

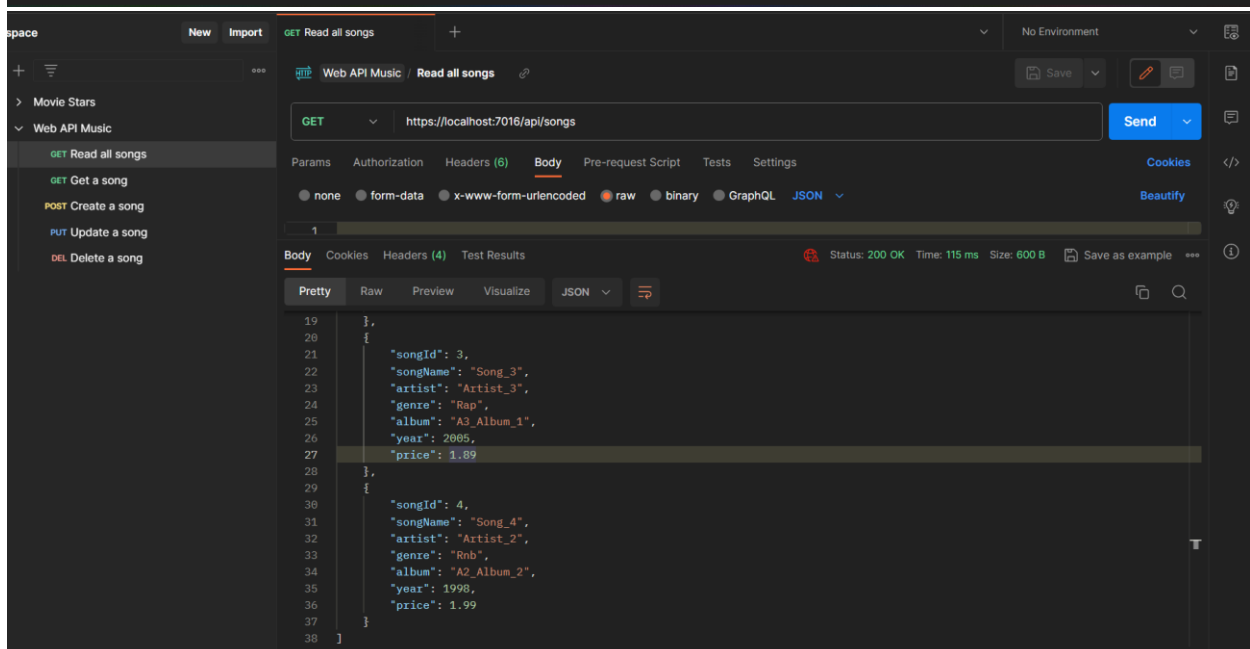
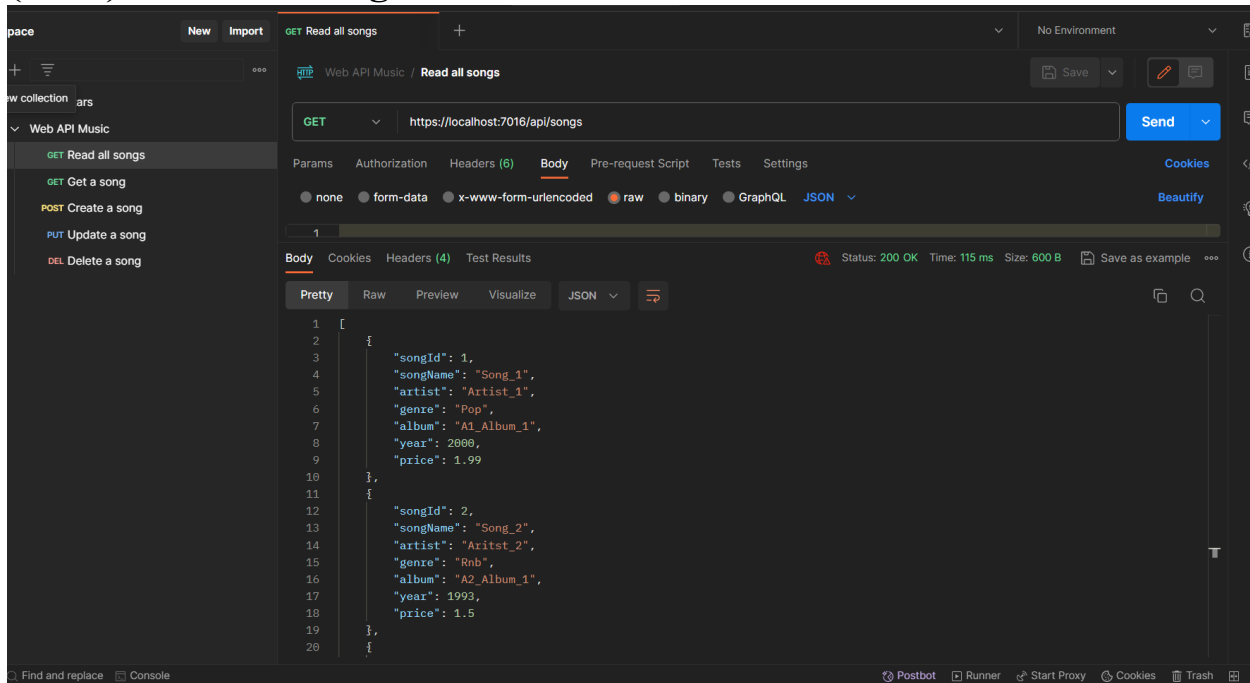
Assignment #1 - Final Project- ASP.NET API

Shanelle Haye

January 10, 2024

Music API

(GET) Read All Songs :



(GET) Get a Song:

The screenshot shows the Postman interface for a GET request to `https://localhost:7016/api/songs/1`. The request is successful, returning a 200 OK status with a response time of 18 ms and a size of 260 B. The response body is displayed in JSON format, showing details for a song with ID 1.

Request:

- Method: GET
- URL: `https://localhost:7016/api/songs/1`

Response Body (JSON):

```
{  "songId": 1,  "songName": "Song_1",  "artist": "Artist_1",  "genre": "Pop",  "album": "A1_Album_1",  "year": 2000,  "price": 1.99}
```

The screenshot shows the Postman interface for a GET request to `https://localhost:7016/api/songs/-1`. The request fails with a 400 Bad Request status, indicating a validation error. The response body is displayed in JSON format, showing the error details.

Request:

- Method: GET
- URL: `https://localhost:7016/api/songs/-1`

Response Body (JSON):

```
{  "title": "One or more validation errors occurred.",  "status": 400,  "errors": {    "SongId": [      "SongId is invalid"    ]  }}
```

GET Read all songs GET Get a song + No Environment

Web API Music / Get a song Save Edit Comments

GET https://localhost:7016/api/songs/9 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 404 Not Found Time: 5 ms Size: 271 B Save as example

Pretty Raw Preview Visualize JSON Beautify

```
1 {
2   "title": "One or more validation errors occurred.",
3   "status": 404,
4   "errors": {
5     "SongId": [
6       "Song does not exist"
7     ]
8   }
9 }
```

(POST) Create a Song:

GET Read all songs GET Get a song POST Create a song + No Environment

Web API Music / Create a song Save Edit Comments

POST https://localhost:7016/api/songs Send

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

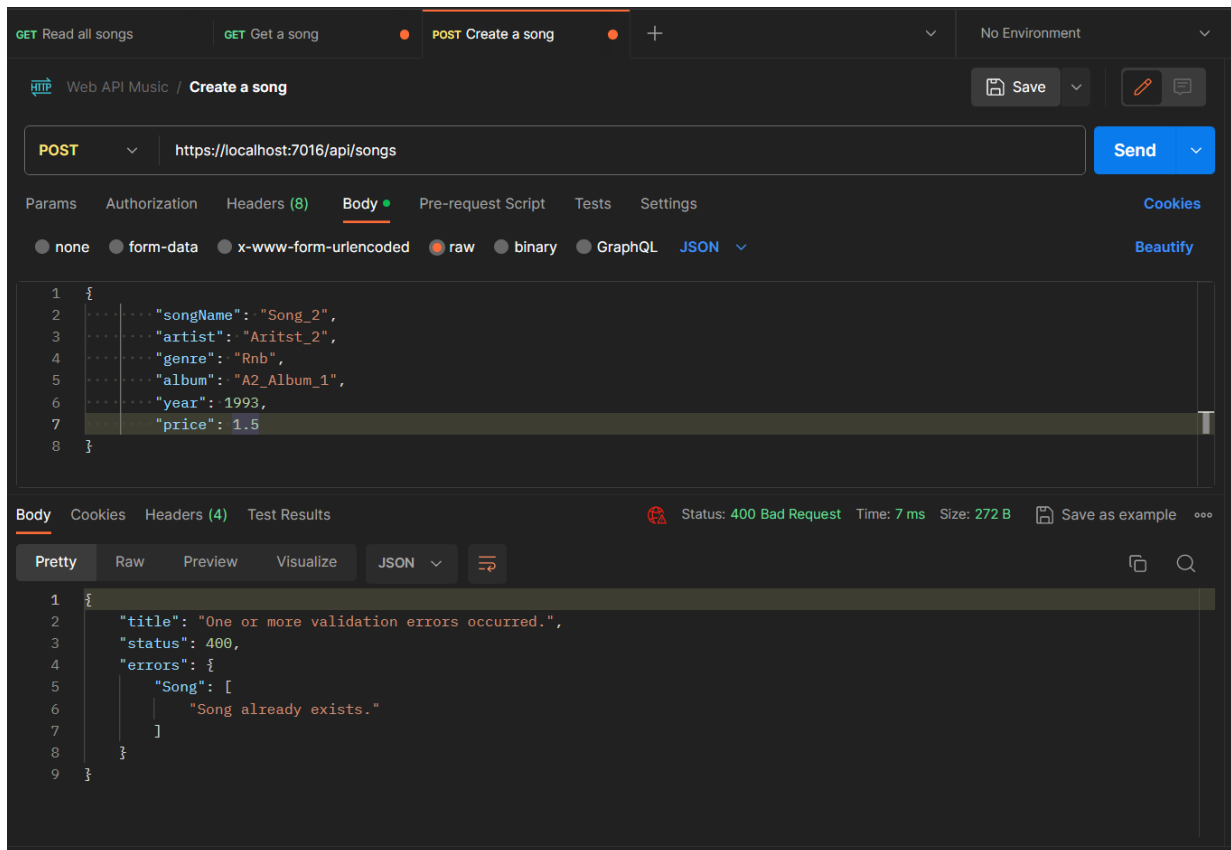
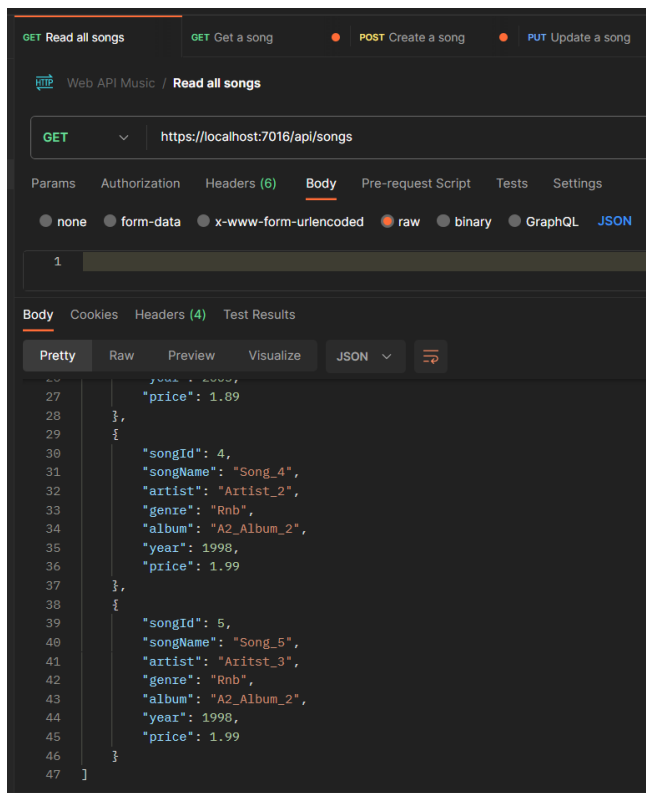
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "songName": "Song_5",
3   "artist": "Artist_3",
4   "genre": "Rnb",
5   "album": "A2_Album_2",
6   "year": 1998,
7   "price": 1.99
8 }
```

Body Cookies Headers (5) Test Results Status: 201 Created Time: 44 ms Size: 311 B Save as example

Pretty Raw Preview Visualize JSON Beautify

```
1 {
2   "songId": 5,
3   "songName": "Song_5",
4   "artist": "Artist_3",
5   "genre": "Rnb",
6   "album": "A2_Album_2",
7   "year": 1998,
8   "price": 1.99
9 }
```



(PUT) Update a Song:

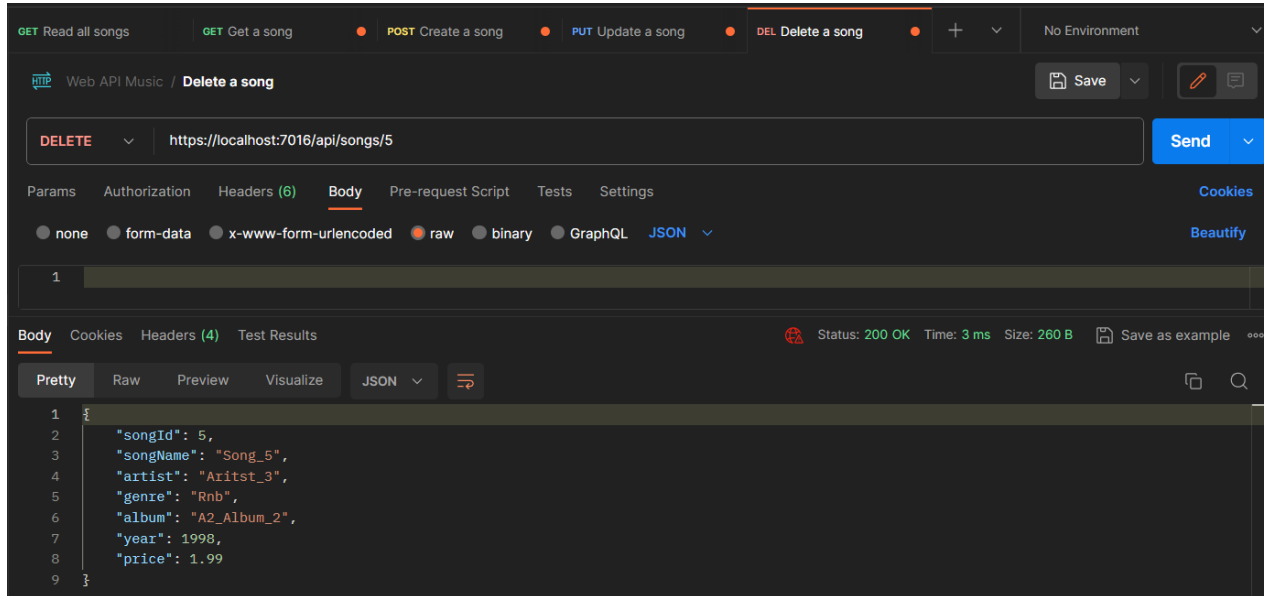
The screenshot shows a REST client interface with a dark theme. At the top, there are tabs for different HTTP methods: GET Read all songs, GET Get a song, POST Create a song, and PUT Update a song (which is selected). The URL bar shows 'https://localhost:7016/api/songs/2'. The 'Body' tab is active, displaying a JSON object with song details. The interface includes a 'Send' button and various settings like 'Headers (8)', 'Pre-request Script', 'Tests', and 'Settings'.

```
1 {
2   "songId": 2,
3   "songName": "Song_2",
4   "artist": "Aristst_2",
5   "genre": "Rnb",
6   "album": "A2_Album_1",
7   "year": 1993,
8   "price": 3.99
9 }
```

The screenshot shows a REST client interface with a dark theme. At the top, there are tabs for different HTTP methods: GET Read all songs, GET Get a song (which is selected), and POST Create a song. The URL bar shows 'https://localhost:7016/api/songs/2'. The 'Body' tab is active, displaying a JSON object with song details. The interface includes a 'Send' button and various settings like 'Headers (6)', 'Pre-request Script', 'Tests', and 'Settings'.

```
1 {
2   "songId": 2,
3   "songName": "Song_2",
4   "artist": "Aristst_2",
5   "genre": "Rnb",
6   "album": "A2_Album_1",
7   "year": 1993,
8   "price": 3.99
9 }
```

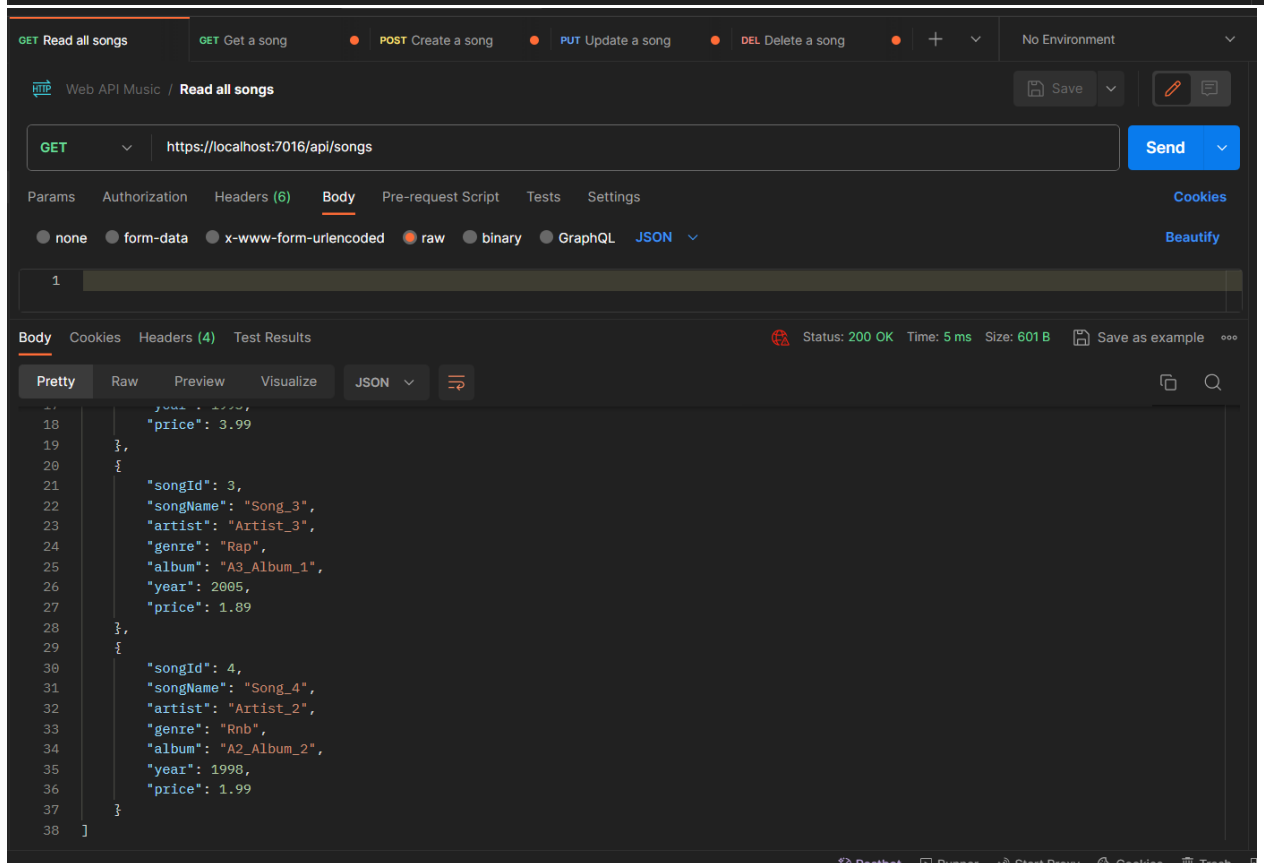
(DEL) Delete a Song:



The screenshot shows the Postman interface for a DELETE request. The top bar includes tabs for various HTTP methods: GET Read all songs, GET Get a song, POST Create a song, PUT Update a song, and DEL Delete a song (which is currently selected). The URL bar shows the endpoint `https://localhost:7016/api/songs/5`. The 'Body' tab is active, displaying a JSON object with the following details:

```
1 {
2   "songId": 5,
3   "songName": "Song_5",
4   "artist": "Aristst_3",
5   "genre": "Rnb",
6   "album": "A2_Album_2",
7   "year": 1998,
8   "price": 1.99
9 }
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 3 ms, Size: 260 B.



The screenshot shows the Postman interface for a GET request. The top bar includes tabs for various HTTP methods: GET Read all songs, GET Get a song, POST Create a song, PUT Update a song, and DEL Delete a song. The URL bar shows the endpoint `https://localhost:7016/api/songs`. The 'Body' tab is active, displaying a JSON array of song objects:

```
17 [
18   {
19     "songId": 2,
20     "songName": "Song_2",
21     "artist": "Artist_1",
22     "genre": "Pop",
23     "album": "A1_Album_1",
24     "year": 2000,
25     "price": 2.99
26   },
27   {
28     "songId": 3,
29     "songName": "Song_3",
30     "artist": "Artist_3",
31     "genre": "Rap",
32     "album": "A3_Album_1",
33     "year": 2005,
34     "price": 1.89
35   },
36   {
37     "songId": 4,
38     "songName": "Song_4",
39     "artist": "Artist_2",
40     "genre": "Rnb",
41     "album": "A2_Album_2",
42     "year": 1998,
43     "price": 1.99
44   }
45 ]
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 5 ms, Size: 601 B.