

CMPE 275 Lab 2

REST and Persistence

Fall 2020

Group 7

Base URL: <http://ec2-18-205-192-46.compute-1.amazonaws.com>

Group Member Details

Apoorv Mehrotra	apoovr.mehrotra@sjsu.edu	014597635
Pranjay Sagar	pranjay.sagar@sjsu.edu	014611922
Shweta Mane	shwetaashokrao.mane@sjsu.edu	014604239
Sourabh Garg	sourabh.garg@sjsu.edu	013727103

Contents

1. Create a player.....	3
2. Get a player.....	8
3. Update a player.....	11
4. Delete a player.....	14
5. Create a sponsor.....	18
6. Get a sponsor.....	19
7. Update a sponsor.....	21
8. Delete a sponsor.....	23
9. Add an opponent.....	25
10. Remove an opponent.....	30

Base URL: <http://ec2-18-205-192-46.compute-1.amazonaws.com>

Test Results

1. Create a player

Scenario 1: Create a player without a sponsor (200 Expected, Creation should pass)

http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Spencer&lastname=Shay&email=spenser@gmail.com&description=Sculptor

The screenshot shows the Postman application interface. The top bar includes tabs for 'Builder' and 'Team Library', and a status indicator 'IN SYNC'. Below the header, there's a search bar and a 'Collections' tab. The main workspace shows a 'POST' request to the URL 'http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Spencer&lastname=Shay&email=spenser@gmail.com&description=Sculptor'. The 'Body' tab displays the JSON response received from the server:

```
1  {
2   "id": 111,
3   "firstname": "Spencer",
4   "lastname": "Shay",
5   "email": "spenser@gmail.com",
6   "description": "Sculptor"
7 }
```

The status bar at the bottom indicates 'Status: 200 OK' and 'Time: 1409 ms'. The taskbar at the bottom of the screen shows various open applications.

Entry inserted in database

The screenshot shows the MySQL Workbench interface. The top menu includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The left sidebar shows the 'Navigator' with 'SCHEMAS' expanded to show 'CMPE275_LAB2' containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The central area has a 'Query 1' tab with the query 'SELECT * FROM CMPE275_LAB2.PLAYER;'. The results grid shows the following data:

	id	firstname	lastname	email	description	sponsorid
96	charles	zhang	charles123@gmail.com	President	NULL	
106	check	name	check.name@gmail.com	new Description	98	
108	NewPlayer	New	new@new.com	Description of new player	NULL	
109	Carly	Shay	carly@gmail.com	web presenter	98	
110	Freddie	Benson	freddie@gmail.com	Tech Producer	NULL	
111	Spencer	Shay	spenser@gmail.com	Sculptor	NULL	

The bottom section shows the 'Action Output' log with three entries:

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec

Scenario 2: Create a player with a sponsor (200 expected , creation should pass)

Using sponsorid 5

MySQL Workbench - GlassDoor New - cmpe275-lab2

File Edit View Query Database Server Tools Scripting Help

Navigator SCHEMAS Filter objects CMPE275_LAB2 Tables Views Stored Procedures Functions

Query 1 SPONSOR PLAYER SQL File 6

1 • SELECT * FROM CMPE275_LAB2.SPONSOR;

Result Grid Filter Rows: Edit Export/Import: Wrap Cell Content: Result Grid Form Editor Context Help Snippets

No object selected

SPONSOR 1

Action Output

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec

Object Info Session

23:09 21/11/2020

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Alex&lastname=Shay&email=alex@gmail.com&description=Actor&sponsorId=5>

Builder Team Library IN SYNC Apoorv Me... 🌐 📡 🚨 🎉

Bring your whole team to Postman for free! Update the app to experience the new features. Download

Create a player Update a player No Environment

POST http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Alex&lastname=Shay&email=alex@gmail.com&description=Actor&sponsorId=5 Params Send Save

Authorization Headers Body Pre-request Script Tests Code

Type: No Auth

Body Cookies Headers (5) Test Results Status: 200 OK Time: 533 ms

```

1 [
2   "id": 112,
3   "firstname": "Alex",
4   "lastname": "Shay",
5   "email": "alex@gmail.com",
6   "description": "Actor",
7   "sponsor": {
8     "id": 5,
9     "name": "pranjay",
10    "description": "sagar",
11    "address": {
12      "street": "1st street",
13      "city": "san jose",
14      "state": "CA",
15      "zip": "95134"
16    }
17  }

```

POST Create a player

GET Get a player

PUT Update a player

DELETE Delete a player

PUT Opponent Creation

POST Create Sponsor

GET Get sponsor

PUT Update sponsor

DELETE Delete sponsor

DELETE Delete opponent

Menu Fetch API 1 request

275 Lab2 10 requests

23:09 21/11/2020

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'cmpe275-lab2'. The main area displays a query results grid for the 'PLAYER' table. The grid shows 11 rows of data, including columns: id, firstname, lastname, email, description, and sponsorid. A specific row for player ID 112 is selected. Below the grid, the 'Action Output' panel shows three log entries corresponding to the executed SQL statements.

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec

Scenario 3: Missing parameters test (400 status code expected)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin>

The screenshot shows the Postman application interface. A POST request is being made to the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin>. The request body contains the following key-value pairs: 'firstname': 'Justin', 'lastname': 'Shay', 'email': 'justin@gmail.com', and 'description': 'Actor'. The 'Authorization' tab is selected, showing 'No Auth'. The response status is 400 Bad Request, indicating a missing parameter error.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin&lastname=Shay>

The screenshot shows the Postman Builder interface. On the left, there's a sidebar with a 'Collections' tab selected, showing a list of API endpoints: 'Create a player' (POST), 'Get a player' (GET), 'Update a player' (PUT), 'Delete a player' (DEL), 'Opponent Creation' (PUT), 'Create Sponsor' (POST), 'Get sponsor' (GET), 'Update sponsor' (PUT), 'Delete sponsor' (DEL), 'Delete opponent' (DEL), and 'Menu Fetch API' (GET). The 'Create a player' endpoint is highlighted. The main panel shows a POST request to 'http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin&lastname=Shay'. The request body contains the following parameters:

Key	Value
firstname	Justin
lastname	Shay
email	justin@gmail.com
description	Actor
sponsorid	5

The 'Authorization' tab is selected, showing 'No Auth'. The response status is '400 Bad Request' with a time of '243 ms'. The bottom status bar shows the date and time as '22:59 21/11/2020'.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin&email=justin@gmail.com>

This screenshot is identical to the one above, except the 'email' parameter has been added to the request body. The 'email' key now has the value 'justin@gmail.com'. The rest of the interface and response details remain the same.

Scenario 4: Duplicate EmailID (400 status code expected)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin1&lastname=Shay&email=justin@gmail.com>

The screenshot shows the Postman application interface. On the left, the sidebar lists various collections and requests. The 'Create a player' request under the 'POST' section is selected. The request URL is set to <http://ec2-18-205-192-46.compute-1.amazonaws.com/player?firstname=Justin1&lastname=Shay&email=justin@gmail.com>. The request body contains the following JSON:

```
1 {  
2   "message": "could not execute statement; SQL [n/a]; constraint [PLAYER.email]; nested exception is org.hibernate.exception  
3     .ConstraintViolationException: could not execute statement"  
4 }
```

The response status is 400 Bad Request, and the time taken was 266 ms. The response body is displayed in the 'Pretty' tab of the Body panel.

2. Get a player

Scenario 1: Fetching a player with no opponents and no sponsor (200 status code and JSON player object expected)

Fetching player with id 111

The screenshot shows the MySQL Workbench interface. In the Query Editor, the following SQL query is run:

```
SELECT * FROM CMPE275_LAB2.PLAYER;
```

The Result Grid displays the following data:

	id	firstname	lastname	email	description	sponsorid
96	charles	zhang		charles123@gmail.com	President	NULL
106	check	name		checkname@gmail.com	new Description	98
108	NewPlayer	New		new@new.com	Description of new player	NULL
109	Carly	Shay		carly@gmail.com	web presenter	98
110	Freddie	Benson		freddie@gmail.com	Tech Producer	NULL
111	Spencer	Shay		spenser@gmail.com	Sculptor	NULL

The Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/111>

The screenshot shows the Postman application interface. A GET request is made to the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/111>. The response body is displayed as follows:

```
1 {
2   "id": 111,
3   "firstname": "Spencer",
4   "lastname": "Shay",
5   "email": "spenser@gmail.com",
6   "description": "Sculptor",
7   "opponents": []
8 }
```

Scenario 2: Fetching a player with sponsor and no opponents (200 status code and sponsor details expected)

Will fetch player with id 112 and sponsor id 5

The screenshot shows the MySQL Workbench interface. In the Query Editor, the following SQL query is run:

```
SELECT * FROM CMPE275_LAB2.PLAYER;
```

The Result Grid displays the following data:

ID	firstname	lastname	email	description	sponsorid
108	NewPlayer	New	new@new.com	Description of new player	HULL
109	Carly	Shay	carly@gmail.com	web presenter	98
110	Freddie	Benson	freddie@gmail.com	Tech Producer	HULL
111	Spencer	Shay	spenser@gmail.com	Sculptor	HULL
112	Alex	Shay	alex@gmail.com	Actor	5

The Output pane shows the results of three actions:

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0,1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0,1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/112>

The screenshot shows the Postman application interface. A GET request is made to the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/112>. The response body is displayed as follows:

```

1  {
2   "id": 112,
3   "firstname": "Alex",
4   "lastname": "Shay",
5   "email": "alex@gmail.com",
6   "description": "Actor",
7   "sponsor": {
8     "id": 5,
9     "name": "pranjay",
10    "description": "sagar",
11    "address": {
12      "street": "1st street",
13      "city": "san jose",
14      "state": "CA",
15      "zip": "95134"
16    },
17    "opponents": []
18  }
19 }
```

Scenario 3: Fetching a player with sponsor and opponents (200 status code and sponsor details expected)

Will fetch player with id 84 and sponsor id 100 and whose opponent is with payer id 85

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the 'PLAYER' tab is selected. A query window contains the SQL command:

```
SELECT * FROM CMPE275_LAB2.PLAYER;
```

The result grid displays the following data:

	id	firstname	lastname	email	description	sponsorid
1	84	jenny	k	b@gmail.com	HULL	100
2	85	jenny	k	c@gmail.com	HULL	100
3	88	joe	biden	mr.biden@gmail.com	HULL	HULL
4	95	Kevin2	Jonas	kevin12@gmail.com	HULL	HULL
5	96	charles	zhang	charles123@gmail.com	President	HULL
6	106	check	name	check@name.com	new Description	98

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	22:12:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	10 row(s) returned	0.062 sec / 0.000 sec
2	22:45:17	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
3	22:46:51	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.094 sec / 0.000 sec
4	23:25:34	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	2 row(s) returned	0.078 sec / 0.000 sec

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84>

The screenshot shows the Postman application interface. The left sidebar lists various API endpoints. The main panel shows a GET request to the URL:

GET http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84

The response body is displayed as JSON:

```

1 {
2   "id": 84,
3   "firstname": "jenny",
4   "lastname": "k",
5   "email": "b@gmail.com",
6   "sponsor": {
7     "id": 100,
8     "name": "Dan Schneider",
9     "address": {
10       "street": "1st street",
11       "city": "san jose",
12       "state": "CA",
13       "zip": "95110"
14     }
15   },
16   "opponents": [
17     {
18       "id": 85,
19       "firstname": "jenny",
20       "lastname": "k",
21       "email": "c@gmail.com"
22     }
23   ]
24 }

```

Scenario 4: Fetching a player with invalid/non-existent Id (404 status code expected)

The screenshot shows the Postman application interface. In the left sidebar, under the 'Collections' tab, there is a list of API endpoints. One endpoint, 'Get a player' (GET), is highlighted. The main workspace shows a GET request to the URL `http://ec2-18-205-192-46.compute-1.amazonaws.com/player/2014`. The response status is 404 Not Found, and the response body is a JSON object with the message `"message": "Player Not Exists!"`.

3. Update a player

Scenario 1: Updating all the attributes of player (200 status code expected, updated JSON object returned)

Fetching player with id 84

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84>

The screenshot shows the Postman application interface. In the left sidebar, under the 'Collections' tab, there is a list of API endpoints. One endpoint, 'Get a player' (GET), is highlighted. The main workspace shows a GET request to the URL `http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84`. The response status is 200 OK, and the response body is a detailed JSON object representing a player with attributes like id, firstname, lastname, email, sponsor, address, city, state, zip, and opponents.

Updating player with id 84

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&lastname=Mehrotra&email=apoovr12@gmail.com&description=Student&sponsordid=98>

The screenshot shows the Postman Builder interface. The URL in the header is <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&lastname=Mehrotra&email=apoovr12@gmail.com&description=Student&sponsordid=98>. The request method is PUT. The request body is a JSON object:

```
1 {
2   "id": 84,
3   "firstname": "Abhijeet",
4   "lastname": "Mehrotra",
5   "email": "apoovr12@gmail.com",
6   "description": "student",
7   "sponsor": {
8     "id": 98,
9     "name": "Dan Schneider",
10    "description": "Rich man",
11    "address": {
12      "street": "First Street",
13      "city": "San Francisco",
14      "state": "CA",
15      "zip": "94126"
16    }
17  },
18  "opponents": [
19    {
20      "id": 85,
21      "firstname": "jenny",
22      "lastname": "k",
23      "email": "c@gmail.com"
24    }
25  ]
}
```

Scenario 2: Not passing required parameters (400 status code expected)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?lastname=Mehrotra&email=apoovr56@gmail.com&description=Student&sponsordid=98>

The screenshot shows the Postman Builder interface. The URL in the header is <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?lastname=Mehrotra&email=apoovr56@gmail.com&description=Student&sponsordid=98>. The request method is PUT. The request body parameters are:

Key	Value	Description
firstname	Abhijeet	
lastname	Mehrotra	
email	apoovr56@gmail.com	
description	Student	
sponsordid	98	

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&email=apoovr56@gmail.com&description=Student&sponsorId=98>

The screenshot shows the Postman Builder interface. On the left, the sidebar lists various API endpoints under '275 Lab2'. The 'Update a player' endpoint is selected. In the main workspace, a PUT request is configured with the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&email=apoovr56@gmail.com&description=Student&sponsorId=98>. The request body contains the following data:

Key	Value	Description
firstname	Abhijeet	
lastname	Mehrotra	
email	apoovr56@gmail.com	
description	Student	
sponsorId	98	

The 'Authorization' tab is selected, showing 'No Auth'. The status bar at the bottom right indicates 'Status: 400 Bad Request' and 'Time: 245 ms'.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&lastname=Mehrotra&description=Student&sponsorId=98>

The screenshot shows the Postman Builder interface again. The 'Update a player' endpoint is selected. The PUT request is now configured with the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84?firstname=Abhijeet&lastname=Mehrotra&description=Student&sponsorId=98>. The request body contains the following data:

Key	Value	Description
firstname	Abhijeet	
lastname	Mehrotra	
email	apoovr56@gmail.com	
description	Student	
sponsorId	98	

The 'Authorization' tab is selected, showing 'No Auth'. The status bar at the bottom right indicates 'Status: 400 Bad Request' and 'Time: 213 ms'.

Scenario 3: Updating a player with invalid id (404 status code expected)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/1034?firstname=Abhijeet&lastname=Mehrotra&email=apoovr56@gmail.com&description=Student&sponsorid=98>

The screenshot shows the Postman interface. In the left sidebar, under 'Collections', there is a list of requests including 'Create a player', 'Get a player', 'Update a player' (which is selected), 'Delete a player', 'Opponent Creation', 'Create Sponsor', 'Get sponsor', 'Update sponsor', 'Delete sponsor', 'Delete opponent', and 'Menu Fetch API'. The main workspace shows a PUT request to <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/1034?firstname=Abhijeet&lastname=Mehrotra&email=apoovr56@gmail.com&description=Student&sponsorid=98>. The request body contains parameters: 'firstname' (Abhijeet), 'lastname' (Mehrotra), 'email' (apoovr56@gmail.com), 'description' (Student), and 'sponsorid' (98). The response status is 404 Not Found, and the response body is a JSON object with a single key 'message': "Player Not Exists!".

4. Delete a Player

Will delete a player with id 84

Fetching it to see the results

The screenshot shows the Postman interface. In the left sidebar, under 'Collections', there is a list of requests including 'Create a player', 'Get a player' (which is selected), 'Update a player', 'Delete a player', 'Opponent Creation', 'Create Sponsor', 'Get sponsor', 'Update sponsor', 'Delete sponsor', 'Delete opponent', and 'Menu Fetch API'. The main workspace shows a GET request to <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84>. The response status is 200 OK, and the response body is a JSON object representing a player with id 84, first name Apoorv, last name Mehrotra, email apoovr56@gmail.com, description Student, sponsor Dan Schneider, address 1st street, city san jose, state CA, zip 95110, and opponents including one with id 85, first name jenny, last name k, and email c@gmail.com.

Checking the database entries before deletion

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema CMPE275_LAB2 with tables SPONSOR, PLAYER, and OPPONENTS.
- Query Editor:** Displays the query `SELECT * FROM CMPE275_LAB2.PLAYER;`. The results grid shows the following data:

ID	firstname	lastname	email	description	sponsorid
84	Ahobject	Mehrotra	apoory12@gmail.com	Student	98
85	jenny	k	c@gmail.com	HALL	100
88	joe	biden	mr.biden@gmail.com	HALL	
95	Kevin2	Jonas	kevin12@gmail.com	HALL	
96	charles	zhang	charles123@gmail.com	President	HALL
106	check	name	check@name.com	new Description	98

- Output Panel:** Shows the execution history with the following log entries:

#	Time	Action	Message	Duration / Fetch
6	23:37:39	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.093 sec / 0.000 sec
7	23:39:48	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
8	23:40:41	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	12 row(s) returned	0.078 sec / 0.000 sec
9	23:54:20	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	12 row(s) returned	0.094 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema CMPE275_LAB2 with tables SPONSOR, PLAYER, and OPPONENTS.
- Query Editor:** Displays the query `SELECT * FROM CMPE275_LAB2.OPPONENTS;`. The results grid shows the following data:

ID	player1	player2
49	84	85
50	85	84
*	HALL	HALL

- Output Panel:** Shows the execution history with the following log entries:

#	Time	Action	Message	Duration / Fetch
7	23:39:48	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	9 row(s) returned	0.094 sec / 0.000 sec
8	23:40:41	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	12 row(s) returned	0.078 sec / 0.000 sec
9	23:54:20	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	12 row(s) returned	0.094 sec / 0.000 sec
10	23:54:46	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	2 row(s) returned	0.094 sec / 0.000 sec

Now deleting the player

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84>

The screenshot shows the Postman application interface. At the top, there are buttons for NEW, Runner, Import, and a search bar. The main area has tabs for Collections, History, and All. A collection named "275 Lab2" is selected, showing 10 requests. One request is highlighted: "POST Create a player". The request details show a DELETE method to the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/player/84>. The response status is 200 OK with a time of 355 ms. The response body is a JSON object:

```

1  {
2     "id": 84,
3     "firstname": "Abhijeet",
4     "lastname": "Mehrotra",
5     "email": "apoorv12@gmail.com",
6     "description": "Student",
7     "sponsor": {
8         "id": 98,
9         "name": "Dan Schneider",
10        "description": "Rich man",
11        "address": {
12            "street": "First Street",
13            "city": "San Francisco",
14            "state": "CA",
15            "zip": "94126"
16        }
17    },
18    "opponents": [
19        {
20            "id": 85,
21            "firstname": "jenny",
22            "lastname": "k",
23            "email": "c@gmail.com"
24        }
25    ]
}

```

Checking the database

The screenshot shows the MySQL Workbench interface. The left sidebar shows the Navigator with the schema CMPE275_LAB2 selected, containing Tables, Views, Stored Procedures, and Functions. The main area has tabs for SPONSOR, PLAYER, SQL File 6, and OPPONENTS. The SQL tab contains the query: `SELECT * FROM CMPE275_LAB2.OPPONENTS;`. The Results tab shows the output of the query:

	id	player1	player2
1	HULL	HULL	HULL
2	HULL	HULL	HULL
3	HULL	HULL	HULL

Below the results, the Object Info tab is visible.

MySQL Workbench

GlassDoor New cmpe275-lab2

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas SPONSOR PLAYER SQL File 6 OPPONENTS

1 • SELECT * FROM CMPE275_LAB2.PLAYER;

Result Grid Filter Rows: Edit Export/Import: Wrap Cell Content: Result Grid Form Editor Context Help Snippets

Administration Schemas Information Table: OPPONENTS Columns: id int AI PK player1 int player2 int

Action Output

#	Time	Action	Message	Duration / Fetch
9	23:54:20	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	12 row(s) returned	0.094 sec / 0.000 sec
10	23:54:46	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	2 row(s) returned	0.094 sec / 0.000 sec
11	23:55:51	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	0 row(s) returned	0.094 sec / 0.000 sec
12	23:56:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.093 sec / 0.000 sec

Scenario 2: Deleting a player with invalid id (404 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/1048>

Builder Team Library Apoorv Me... IN SYNC

History Collections All Me Team

Filter

Create a player Update a player Get a player Delete a player

DELETE http://ec2-18-205-192-46.compute-1.amazonaws.com/player/1048

Params Send Save

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 287 ms

Pretty Raw Preview JSON

```
{
  "message": "No class com.cmpe275.GameApp.Entity.Player entity with id 1048 exists!"
}
```

5. Create a sponsor

Scenario 1: Create a sponsor with all the information (200 status code and JSON object of sponsor)

http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor?name=Faye Oshima Belyeu&description=Producer&street=First Street&city=San Jose&state=CA&zip=95112

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a folder named '275 Lab2' containing 10 requests. One request, 'POST Create Sponsor', is highlighted. The main workspace shows a POST request to the URL `http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor?name=Faye Oshima Belyeu&description=Producer&street=First Street&city=San Jose&state=CA&zip=95112`. The 'Authorization' tab is selected, showing 'No Auth'. The 'Body' tab displays a JSON object:

```
1 {
2     "id": 115,
3     "name": "Faye Oshima Belyeu",
4     "description": "Producer",
5     "address": {
6         "street": "First Street",
7         "city": "San Jose",
8         "state": "CA",
9         "zip": "95112"
10    }
11 }
```

The response status is 200 OK and the time taken is 510 ms.

The screenshot shows the MySQL Workbench interface. A connection to 'GlassDoor New' is active. In the 'Navigator' pane, the 'SCHEMAS' section shows 'CMPE275_LAB2' selected. The 'Query 1' tab contains the SQL query `SELECT * FROM CMPE275_LAB2.SPONSOR;`. The 'Result Grid' pane displays the following data:

	id	name	description	street	city	state	zip
102	Dan Schneider	Rich man	First Street	NULL	CA	94126	
103	Dan Schneider	Rich man	NULL	San Francisco	CA	94126	
104	Dan Schneider	NULL	First Street	San Francisco	CA	94126	
105	Dan Schneider	NULL	NULL	CA	94126		
115	Faye Oshima Belyeu	Producer	First Street	San Jose	CA	95112	

The 'Output' pane at the bottom shows the execution history of the query:

#	Time	Action	Message	Duration / Fetch
10	23:55:46	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	2 row(s) returned	0.094 sec / 0.000 sec
11	23:55:51	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	0 row(s) returned	0.094 sec / 0.000 sec
12	23:56:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.093 sec / 0.000 sec
13	00:03:58	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	10 row(s) returned	0.078 sec / 0.000 sec

Scenario 2: Creating sponsor without required field name(400 status code)

http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor?description=Producer&street=First Street&city=San Jose&state=CA&zip=95112

The screenshot shows the Postman application interface. On the left, the sidebar lists various collections and requests. The 'Create Sponsor' request under 'POST' is selected. The 'Body' tab contains a JSON object with fields: 'name' (value: George Lucas), 'description' (value: Producer), 'street' (value: First Street), 'city' (value: San Jose), 'state' (value: CA), and 'zip' (value: 95112). The 'Authorization' tab shows 'No Auth'. The status bar at the bottom indicates a 'Status: 400 Bad Request' and a 'Time: 240 ms'.

6. Get a sponsor

Scenario 1: Getting a sponsor with a valid id 115 (200 status code and JSON object of sponsor)

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane displays the 'CMPE275_LAB2' database. The 'Query 1' pane contains the SQL query: 'SELECT * FROM CMPE275_LAB2.SPONSOR;'. The 'Result Grid' pane shows the following data:

ID	Name	Description	Street	City	State	Zip
102	Dan Schneider	Rich man	First Street	NULL	CA	94126
103	Dan Schneider	Rich man	NULL	San Francisco	CA	94126
104	Dan Schneider	NULL	First Street	San Francisco	CA	94126
105	Dan Schneider	NULL	NULL	NULL	CA	94126
115	Faye Oshema Belyeu	Producer	First Street	San Jose	CA	95112

The status bar at the bottom indicates a 'Status: 200 OK' and a 'Time: 00:08'.

Fetching a sponsor with id as 115

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115>

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'Collections' tab selected, showing a list of API endpoints for '275 Lab2'. The 'Get sponsor' endpoint is highlighted. In the main workspace, a 'Get sponsor' request is configured with the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115>. The response status is 200 OK, and the JSON body contains the details of a sponsor with ID 115.

```
1 {  
2   "id": 115,  
3   "name": "Faye Oshima Belyeu",  
4   "description": "Producer",  
5   "address": {  
6     "street": "First Street",  
7     "city": "San Jose",  
8     "state": "CA",  
9     "zip": "95112"  
10    },  
11    "players": []  
12 }
```

Scenario 2: Fetching a sponsor with invalid id (404 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/1983>

The screenshot shows the Postman application interface. The 'Get sponsor' endpoint is highlighted. In the main workspace, a 'Get sponsor' request is configured with the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/1983>. The response status is 404 Not Found, and the JSON body indicates that the sponsor ID does not exist.

```
1 {  
2   "message": "Sponsor Id Does Not Exist!"  
3 }
```

7. Update a sponsor

Scenario 1: Update a sponsor with all the details (200 status code and updated JSON object of sponsor)

Updating the sponsor with id 115

Original data

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with the SPONSOR table selected. The main pane shows a result grid for the query: "SELECT * FROM CMPE275_LAB2.SPONSOR;". The grid contains the following data:

	id	name	description	street	city	state	zip
102	Dan Schneider	Rich man	First Street	HULL	CA	94126	
103	Dan Schneider	Rich man	HULL	San Francisco	CA	94126	
104	Dan Schneider	HULL	First Street	San Francisco	CA	94126	
105	Dan Schneider	HULL	HULL	CA	94126		
115	Faye Oshima Belyeu	Producer	First Street	San Jose	CA	95112	

The status bar at the bottom right indicates the date and time: 22/11/2020 00:17.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115?name=Nick&Robinson&description=Actor&street=Santa%20Clara&city=San%20Jose&state=CA&zip=95126>

The screenshot shows the Postman application interface. The top navigation bar includes buttons for NEW, Runner, Import, and a search bar. The main workspace shows a collection named "275 Lab2" with one request. The request details are as follows:

- Method:** PUT
- URL:** <http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115?name=Nick&Robinson&description=Actor&street=Santa%20Clara&city=San%20Jose&state=CA&zip=95126>
- Type:** No Auth
- Body:** JSON (Pretty) view showing the updated sponsor data:

```

1 - {
2   "id": 115,
3   "name": "Nick Robinson",
4   "description": "Actor",
5   "address": {
6     "street": "Santa Clara",
7     "city": "San Jose",
8     "state": "CA",
9     "zip": "95126"
10   }
11 }

```

The status bar at the bottom right indicates the date and time: 22/11/2020 00:18.

Checking the database

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Navigator' with 'SCHEMAS' expanded to show 'CMPE275_LAB2' containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The main area has tabs for 'SPONSOR', 'PLAYER', 'SQL File 6', and 'OPPONENTS'. A query window titled 'Query 1' contains the SQL command: 'SELECT * FROM CMPE275_LAB2.SPONSOR;'. The results grid shows the following data:

	id	name	description	street	city	state	zip
102	Dan Schneider	Rich man	Fist Street	HULL	CA	94126	
103	Dan Schneider	Rich man	HULL	San Francisco	CA	94126	
104	Dan Schneider	HULL	Fist Street	San Francisco	CA	94126	
105	Dan Schneider	HULL	HULL	HULL	CA	94126	
115	Nick Robinson	Actor	Santa Clara	San Jose	CA	95126	

The 'Information' pane shows the 'Table: OPPONENTS' with columns: id (int AI PK), player1 (int), and player2 (int). The 'Output' pane displays the execution history of the previous queries.

Scenario 2: Updating the sponsor with invalid/ non-existing id (404 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/1045?name=NickRobinson&description=Actor&street=Santa Clara&city=San Jose&state=CA&zip=95126>

The screenshot shows the Postman application interface. The left sidebar lists collections: 'History' (275 Lab2, 10 requests) and 'Collections' (All, Me, Team). The main area shows a 'PUT' request to the URL: 'http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/1045?name=NickRobinson&description=Actor&street=Santa Clara&city=San Jose&state=CA&zip=95126'. The 'Body' tab shows the JSON payload: { "message": "Sponsor Id Does Not Exist!" }. The response status is '404 Not Found' and the time taken is '261 ms'.

Scenario 3: Updating the sponsor without required name parameter (400 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115?description=Actor&street=SantaClara&city=San Jose&state=CA&zip=95126>

The screenshot shows the Postman interface with a PUT request to <http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/115?description=Actor&street=SantaClara&city=San Jose&state=CA&zip=95126>. The request body includes:

Key	Value	Description
name	Nick Robinson	
description	Actor	
street	Santa Clara	
city	San Jose	
state	CA	
zip	95126	

The response status is 400 Bad Request.

8. Delete a sponsor

Scenario 1: Delete a sponsor with invalid/ non-existing id (404 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/5612>

The screenshot shows the Postman interface with a DELETE request to <http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/5612>. The response status is 404 Not Found. The JSON response body is:

```
1 <pre>{</pre>
2   "message": "No class com.cmpe275.GameApp.Entity.Sponsor entity with id 5612 exists!"</pre>
3 }
```

Scenario 2: Deleting a sponsor with valid id (200 status code and JSON object of deleted sponsor)

Deleting a sponsor with id 98

Checking the database before deletion

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has a Navigator section with SCHEMAS and CMPE275_LAB2 tables, views, stored procedures, and functions. The main query editor window displays the following SQL query:

```
1 • SELECT * FROM CMPE275_LAB2.SPONSOR;
```

The Result Grid shows the following data:

	id	name	description	street	city	state	zip
5	pranjay	sagar		1st street	san jose	CA	95134
74	david			HULL	HULL	HULL	HULL
98	Dan Schneider	Rich man		First Street	San Francisco	CA	94126
99	Dan Schneider	Rich man		First Street	San Francisco	CA	94126
100	Dan Schneider			1st street	san jose	CA	95110
102	Dan Schneider	Rich man		First Street	HULL	CA	94126

The bottom pane shows the Output log with the following entries:

#	Time	Action	Message	Duration / Fetch
11	23:55:51	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	0 row(s) returned	0.094 sec / 0.000 sec
12	23:56:08	SELECT * FROM CMPE275_LAB2.PLAYER LIMIT 0, 1000	11 row(s) returned	0.093 sec / 0.000 sec
13	00:03:58	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	10 row(s) returned	0.078 sec / 0.000 sec
14	00:18:37	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	10 row(s) returned	0.093 sec / 0.000 sec

<http://ec2-18-205-192-46.compute-1.amazonaws.com/sponsor/98>

The screenshot shows the Postman application interface. The top header includes NEW, Runner, Import, and a search bar. The main workspace shows a DELETE request to the URL above. The request body is a JSON object:

```
1 {  
2   "id": 98,  
3   "name": "Dan Schneider",  
4   "description": "Rich man",  
5   "address": {  
6     "street": "First Street",  
7     "city": "San Francisco",  
8     "state": "CA",  
9     "zip": "94126"  
10 },  
11   "players": [  
12     {  
13       "id": 106,  
14       "firstname": "check",  
15       "lastname": "name",  
16       "email": "check@name.com",  
17       "description": "new Description"  
18     },  
19     {  
20       "id": 109,  
21       "firstname": "Carly",  
22       "lastname": "Shay",  
23       "email": "carly@gmail.com",  
24       "description": "web presenter"  
25     }  
26   ]  
27 }
```

The bottom status bar shows the time as 00:29 and the date as 22/11/2020.

Checking the database after deletion of sponsor with id 98

MySQL Workbench Screenshot:

- Navigator:** Shows SCHEMAS and CMPE275_LAB2.
- Query Editor:** Displays the query: `SELECT * FROM CMPE275_LAB2.SPONSOR;`
- Result Grid:** Shows the results of the query, including the deleted row (id 98).
- Output:** Shows the execution history with log entries for each SELECT statement.

9. Add an opponent

Scenario 1: Providing invalid ids and trying to make them opponents (**404 status code**)

Valid id 112 and invalid id 1024

<http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/112/1024>

Postman Screenshot:

- Collections:** Opponent Creation
- Authorization:** Type: No Auth
- Body:** Body tab shows a JSON response: `{"message": "Player Not Exists!"}`

Invalid id 1024 and valid id 112

<http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/1024/112>

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'Collections' tab selected, displaying various API endpoints for player and sponsor management. The main area shows a 'PUT' request to the URL `http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/1024/112`. The 'Authorization' tab is selected, showing 'No Auth'. The 'Body' tab contains a JSON payload:

```
1 {  
2   "message": "Player Not Exists!"  
3 }
```

The status bar at the bottom right indicates a 404 Not Found error and a time of 133 ms.

Scenario 2: Create two players as opponents of each other (200 status code and informative text message)

Taking players with id 109 and 112

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'cmpe275-lab2'. The 'PLAYER' tab is active, showing a query window with the command `SELECT * FROM CMPE275_LAB2.PLAYER;`. Below the query window, the 'Result Grid' shows the following data:

ID	First Name	Last Name	Email	Description	Sponsor ID
109	Carly	Shay	carly@gmail.com	web presenter	98
110	Apoorv	Mehrrota	apoovr1@gmail.com	Student	NULL
111	Spencer	Shay	spenser@gmail.com	Sculptor	NULL
112	Alex	Shay	alex@gmail.com	Actor	5
113	Justin1	Shay	justin@gmail.com	Actor	NULL

In the bottom right corner of the result grid, there is a note: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

Before opponents created

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'CMPE275_LAB2' selected. Under 'Tables', there is one entry: 'OPPONENTS'. The main pane shows a query editor with the SQL command: `SELECT * FROM CMPE275_LAB2.OPPONENTS;`. Below the query is a result grid table:

	id	player1	player2
*	MULL	MULL	MULL

On the right side, a message box states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The bottom status bar shows the date and time: 22/11/2020 00:38.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/109/112>

The screenshot shows the Postman application interface. The left sidebar lists collections and requests. The 'Opponent Creation' request is highlighted. The details panel shows a 'PUT' request to the URL: `http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/109/112`. The 'Authorization' tab is selected, showing 'Type: No Auth'. The 'Body' tab contains the response body: "Opponent created". The status bar at the bottom shows the date and time: 22/11/2020 00:37.

Checking the database now after opponent creation

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'CMPE275_LAB2' selected. The main pane shows the 'OPPONENTS' table with the following data:

	id	player1	player2
59	109	112	
60	112	109	

The status bar at the bottom right indicates the date and time: 22/11/2020 00:39.

<http://ec2-18-205-192-46.compute-1.amazonaws.com/player/109>

The screenshot shows the Postman application interface. The left sidebar lists various API endpoints. The main panel shows a successful GET request to the specified URL, with the response body displayed in JSON format:

```

1 {
2   "id": 109,
3   "firstname": "Carly",
4   "lastname": "Shay",
5   "email": "carly@gmail.com",
6   "description": "web presenter",
7   "opponents": [
8     {
9       "id": 112,
10      "firstname": "Alex",
11      "lastname": "Shay",
12      "email": "alex@gmail.com",
13      "description": "Actor"
14    }
15  ]
16 }

```

The status bar at the bottom right indicates the date and time: 22/11/2020 00:42.

Scenario 3: Making two players opponents who are already each other's opponents (400 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/109/112>

The screenshot shows the Postman application interface. In the left sidebar, under the 'Collections' tab, there is a list of API endpoints. One endpoint, 'Opponent Creation' (PUT), is highlighted. The main workspace shows a 'PUT' request to the URL <http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/109/112>. The 'Authorization' tab is selected, showing 'No Auth'. The 'Body' tab is selected, displaying a JSON response:

```
1 + {  
2     "message": "Players Are Already Opponents!"  
3 }
```

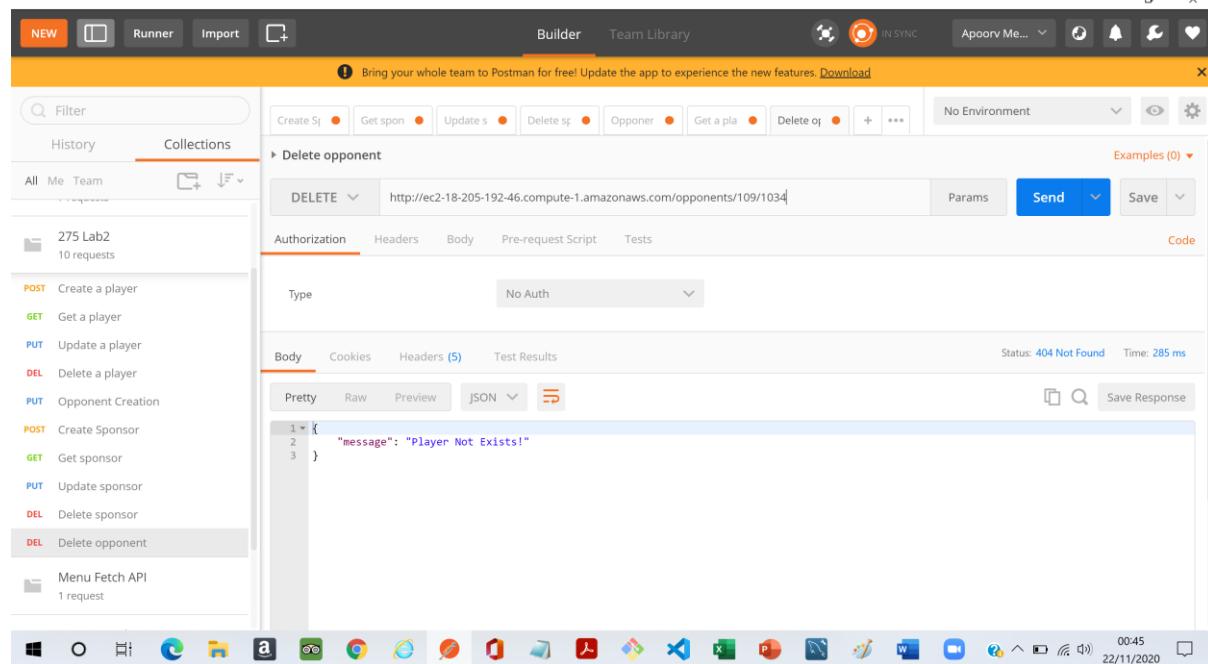
The status bar at the bottom right indicates 'Status: 400 Bad Request' and 'Time: 301 ms'. The taskbar at the bottom of the screen shows various open applications.

10. Remove an opponent

Scenario 1: Removing opponents with invalid ids (404 status code)

Taking valid id 109 and invalid id 1034

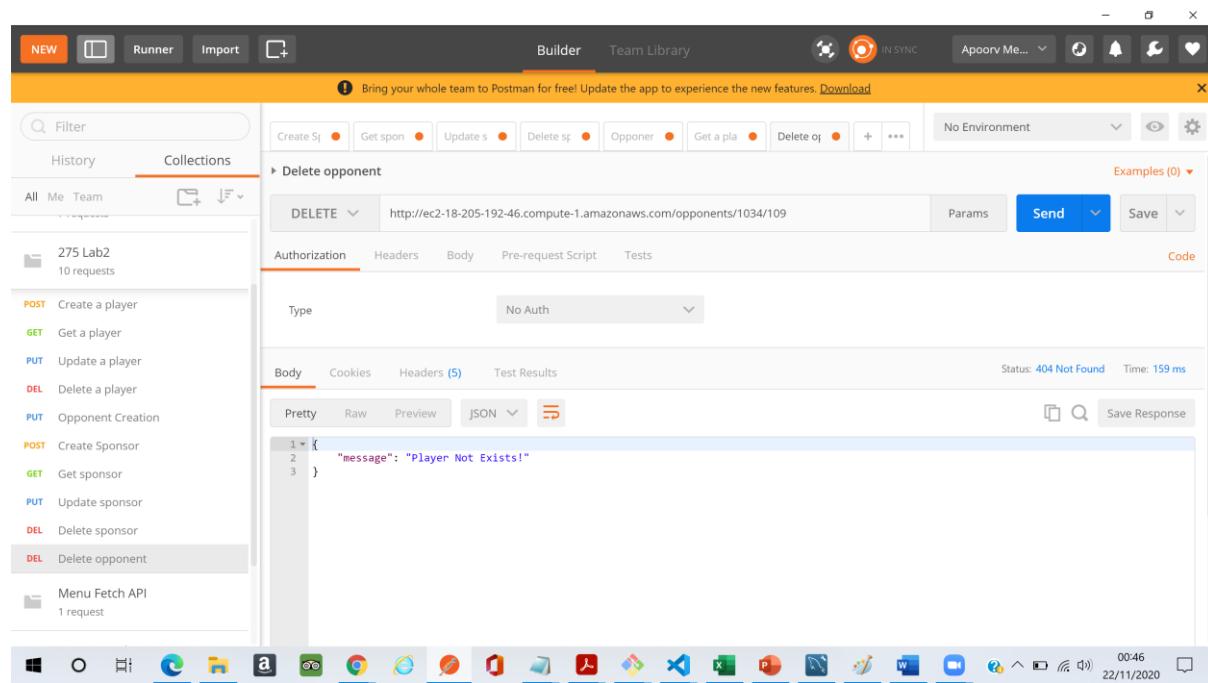
<http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/109/1034>



The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'Collections' tab selected, showing various API endpoints like 'Create a player', 'Get a player', etc. In the main area, a 'Delete opponent' request is being viewed. The URL field contains 'http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/109/1034'. The 'Body' tab is selected, showing a JSON response with the message 'Player Not Exists!'. The status bar at the bottom indicates 'Status: 404 Not Found' and 'Time: 285 ms'. The timestamp is 22/11/2020 00:45.

Taking invalid id 1034 and valid id 109

<http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/1034/109>



This screenshot is identical to the one above, showing the Postman interface with a 'Delete opponent' request. The URL is now 'http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/1034/109'. The response body still shows 'Player Not Exists!', indicating the endpoint is correctly handling invalid IDs. The status bar shows 'Status: 404 Not Found' and 'Time: 159 ms'. The timestamp is 22/11/2020 00:46.

Scenario 2: Removing opponents when two players are not opponents (400 status code)

<http://ec2-18-205-192-46.compute-1.amazonaws.com/oppone...nts/110/111>

The screenshot shows the Postman application interface. On the left, there's a sidebar with various API endpoints listed under 'Collections'. The main area shows a 'DELETE' request to the URL provided in the question. The 'Body' tab contains a JSON response with the message 'Players Are Not Opponents!'. The status bar at the bottom indicates a 'Bad Request' status.

Scenario 3: Removing opponents (200 status code and a meaningful message)

Checking the database

The screenshot shows the MySQL Workbench interface. In the 'Navigator' panel, the 'SCHEMAS' section shows the 'CMPE275_LAB2' schema. The 'Tables' section lists the 'OPPONENTS' table. The 'Query 1' tab displays the result of the query 'SELECT * FROM CMPE275_LAB2.OPPONENTS;'. The results are shown in a grid:

#	Time	Action	Message	Duration / Fetch
13	00:03:58	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	10 row(s) returned	0.078 sec / 0.000 sec
14	00:18:37	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	10 row(s) returned	0.093 sec / 0.000 sec
15	00:30:49	SELECT * FROM CMPE275_LAB2.SPONSOR LIMIT 0, 1000	8 row(s) returned	0.078 sec / 0.000 sec
16	00:39:05	SELECT * FROM CMPE275_LAB2.OPPONENTS LIMIT 0, 1000	2 row(s) returned	0.078 sec / 0.000 sec

POST Create a player

GET Get a player

PUT Update a player

DEL Delete a player

PUT Opponent Creation

POST Create Sponsor

GET Get sponsor

PUT Update sponsor

DEL Delete sponsor

DEL Delete opponent

Menu Fetch API

<http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/109/112>

DELETE http://ec2-18-205-192-46.compute-1.amazonaws.com/opponents/109/112

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a section for '275 Lab2' containing several API endpoints. One endpoint, 'Get a player' (GET), is highlighted. The main workspace displays a successful GET request to the URL `http://ec2-18-205-192-46.compute-1.amazonaws.com/player/109`. The response body is shown in JSON format:

```

1 {
2   "id": 109,
3   "firstname": "Carly",
4   "lastname": "Shay",
5   "email": "carly@gmail.com",
6   "description": "web presenter",
7   "opponents": []
8 }

```

Checking the database

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane shows the 'SCHEMAS' section with 'CMPE275_LAB2' selected. The 'Query' pane displays the following SQL query:

```
1 • SELECT * FROM CMPE275_LAB2.OPPONENTS;
```

The 'Result Grid' pane shows the results of the query:

	id	player1	player2
1	NULL	NULL	NULL
2	NULL	NULL	NULL

The status bar at the bottom right indicates the time as 00:51 and the date as 22/11/2020.