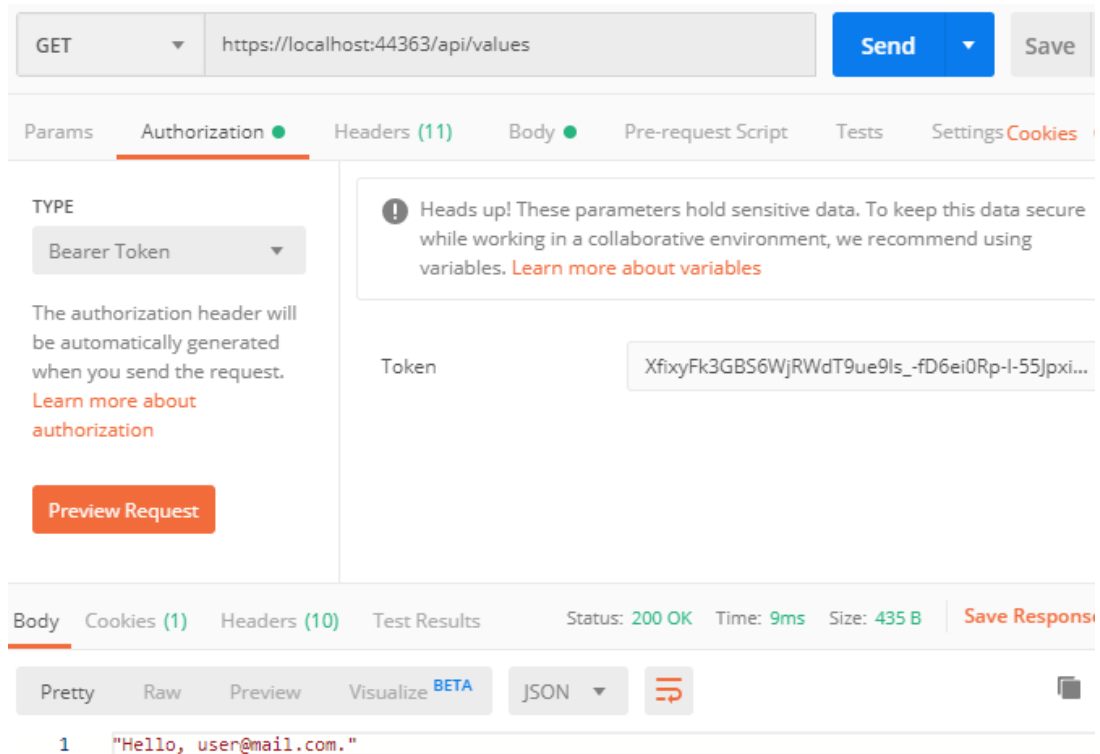
```
1 {
2   "error": "invalid_grant",
3   "error_description": "The user name or password is incorrect."
4 }
```

GET: api/values

Link: <https://localhost:44363/api/values>

(Requires Bearer token)

Get request with bearer token. (Email is derived from a ClaimType in the token)



The screenshot displays the Swagger client interface for a REST API. At the top, the method is set to GET and the URL is `https://localhost:44363/api/values. The 'Authorization' tab is selected, showing a 'Bearer Token' type. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The token value is XfixyFk3GBS6WjRWdT9ue9ls_-fD6ei0Rp-l-55Jpxi.... A 'Preview Request' button is visible. The bottom section shows the response body as "Hello, user@mail.com." with a status of 200 OK, time of 9ms, and size of 435 B.`

GET `https://localhost:44363/api/values` **Send** **Save**

Params **Authorization** Headers (11) Body ● Pre-request Script Tests Settings **Cookies**

**TYPE**  
Bearer Token ▼


The authorization header will be automatically generated when you send the request.  
[Learn more about authorization](#)

**Preview Request**

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Token `XfixyFk3GBS6WjRWdT9ue9ls_-fD6ei0Rp-l-55Jpxi...`

**Body** **Cookies (1)** **Headers (10)** Test Results Status: 200 OK Time: 9ms Size: 435 B **Save Response**

Pretty Raw Preview Visualize **BETA** JSON ▼ 

```
1 "Hello, user@mail.com."
```

Get request without bearer token will lead to 403 Unauthorized .

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `https://localhost:44363/api/values`, Buttons: Send, Save.
- Request Configuration:** Params, Authorization (selected), Headers (10), Body, Pre-request Script, Tests, Settings, Cookies.
- Authorization Tab:**
  - TYPE: Inherit auth from pa... (dropdown)
  - Text: "The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)"
  - Message: "This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper."
- Response Bar:** Body (selected), Cookies (1), Headers (11), Test Results, Status: 401 Unauthorized, Time: 8ms, Size: 509 B, Save Response.
- Response Body:** Pretty, Raw, Preview, Visualize BETA, JSON (selected), Copy icon.
- Response Content:**

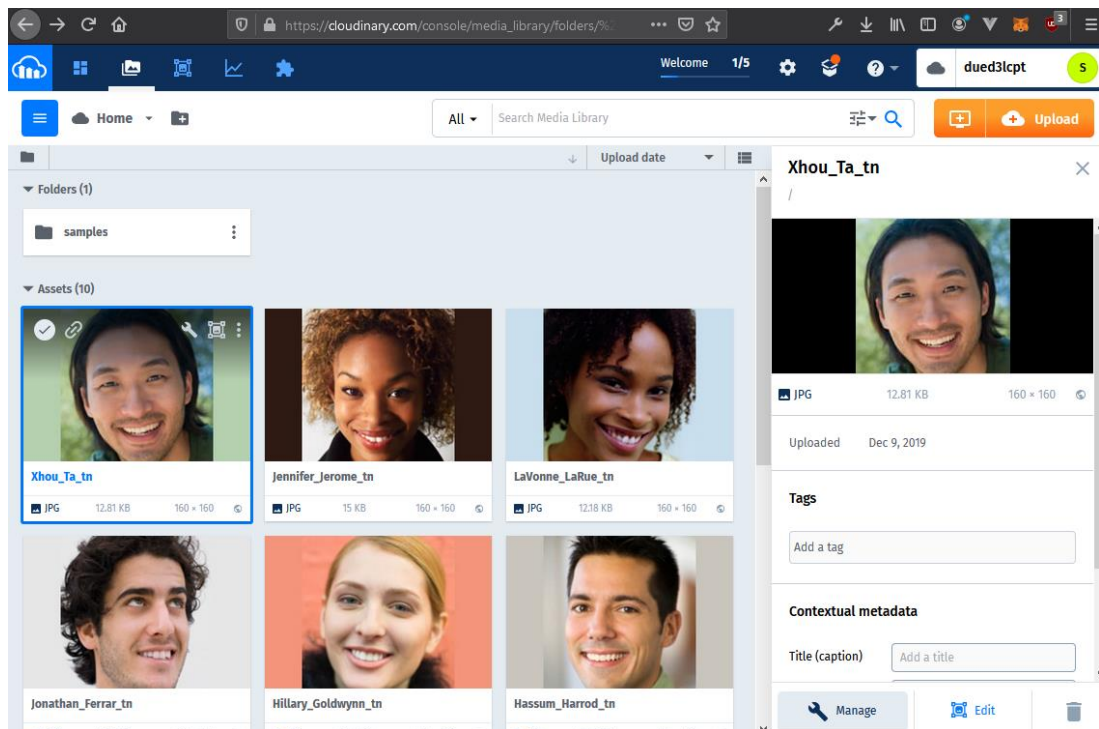
```
1 {
2   "Message": "Authorization has been denied for this request."
3 }
```

## Task 4

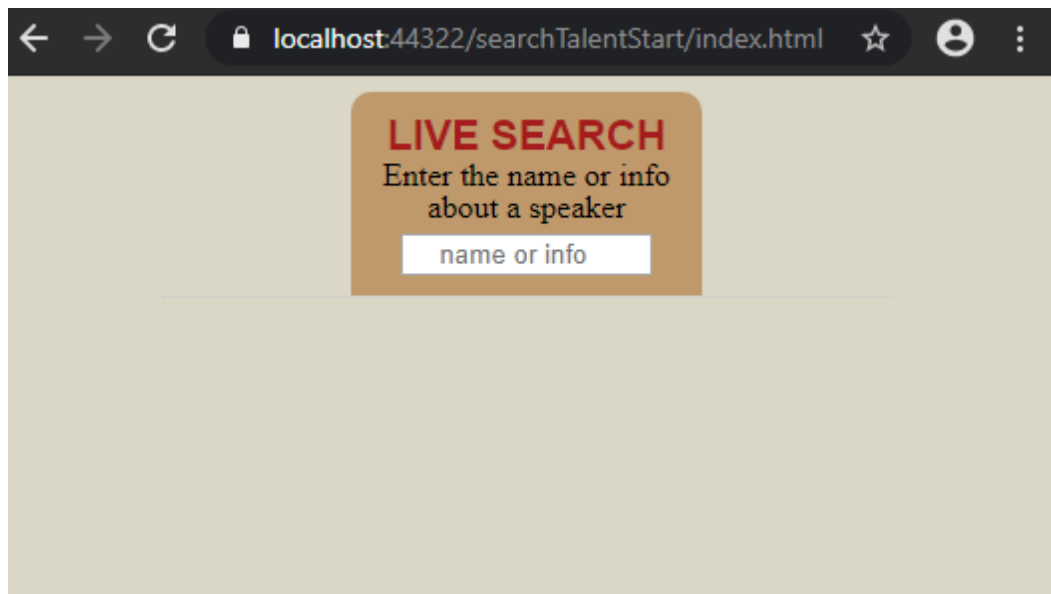
### Browser

Assign your link in searchTalentStart/script.js from the Cloudinary media library  
<https://cloudinary.com/>.

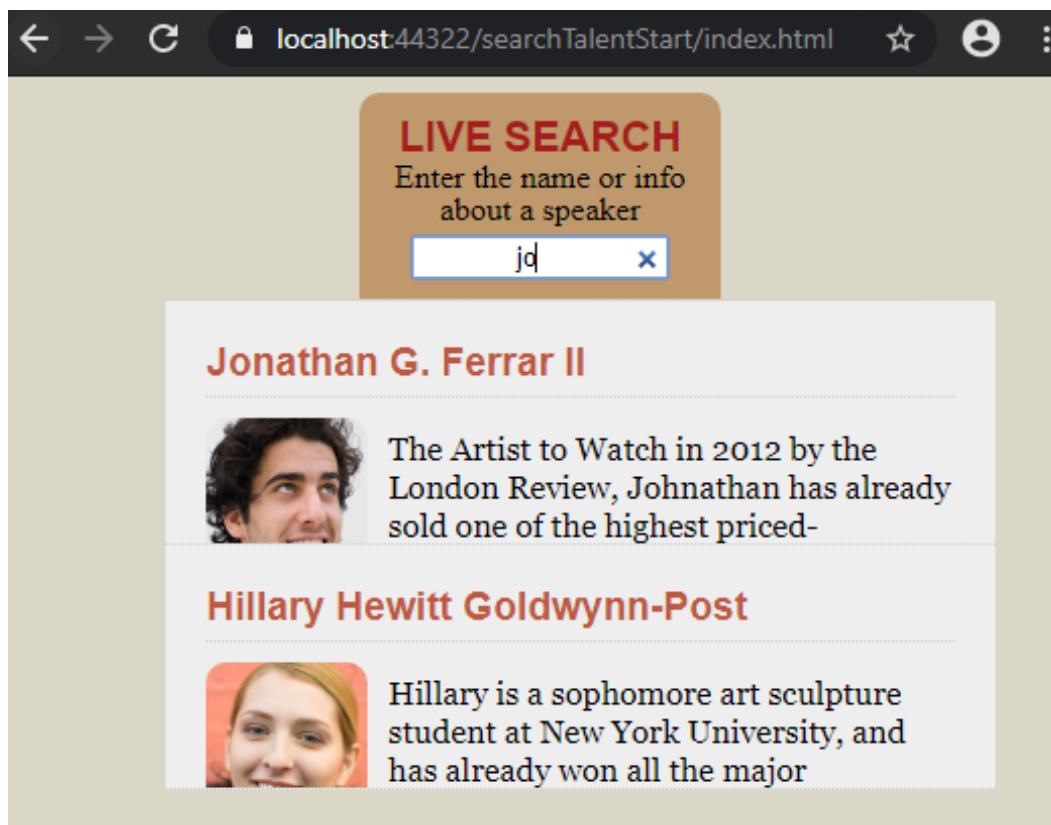
```
//Url for the Cloud image hosting  
var urlForCloudImage = "https://res.cloudinary.com/XXXXXXX/image/upload/XXXXXXX/";
```



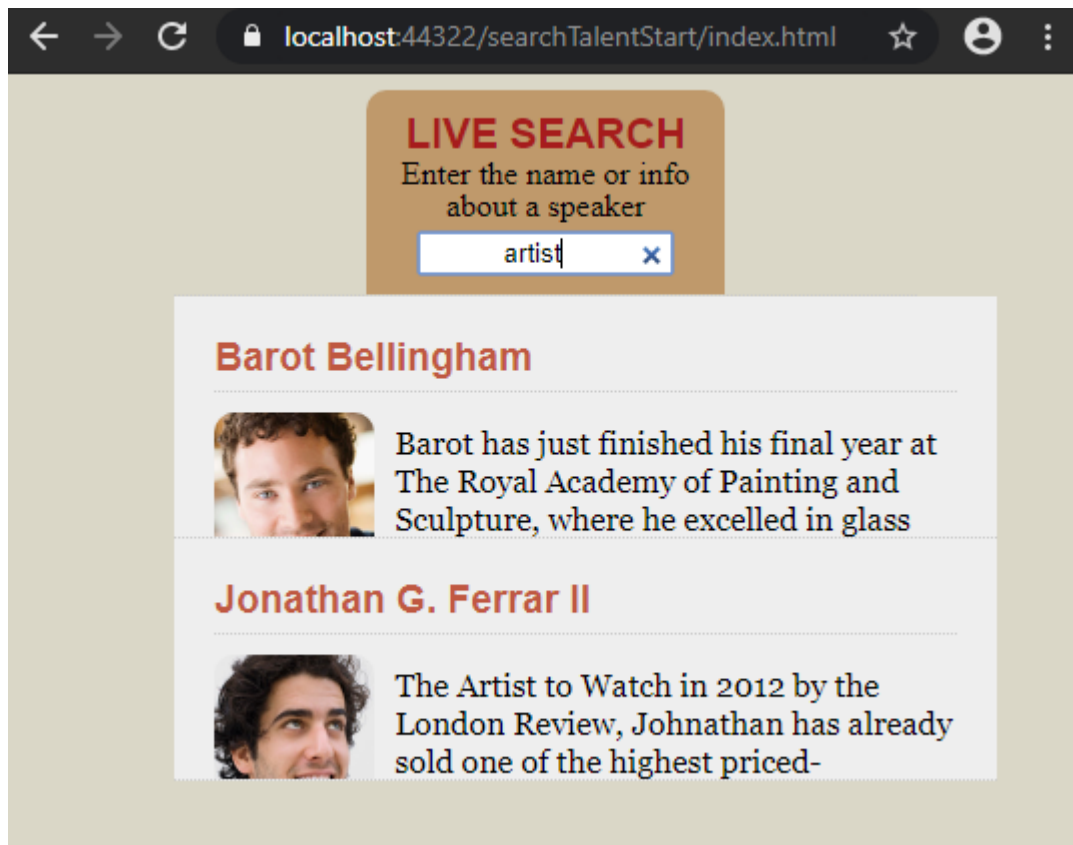
Debug run the application and you will see the following page on load.



You can search by typing the name.

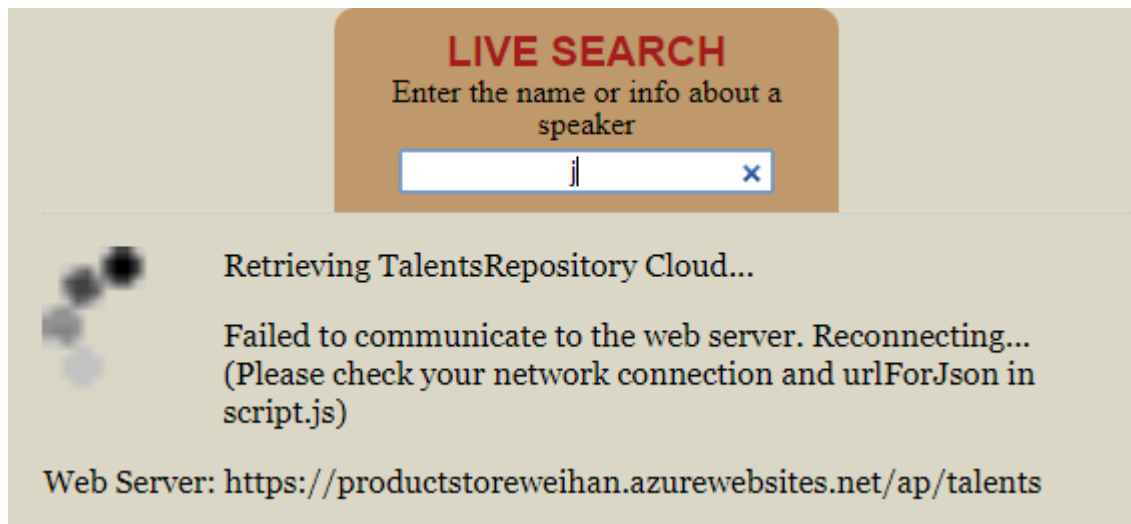


You can also search by typing the info.



The screenshot shows a web browser window with the address bar displaying 'localhost:44322/searchTalentStart/index.html'. The page features a 'LIVE SEARCH' section with the prompt 'Enter the name or info about a speaker'. A search input field contains the text 'artist'. Below the search bar, two search results are displayed. The first result is for 'Barot Bellingham', accompanied by a small portrait photo and the text: 'Barot has just finished his final year at The Royal Academy of Painting and Sculpture, where he excelled in glass'. The second result is for 'Jonathan G. Ferrar II', also with a small portrait photo and the text: 'The Artist to Watch in 2012 by the London Review, Johnathan has already sold one of the highest priced-'.

In the case of error, client will reconnect and prompt to check.



The screenshot shows the same web browser window, but the search results area now displays an error message. At the top, the 'LIVE SEARCH' section is still visible with the input field containing 'j'. Below this, there is a loading spinner icon (three dots) and the text: 'Retrieving TalentsRepository Cloud...'. This is followed by an error message: 'Failed to communicate to the web server. Reconnecting... (Please check your network connection and urlForJson in script.js)'. At the bottom, the 'Web Server' URL is listed as 'https://productstoreweihan.azurewebsites.net/ap/talents'.

## Postman

Localhost:

<https://localhost:44322/>

Cloud Host:

<https://productstoreweihan.azurewebsites.net>

APIs:

Type	API Route
GET	api/talents
GET	api/talents/{id}



GET: api/talents

Link: <https://localhost:44322/api/talents>

Get all talents

GET

https://localhost:44322/api/talents

Send

Save

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

C

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk E
Key	Value	Description		

Body

Cookies (1)

Headers (10)

Test Results

Status: 200 OK

Time: 13ms

Size: 2.49 KB

Save Response

Pretty

Raw

Preview

Visualize BETA

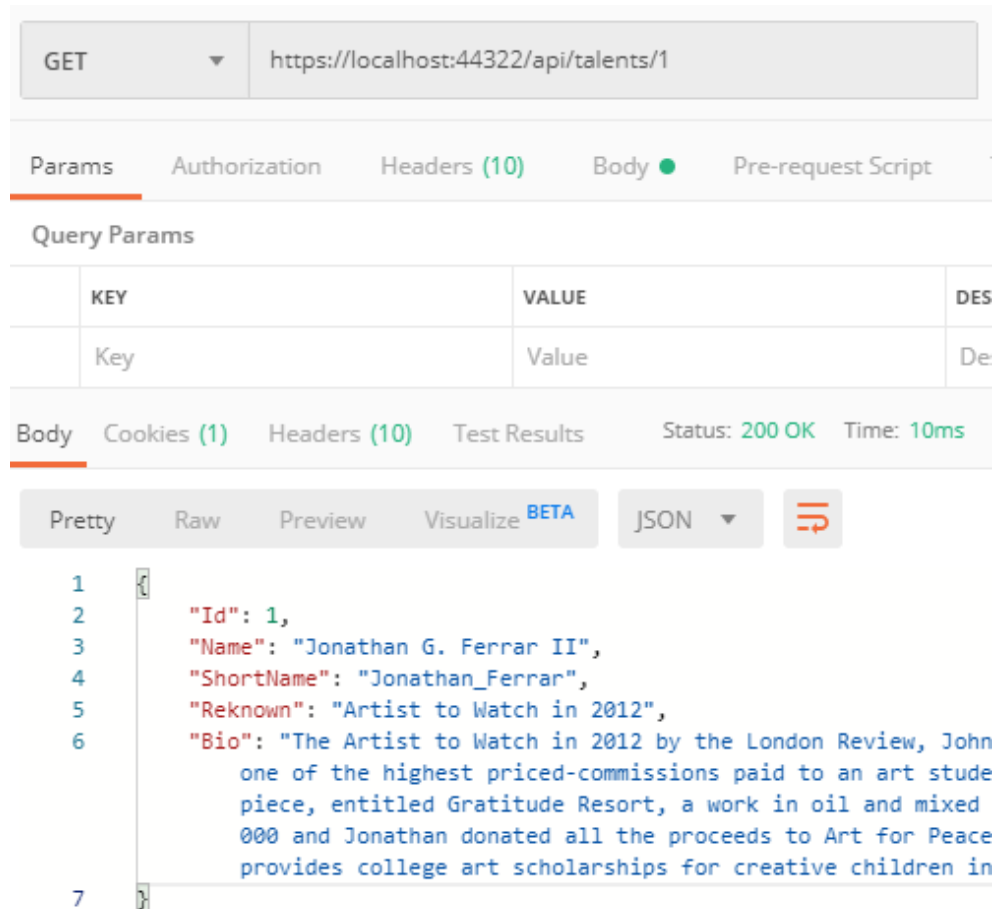
JSON

```
1  {
2    {
3      "Id": 0,
4      "Name": "Barot Bellingham",
5      "ShortName": "Barot_Bellingham",
6      "Reknown": "Royal Academy of Painting and Sculpture",
7      "Bio": "Barot has just finished his final year at The Royal Academy of Painting and
8        Sculpture, where he excelled in glass etching paintings and portraiture. Hailed
9        as one of the most diverse artists of his generation, Barot is equally as
10       skilled with watercolors as he is with oils, and is just as well-balanced in
11       different subject areas. Barot's collection entitled \"The Un-Collection\" will
12       adorn the walls of Gilbert Hall, depicting his range of skills and
13       sensibilities - all of them, uniquely Barot, yet undeniably different"
14     },
15     {
16       "Id": 1,
17       "Name": "Jonathan G. Ferrar II",
18       "ShortName": "Jonathan_Ferrar",
19       "Reknown": "Artist to Watch in 2012",
20       "Bio": "The Artist to Watch in 2012 by the London Review, Johnathan has already
21         sold one of the highest priced-commissions paid to an art student, ever on
22         record. The piece, entitled Gratitude Resort, a work in oil and mixed media,
23         was sold for $750,000 and Jonathan donated all the proceeds to Art for Peace,
24         an organization that provides college art scholarships for creative children in
25         developing nations"
26     },
27     {
28       "Id": 2,
29       "Name": "Hillary Hewitt Goldwynn-Post",
30       "ShortName": "Hillary_Goldwynn",
31       "Reknown": "New York University",
32       "Bio": "Hillary is a sophomore art sculpture student at New York University, and
33         has already won all the major international prizes for new sculptors, including
34         the Divinity Circle, the International Sculptor's Medal, and the Academy of
35         Paris Award. Hillary's CAC exhibit features 25 abstract watercolor paintings
36         that contain only water images including waves, deep sea, and river."
37     },
38     {
39       "Id": 3,
40       "Name": "Hassum Harrod",
41       "ShortName": "Hassum_Harrod",
42       "Reknown": "Art College in New Dehli",
43       "Bio": "The Art College in New Dehli has sponsored Hassum on scholarship for his
44         entire undergraduate career at the university, seeing great promise in his
45         contemporary paintings of landscapes - that use equal parts muted and vibrant
46         tones, and are almost a contradiction in art. Hassum will be speaking on \"The
47         use and absence of color in modern art\" during Thursday's agenda."
48     }
49   }
50 }
```

GET: api/talents/{id}

Link: <https://localhost:44322/api/talents/1>

Get talent based on Id. (Available Id between 0 to 3)




GET ▼ <https://localhost:44322/api/talents/1>

Params Authorization Headers (10) Body ● Pre-request Script

Query Params

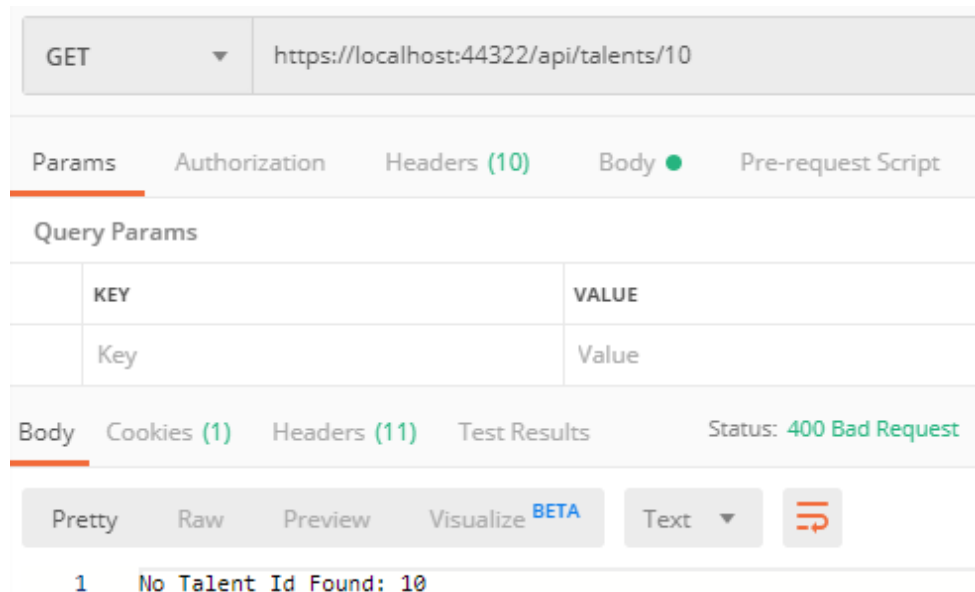
	KEY	VALUE	DES
	Key	Value	De

Body Cookies (1) Headers (10) Test Results Status: 200 OK Time: 10ms

Pretty Raw Preview Visualize BETA JSON ▼ 

```
1 {  
2   "Id": 1,  
3   "Name": "Jonathan G. Ferrar II",  
4   "ShortName": "Jonathan_Ferrar",  
5   "Reknown": "Artist to Watch in 2012",  
6   "Bio": "The Artist to Watch in 2012 by the London Review, John  
       one of the highest priced-commissions paid to an art stude  
       piece, entitled Gratitude Resort, a work in oil and mixed  
       000 and Jonathan donated all the proceeds to Art for Peace  
       provides college art scholarships for creative children in  
7 }
```

Providing invalid id will lead to 400 Bad Request




GET ▼ <https://localhost:44322/api/talents/10>

Params Authorization Headers (10) Body ● Pre-request Script

Query Params

	KEY	VALUE
	Key	Value

Body Cookies (1) Headers (11) Test Results Status: 400 Bad Request

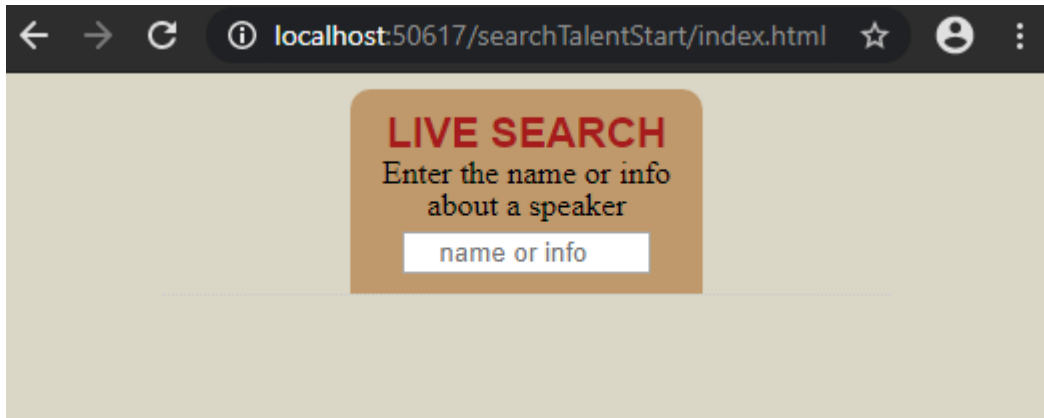
Pretty Raw Preview Visualize BETA Text ▼ 

```
1 No Talent Id Found: 10
```

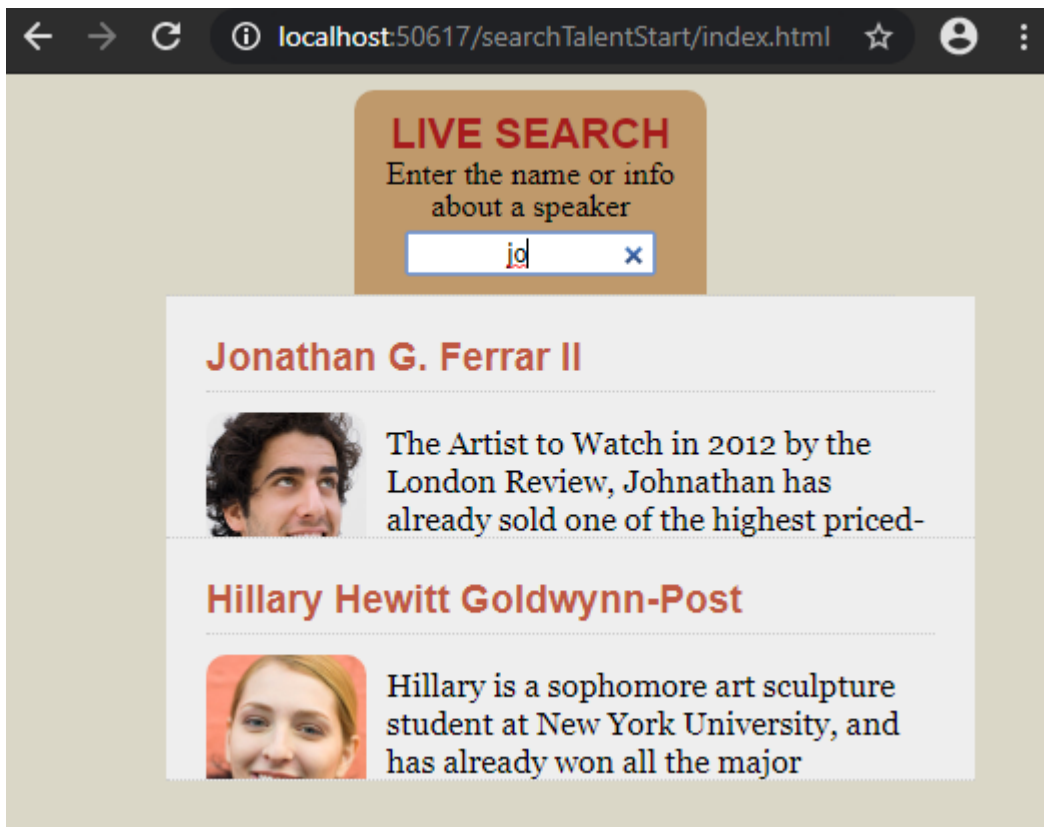
## Task 5

### Browser

Debug run the application and you will see the following page on load.




You can search by typing the name.



In the case of error, client will reconnect and prompt to check.

**LIVE SEARCH**  
Enter the name or info about a speaker



Retrieving TalentsRepository Cloud...  
Failed to establish API connection. Reconnecting...  
(Please check your network connection and urlForJson in script.js)  
API Call: /api/talets

Using the browser console, we can see that https is used for data in transit.

Status	Method	Domain	File	Cause	Type	Transferred	Size
302	GET	localhost:50617	talents	xhr	json	2.53 KB	2.09 KB
200	OPTIONS	localhost:44357	talents	xhr	plain	458 B	0 B
200	GET	localhost:44357	talents	xhr	json	2.54 KB	2.09 KB

3 requests

4.17 KB / 5.51 KB transferred

Finish: 14 ms

Headers

Cookies

Params

Response

Timings

Stack Trace

Request URL: http://localhost:50617/api/talents

Request Method: GET

Remote Address: [::1]:50617

Status Code: 302 Found

Version: HTTP/1.1

Referrer Policy: no-referrer-when-downgrade

Edit and Resend

Status	Method	Domain	File	Cause	Type	Transferred	Size
302	GET	localhost:50617	talents	xhr	json	2.53 KB	2.09 KB
200	OPTIONS	localhost:44357	talents	xhr	plain	458 B	0 B
200	GET	localhost:44357	talents	xhr	json	2.54 KB	2.09 KB

3 requests

4.17 KB / 5.51 KB transferred

Finish: 14 ms

Headers

Cookies

Params

Response

Timings

Stack Trace

Security

Request URL: https://localhost:44357/api/talents

Request Method: GET

Remote Address: [::1]:44357

Status Code: 200 OK

Version: HTTP/2

Referrer Policy: no-referrer-when-downgrade

Edit and Resend

Filter Headers

Response Headers (460 B)

Raw Headers

Request Headers (386 B)

Raw Headers

Accept: \*/\*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.5

Connection: keep-alive

DNT: 1

Host: localhost:44357

Origin: http://localhost:50617

Referer: http://localhost:50617/searchTalentStart/index.html

TE: Trailers

User-Agent: Mozilla/5.0 (Windows NT 10.0; ...) Gecko/20100101 Firefox/72.0

X-Requested-With: XMLHttpRequest

## Fiddler (Postman)

Localhost:

<http://localhost:50617/>

<https://localhost:44357/>

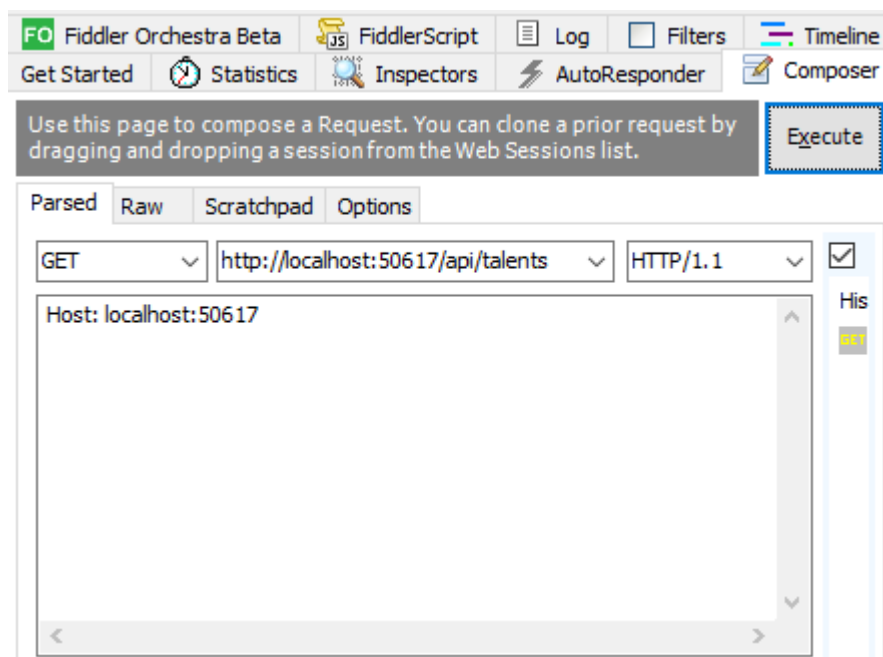
APIs:

Type	API Route
GET	api/talents
GET	api/talents/{id}

GET: api/talents

Link: <http://localhost:50617/api/talents>

Get all talents using http will result a redirect to https



Progress Telerik Fiddler Web Debugger

File Edit Rules Tools View Help

#	Result	Protocol	Host	URL	Content-Type
1	302	HTTP	localhost:50617	/api/talents	text/html; charset=utf-8
2	200	HTTPS	localhost:44357	/api/talents	application/json; charset=utf-8

1. Header shows transport to <https://localhost:44357> with response 302 Found.

1 302 HTTP localhost:50617 /api/talents text/html; charset=utf-8

The screenshot displays the Fiddler Orchestra Beta interface. The top toolbar includes buttons for FiddlerScript, Log, Filters, Timeline, Get Started, Statistics, Inspectors, AutoResponder, and Composer. Below the toolbar, the 'Inspectors' tab is active, showing the 'Request Headers' section. The request is a GET to /api/talents on localhost:50617. The 'Response Headers' section shows a 302 Found status. The 'Entity' section displays the response body: <p>Use https instead of http</p>.

**Request Headers** [Raw] [Header Definitions]

GET /api/talents HTTP/1.1

**Transport**  
Host: localhost:50617

Transformer Headers TextView SyntaxView ImageView HexView  
WebView Auth Caching Cookies Raw JSON XML

**Response Headers** [Raw] [Header Definitions]

HTTP/1.1 302 Found

**Cache**  
Cache-Control: no-cache  
Date: Sat, 04 Jan 2020 12:02:26 GMT  
Expires: -1  
Pragma: no-cache

**Entity**  
Content-Length: 32  
Content-Type: text/html; charset=utf-8

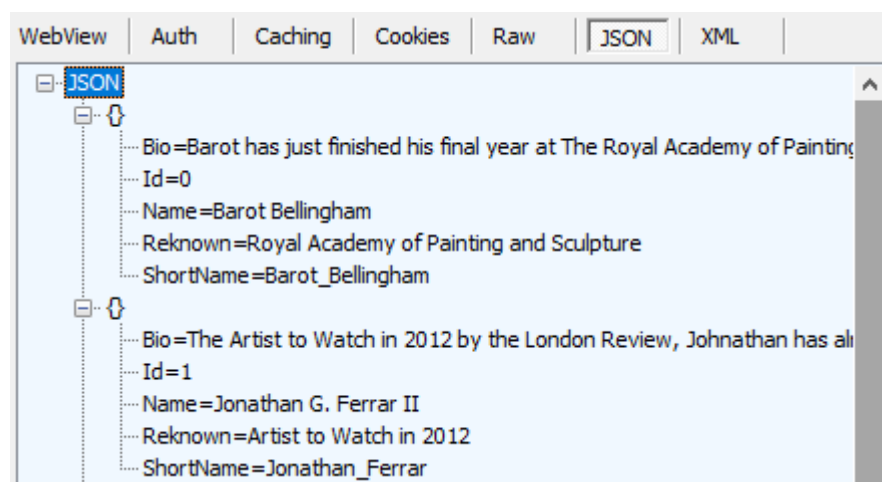
**Miscellaneous**  
Server: Microsoft-IIS/10.0  
X-AspNet-Version: 4.0.30319  
X-Powered-By: ASP.NET  
X-SourceFiles: =?UTF-8?B?QzpcVXNlcncQWRtaW5cRG9jdW1lbnRzXEEdpdEh1YlxDU0U0

**Transport**  
Location: <https://localhost:44357/api/talents>

Transformer Headers **TextView** SyntaxView ImageView HexView  
WebView Auth Caching Cookies Raw JSON XML

<p>Use https instead of http</p>

```
127.0.0.1:2000 HTTPS localhost:44357 /api/talents application/json; charset=utf-8
```

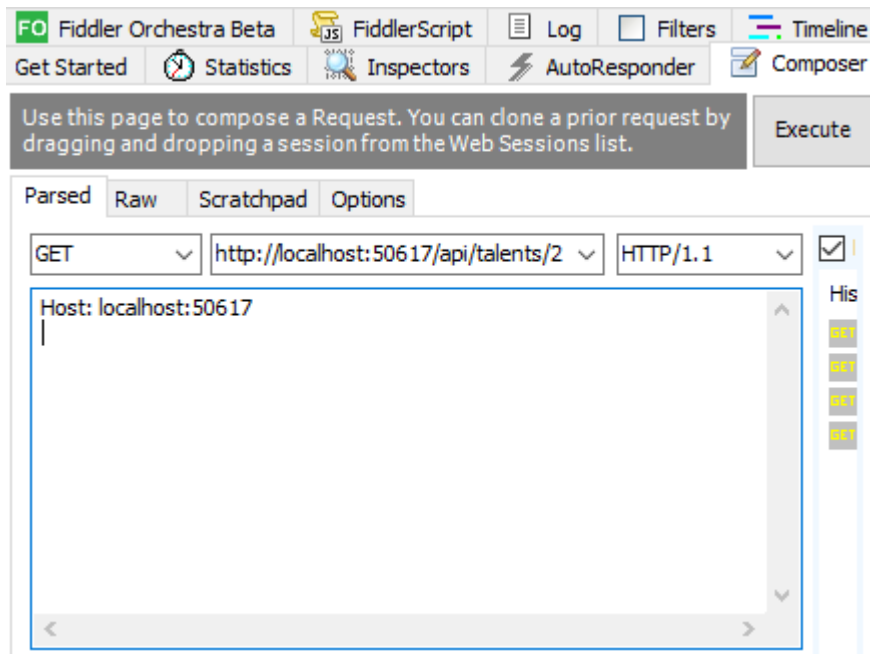




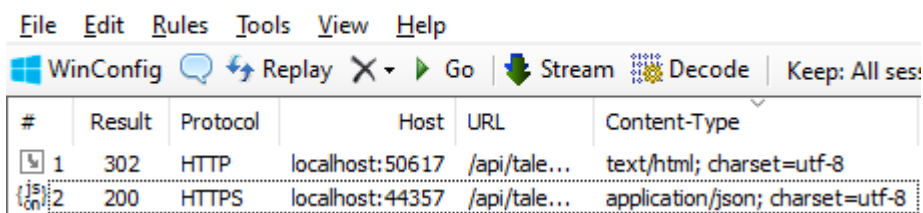
GET: api/talents/{id}

Link: <http://localhost:50617/api/talents/2>

Get talent using http will result a redirect to https. (Available Id between 0 to 3)



Progress Telerik Fiddler Web Debugger



1. Header shows transport to <https://localhost:44357> with response 302 Found.

1 302 HTTP localhost:50617 /api/talents/2 text/html; charset=utf-8

The screenshot displays the Fiddler Orchestra Beta interface. The top toolbar includes buttons for 'Get Started', 'Statistics', 'Inspectors', 'AutoResponder', and 'Composer'. The 'Inspectors' tab is active, showing the 'Request Headers' section with a single header: 'GET /api/talents/2 HTTP/1.1'. Below this, the 'Transport' section shows 'Host: localhost:50617'. The bottom toolbar includes buttons for 'Transformer', 'Headers', 'TextView', 'SyntaxView', 'ImageView', 'HexView', 'WebView', 'Auth', 'Caching', 'Cookies', 'Raw', 'JSON', and 'XML'. The 'TextView' tab is selected, showing the 'Response Headers' section with a single header: 'HTTP/1.1 302 Found'. Below this, the 'Cache' section shows 'Cache-Control: no-cache', 'Date: Sat, 04 Jan 2020 13:42:09 GMT', 'Expires: -1', and 'Pragma: no-cache'. The 'Entity' section shows 'Content-Length: 32' and 'Content-Type: text/html; charset=utf-8'. The 'Miscellaneous' section shows 'Server: Microsoft-IIS/10.0', 'X-AspNet-Version: 4.0.30319', 'X-Powered-By: ASP.NET', and 'X-SourceFiles: =?UTF-8?B?QzpcVXNlcnNcQWRtaW5cRG9jdW1lbnRzXEdpdEh1YlxDU0N'. The 'Transport' section shows 'Location: https://localhost:44357/api/talents/2'. The bottom toolbar is also visible, with 'TextView' selected, showing the response body: '<p>Use https instead of http</p>'. The response body is displayed in a light blue box.

**Request Headers** [Raw] [Header Definitions]  
GET /api/talents/2 HTTP/1.1

**Transport**  
Host: localhost:50617

**Response Headers** [Raw] [Header Definitions]  
HTTP/1.1 302 Found

**Cache**  
Cache-Control: no-cache  
Date: Sat, 04 Jan 2020 13:42:09 GMT  
Expires: -1  
Pragma: no-cache

**Entity**  
Content-Length: 32  
Content-Type: text/html; charset=utf-8

**Miscellaneous**  
Server: Microsoft-IIS/10.0  
X-AspNet-Version: 4.0.30319  
X-Powered-By: ASP.NET  
X-SourceFiles: =?UTF-8?B?QzpcVXNlcnNcQWRtaW5cRG9jdW1lbnRzXEdpdEh1YlxDU0N

**Transport**  
Location: https://localhost:44357/api/talents/2

**TextView**  
<p>Use https instead of http</p>

2. JSON can be view using <https://localhost:44357> with response 200 OK.

2 200 HTTPS localhost:44357 /api/talents/2 application/json; charset=utf-8

The screenshot displays the Fiddler Orchestra Beta interface. The top toolbar includes buttons for 'Get Started', 'Statistics', 'Inspectors', 'AutoResponder', and 'Composer'. Below the toolbar, the 'Inspectors' tab is active, showing the 'Request Headers' section with a 'GET /api/talents/2 HTTP/1.1' request. The 'Transport' section indicates the host is 'localhost:44357'. The 'Response Headers' section shows an 'HTTP/1.1 200 OK' response. The 'Cache' section lists headers: 'Cache-Control: no-cache', 'Date: Sat, 04 Jan 2020 13:42:09 GMT', 'Expires: -1', and 'Pragma: no-cache'. The 'Entity' section shows 'Content-Length: 492' and 'Content-Type: application/json; charset=utf-8'. The 'Miscellaneous' section lists server information: 'Server: Microsoft-IIS/10.0', 'X-AspNet-Version: 4.0.30319', 'X-Powered-By: ASP.NET', and 'X-SourceFiles: =?UTF-8?B?QzpcVXNlcnNcQWRtaW5cRG9jdW1lbnRzXEdpdEh1YlxDU0N'. The 'JSON' tab is selected, showing the response body as a JSON object: { 'Bio': 'Hillary is a sophomore art sculpture student at New York University, and has', 'Id': 2, 'Name': 'Hillary Hewitt Goldwynn-Post', 'Reknown': 'New York University', 'ShortName': 'Hillary\_Goldwynn' }.

**Request Headers** [Raw] [Header Definitions]  
GET /api/talents/2 HTTP/1.1

**Transport**  
Host: localhost:44357

**Response Headers** [Raw] [Header Definitions]  
HTTP/1.1 200 OK

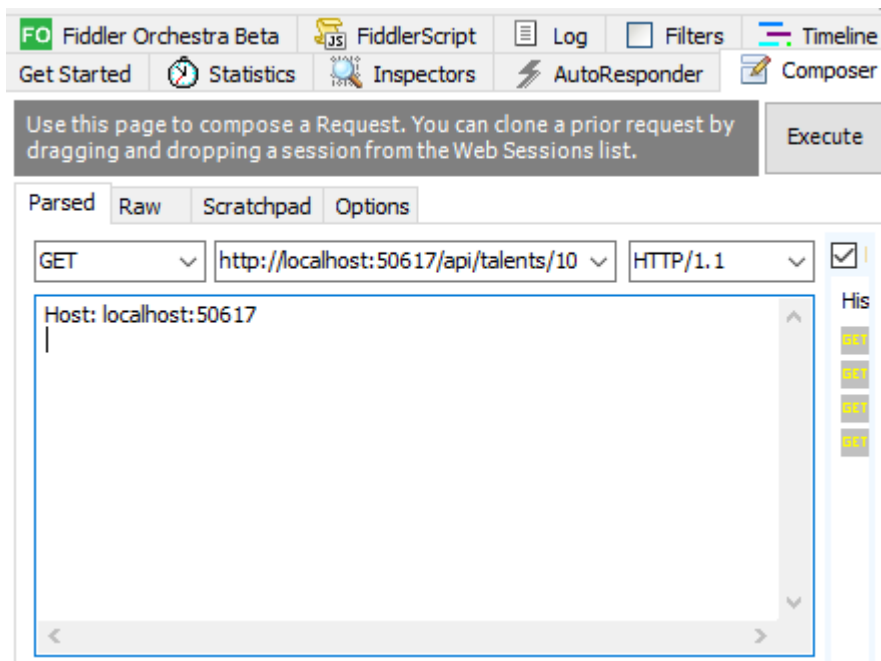
**Cache**  
Cache-Control: no-cache  
Date: Sat, 04 Jan 2020 13:42:09 GMT  
Expires: -1  
Pragma: no-cache

**Entity**  
Content-Length: 492  
Content-Type: application/json; charset=utf-8

**Miscellaneous**  
Server: Microsoft-IIS/10.0  
X-AspNet-Version: 4.0.30319  
X-Powered-By: ASP.NET  
X-SourceFiles: =?UTF-8?B?QzpcVXNlcnNcQWRtaW5cRG9jdW1lbnRzXEdpdEh1YlxDU0N

**JSON**  
{  
 "Bio": "Hillary is a sophomore art sculpture student at New York University, and has",  
 "Id": 2,  
 "Name": "Hillary Hewitt Goldwynn-Post",  
 "Reknown": "New York University",  
 "ShortName": "Hillary\_Goldwynn"  
}

Entering invalid Talent id will still redirect to https but with 400 Bad Request.



Progress Telerik Fiddler Web Debugger

File Edit Rules Tools View Help					
WinConfig Win Replay Go Stream Decode Keep: All s					
#	Result	Protocol	Host	URL	Content-Type
1	302	HTTP	localhost:50617	/api/talents/10	text/html; charset=utf-8
2	400	HTTPS	localhost:44357	/api/talents/10	text/plain; charset=utf-8

## Task 6

### Browser

Insert your publishable key and secret key inside StripeChargeModel.cs from <https://dashboard.stripe.com/test/dashboard>.

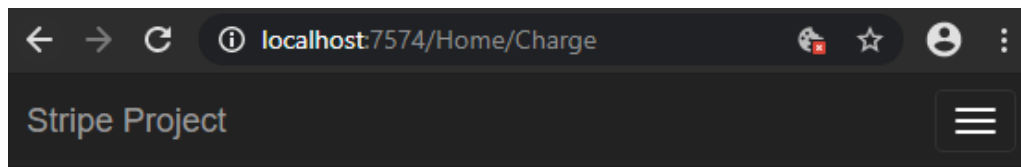
#### ▼ Get your test API keys

Publishable key    pk\_test\_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Secret key         sk\_test\_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```
public class StripeChargeModel
{
    //API Testing Keys
    //PKey = publishable key, SKey = secret key
    1 reference | 0 changes | 0 authors, 0 changes
    public string PKey { get; } = "pk_test_XXX";
    0 references | 0 changes | 0 authors, 0 changes
    public static string SKey { get; } = "sk_test_XXX";
}
```

Debug run the application and you will see the following page on load.



# Stripe Charge Example with Stripe.js

## Stripe Charge Example

**Card Number \***

**Cvc \***

**Expiry Month (MM) \***

**Expiry Year (YYYY) \***

**CardHolderName**

**Amount**

**AddressLine1**

**AddressLine2**

**AddressCity**

**AddressPostcode**

**AddressCountry**

[ProcessPayment](#)

Fill in the details

(Card Holder Name and Addresses are optional)

(Minimum amount: 0.01)

# Stripe Charge Example with Stripe.js

## Stripe Charge Example

---

**Card Number \***

**Cvc \***

**Expiry Month (MM) \***

**Expiry Year (YYYY) \***

**CardHolderName**

**Amount**

**AddressLine1**

**AddressLine2**

**AddressCity**

**AddressPostcode**

**AddressCountry**

[ProcessPayment](#)

Stripe will start validating card, create token and charge.  
(indicated by the loading icon)

# Stripe Charge Example with Stripe.js

## Stripe Charge Example

---

**Card Number \***

**Cvc \***

**Expiry Month (MM) \***

**Expiry Year (YYYY) \***

**CardHolderName**

**Amount**

**AddressLine1**

**AddressLine2**

**AddressCity**

**AddressPostcode**

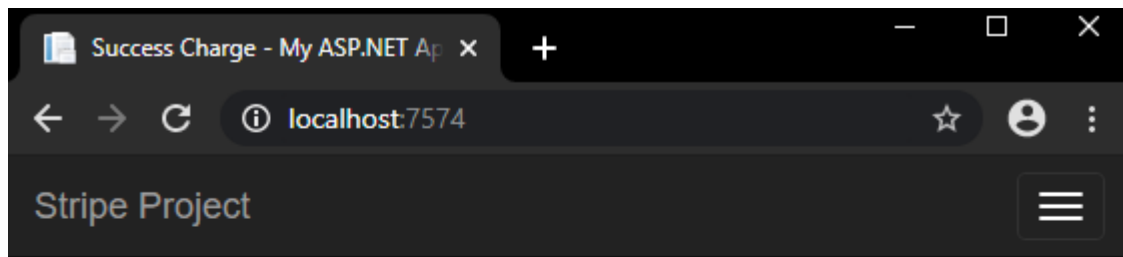
**AddressCountry**

[ProcessPayment](#)





Successful charge will result in successful page with redirect.



# Charge Successful! Redirecting back to Stripe Charge Example page...



Successful charge will be reflected in <https://dashboard.stripe.com/test/payments> .

Payment

TEST DATA

ch\_1FxJQkItmXAp9SBazkYD6frI

\$100.00 SGD

Succeeded ✓

Refund...

Date	Customer	Payment method	Risk evaluation
Jan 5, 4:41 AM	None	**** 5100	14 Normal

Timeline

+ Add note

✓ Payment succeeded

Jan 5, 2020, 4:41 AM

14 Stripe risk evaluation: normal

Jan 5, 2020, 4:41 AM

Payment details

Statement descriptor	Stripe
Amount	\$100.00
Fee	\$3.90 ⓘ
Net	\$96.10
Status	Succeeded
Description	Description for test charge <a href="#">Edit</a>

Payment method

Number	**** 5100	Address	No address
Fingerprint	oqYeuygtSUazJwMn	Origin	United States 🇺🇸
Expires	02 / 2020	CVC check	Passed ✓
Type	Mastercard prepaid card		
ID	card_1FxJQjItmXAp9SBaFmTFWtct		

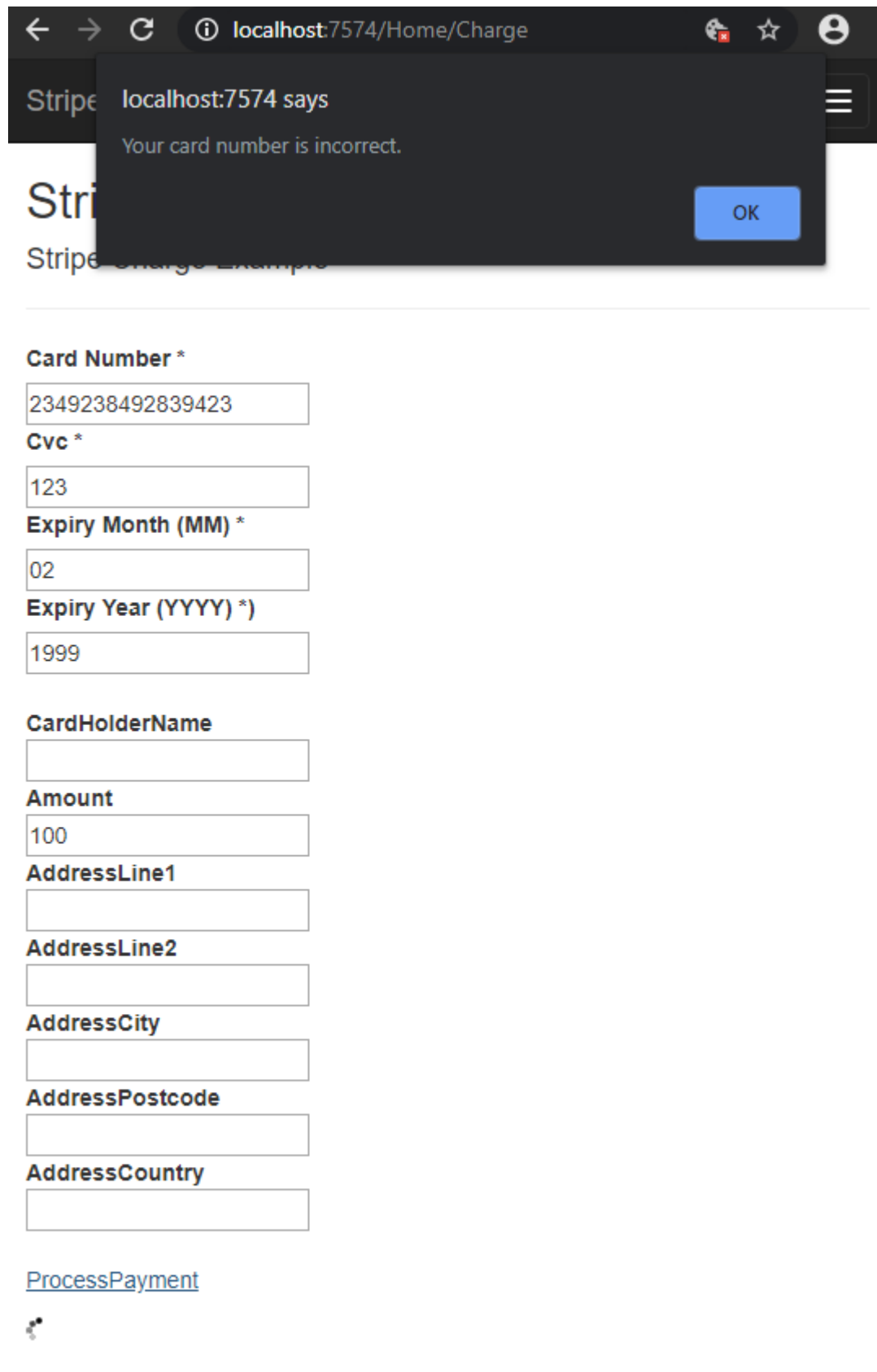
Logs

200 OK	POST /v1/charges	1/5/20, 4:41:06 AM
200 OK	POST /v1/tokens	1/5/20, 4:41:05 AM

Events

A successful payment was made for \$100.00 SGD	1/4/20, 8:41:06 PM
--	--------------------

Error from Stripe if card details are invalid .



The screenshot shows a web browser window with the address bar displaying "localhost:7574/Home/Charge". A dark modal dialog box is overlaid on the page, containing the text "localhost:7574 says" and "Your card number is incorrect." with an "OK" button. Below the dialog, the Stripe payment form is visible. The form includes fields for Card Number (2349238492839423), Cvc (123), Expiry Month (MM) (02), Expiry Year (YYYY) (1999), CardHolderName, Amount (100), AddressLine1, AddressLine2, AddressCity, AddressPostcode, and AddressCountry. A "ProcessPayment" button is at the bottom of the form.

localhost:7574 says

Your card number is incorrect.

OK

**Card Number \***

2349238492839423

**Cvc \***

123

**Expiry Month (MM) \***

02

**Expiry Year (YYYY) \***

1999

**CardHolderName**

**Amount**

100

**AddressLine1**

**AddressLine2**

**AddressCity**

**AddressPostcode**

**AddressCountry**

[ProcessPayment](#)

Error from Model Validation if amount is invalid.

**CardHolderName**

**Amount**

The field Amount must be between 0.01 and ∞.

**AddressLine1**

**AddressLine2**

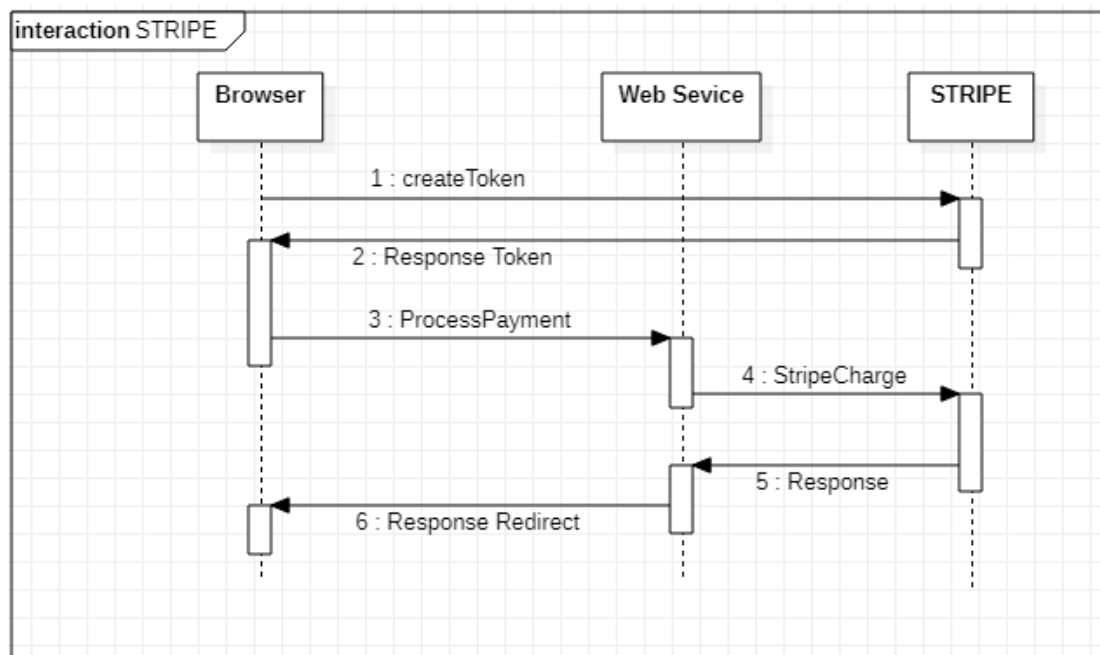
**AddressCity**

**AddressPostcode**

**AddressCountry**

[ProcessPayment](#)

## Sequence Diagram



## AWS Talents Images

Access: <https://s3-ap-southeast-1.amazonaws.com/talentsweihan>

Images uploaded into AWS S3 Bucket.

aws

Services ▾

Resource Groups ▾

★

🔔

Wong Wei Han ▾

Global ▾

Amazon S3 > talentsweihan

talentsweihan

Overview

Properties

PermissionsPublic

Management

Access points

🔍 Type a prefix and press Enter to search. Press ESC to clear.

📁 Upload










➕ Create folder

⬇️ Download

⚙️ Actions ▾

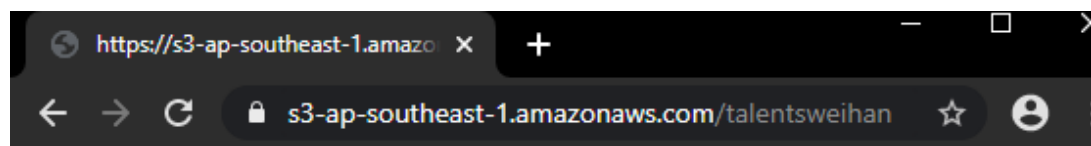
Asia Pacific (Singapore) 🔄

Viewing 1 to 9

<input type="checkbox"/> Name ▾	Last modified ▾	Size ▾	Storage class ▾
<input type="checkbox"/>  Barot_Bellingham_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	12.5 KB	Standard
<input type="checkbox"/>  Constance_Smith_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	12.5 KB	Standard
<input type="checkbox"/>  Hassum_Harrod_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	11.1 KB	Standard
<input type="checkbox"/>  Hillary_Goldwynn_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	10.8 KB	Standard
<input type="checkbox"/>  Jennifer_Jerome_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	15.0 KB	Standard
<input type="checkbox"/>  Jonathan_Ferrar_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	13.0 KB	Standard
<input type="checkbox"/>  LaVonne_LaRue_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	12.2 KB	Standard
<input type="checkbox"/>  Riley_Rewington_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	11.8 KB	Standard
<input type="checkbox"/>  Xhou_Ta_tn.jpg	Jan 7, 2020 4:44:48 AM GMT+0800	12.8 KB	Standard

Viewing 1 to 9

Public access to S3 Bucket.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>talentsweihaan</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>Barot_Bellingham_tn.jpg</Key>
    <LastModified>2020-01-06T20:44:48.000Z</LastModified>
    <ETag>"2d7605292499f1cf635b2dd7c28982fb"</ETag>
    <Size>12755</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>Constance_Smith_tn.jpg</Key>
    <LastModified>2020-01-06T20:44:48.000Z</LastModified>
    <ETag>"58f2cb1520ceb1bc71107e87ef3a14ab"</ETag>
    <Size>12795</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>Hassum_Harrod_tn.jpg</Key>
    <LastModified>2020-01-06T20:44:48.000Z</LastModified>
    <ETag>"817bd23c23477d4076534ebf956ef4bc"</ETag>
    <Size>11356</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>Hillary_Goldwynn_tn.jpg</Key>
    <LastModified>2020-01-06T20:44:48.000Z</LastModified>
    <ETag>"4de2340b0d4649bc324286b945e13ac9"</ETag>
    <Size>11051</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>Jennifer_Jerome_tn.jpg</Key>
    <LastModified>2020-01-06T20:44:48.000Z</LastModified>
    <ETag>"e1aa96736e3c759d3a9efd519d1fbf3b"</ETag>
    <Size>15361</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

## Task 7

### Browser (ASP.NET)

Insert your Clarifai API key inside ClarifaiModel.cs from <https://portal.clarifai.com/apps> .

#### API Keys ?

[DETAILS](#)

##### ClarifaiProject-all-scopes

XX

Dec 12, 2019

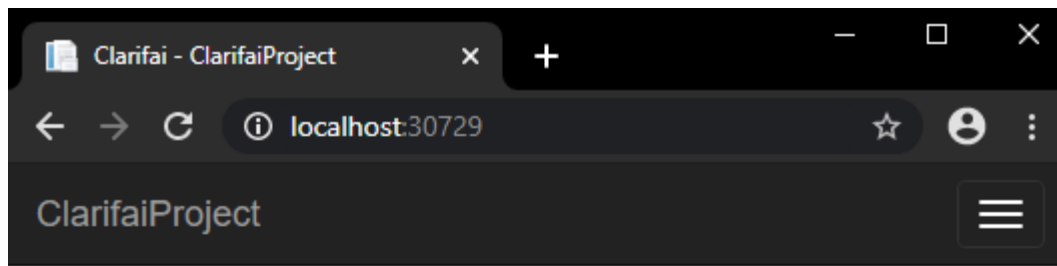


[CREATE NEW API KEY](#)

```
public class ClarifaiModel
{
    //API key for clarifai project
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public static string APIKey { get; } = "XXX";
}
```



Debug run the application and you will see the following page on load.



## Submit Web image

Submit

## Submit Local image

Drag and Drop your Image

Choose File No file chosen

Upload

## Image preview

Submit: Web Image

Submit image using web image and wait for tag to load.

## Submit Web image

Submit

## Submit Local image

Drag and Drop your Image

Choose File No file chosen

Upload

## Image preview



Tag

It will take a while for the tags to load.



Tag

Confidence

Image tag will be generated after a few seconds.

## Image preview

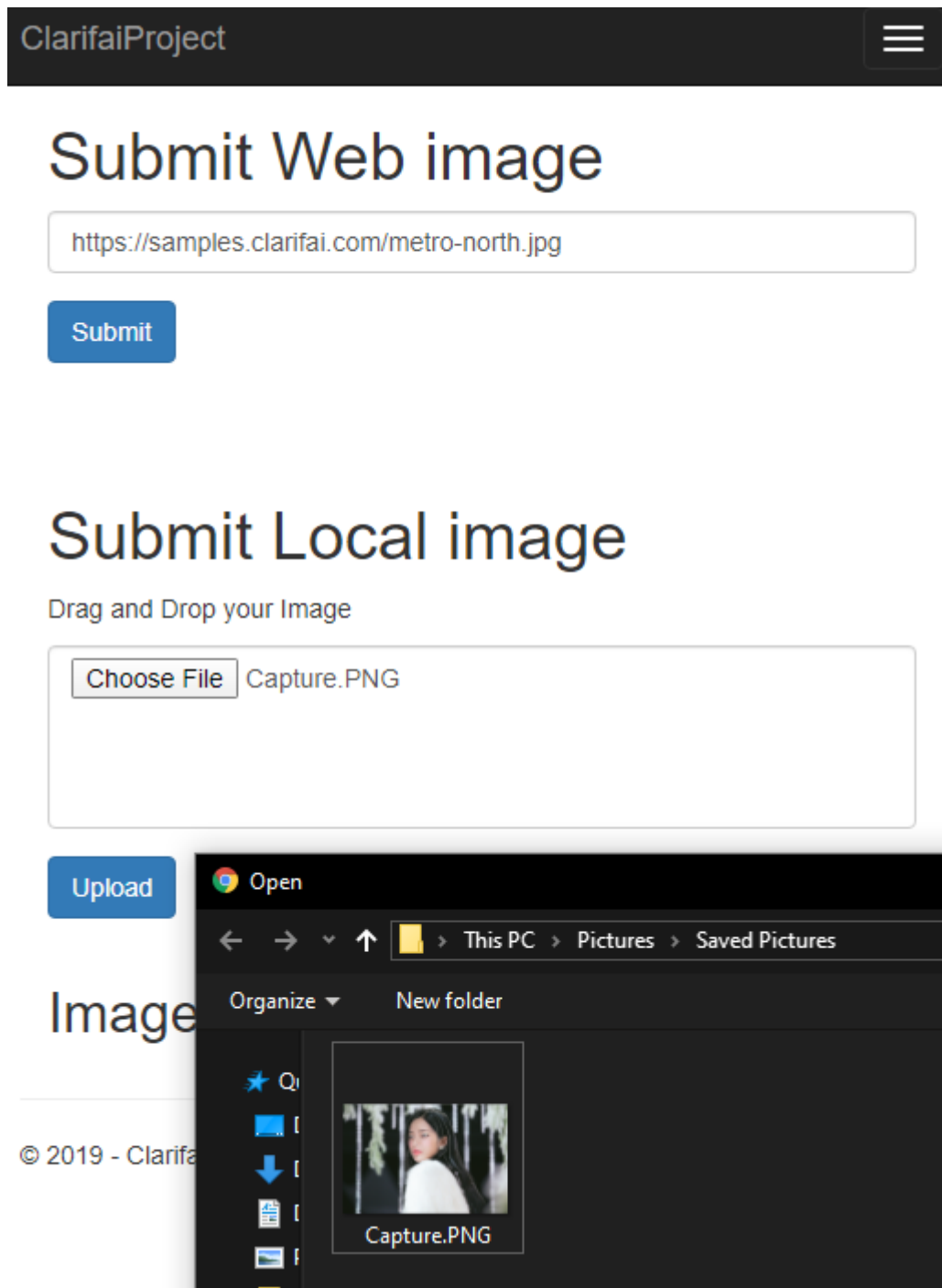


### Tag

Tag	Confidence
train	0.9996053
railway	0.9992987
subway system	0.99825156
station	0.9980105
locomotive	0.99725723
transportation system	0.9969802
travel	0.9889797
commuter	0.98087525
platform	0.98064405
light	0.97420406
train station	0.96874046
blur	0.9672204
city	0.9614798
road	0.9613829

Submit: Local Image

Image can be uploaded from local computer via drag and drop.



Uploaded image will undergo tagging.

ClarifaiProject



## Submit Web image

<https://samples.clarifai.com/metro-north.jpg>

Submit

## Submit Local image

Drag and Drop your Image

Choose File Capture.PNG

Upload

## Image preview



Tag

It will take a while for the tags to load.



Tag

Confidence

Image tag will be generated after a few seconds.

## Image preview



### Tag

Tag	Confidence
fashion	0.99193025
girl	0.9905751
winter	0.98973143
model	0.9879178
woman	0.98600954
portrait	0.9820337
beautiful	0.97932243
snow	0.97022843
cold	0.9481993

In the case of error, server will reconnect and prompt to check.

## Tag

Web Server has failed to establish a connection to Clarifai. Reconnecting...  
(Check your network and API key in ClarifaiModel)



Tag	Confidence

## Browser (javascript)

*Located in ClarifaiProject\_js Folder*

Insert your Clarifai API key inside keys.js from <https://portal.clarifai.com/apps> .

### API Keys

[DETAILS](#)


#### ClarifaiProject-all-scopes

XX

Dec 12, 2019



[CREATE NEW API KEY](#)

```
JS keys.js >  myApiKey  
1 var myApiKey = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';
```



This Javascript version is the same version as ASP.NET, without a web server.



# ClarifaiProject

## Submit Web image

<https://samples.clarifai.com/metro-north.jpg>

Submit

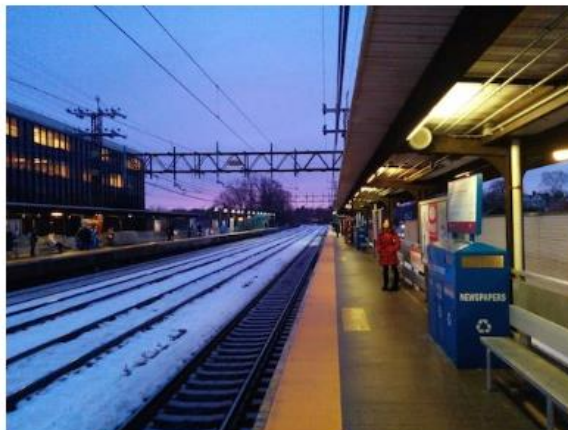
## Submit Local image

Drag and Drop your Image

Browse... Capture.PNG

Upload

## Image preview



## Tag

Tag	Confidence
train	0.9996053
railway	0.9992987
subway system	0.99825156
station	0.9980105
locomotive	0.99725723
transportation system	0.9969802