SOFTWARE ENGINEERING - HW5(gr485)

Files:

Code is written in sever.js
Testcases are written in codespec.spec.js

Commands:

```
node server.js
npx jasmine codespec.spec.js
```

Postman Requests

Creating a record to the table

Type: POST
Headers: `Content-Type: application/json`
URL: http://localhost:3000
Body format:

```
{
"name": "Test A1",
"city": "New York",
"school": "High School A"
}
```

Type: GET
Headers: `Content-Type: application/json`
URL: http://localhost:3000
Shows all the records from the table

Type: DELETE
Headers: `Content-Type: application/json`
URL: http://localhost:3000/{id}
(OR)
We can try with any of the individual or combination of parameters as well
Id, name, school, city

```
//id:60
name:User 1
//school:High School A
//city:New City
```

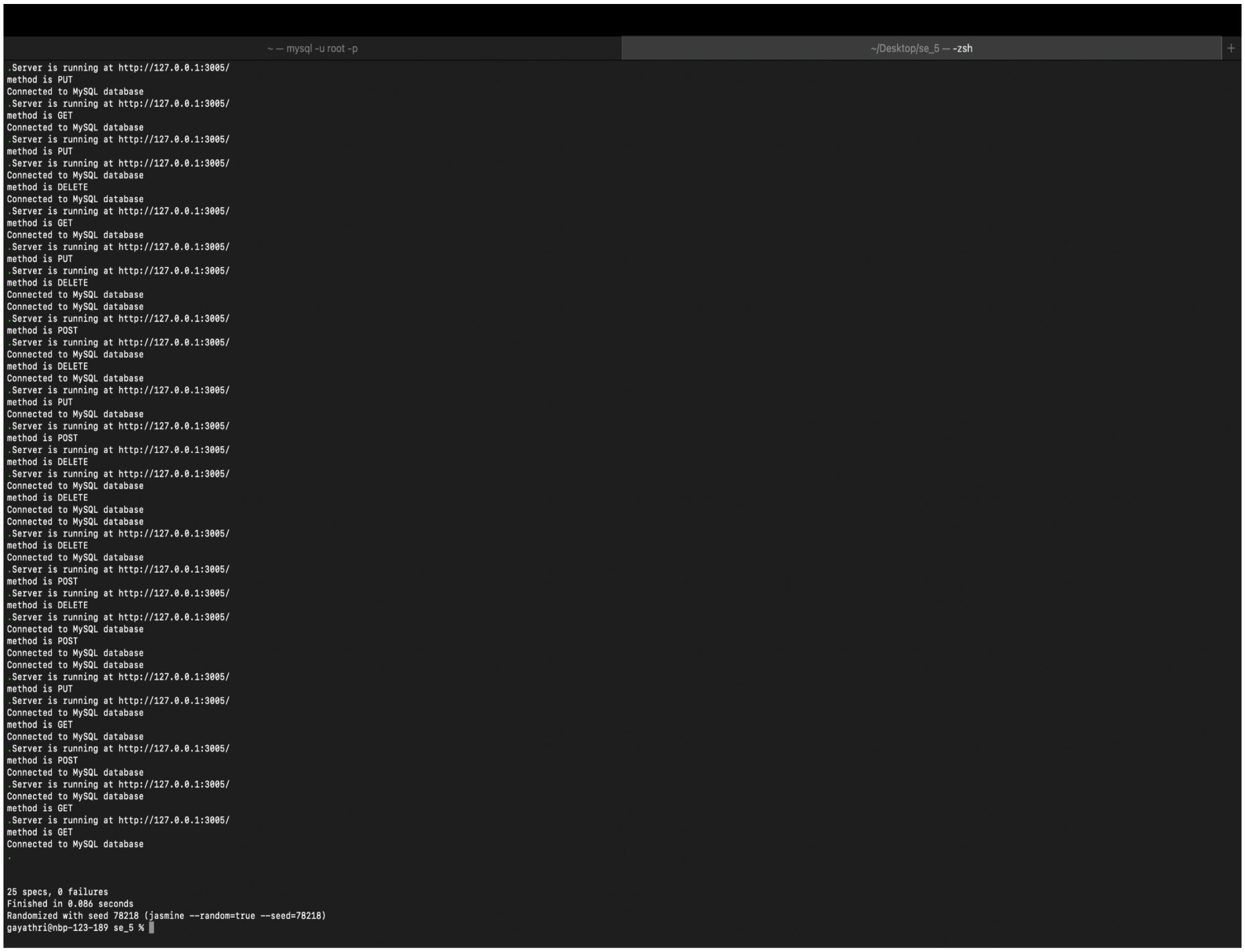
Type: PUT

Headers: `Content-Type: application/json`
URL: http://localhost:3000/{id}

```
{
"id": 3, // Replace with the actual ID of the record you want to update
"name": "Updated Name",
"city": "Updated City",
"school": "Updated School"
}
```

Testcases:

1. Test case: should respond with a specific resource for a valid GET request with ID
2. Test case: should respond with 404 Resource Not Found for a GET request with an invalid ID
3. Test case: should respond with all resources for a valid GET request without ID
4. Test case: should respond with a single resource for a valid GET request with an ID parameter
5. Test case: should respond with all resources for a valid GET request without query parameters
6. Test case: should respond with filtered resources for a valid GET request with query parameters
7. Test case: should respond with all resources for a valid GET request without query parameters
8. Test case: should respond with a 201 status code for a valid POST request with a single record
9. Test case: should respond with 413 Request Entity Too Large for a large POST request body
10. Test case: should respond with a 201 status code for a valid POST request with multiple records
11. Test case: should respond with a 400 status code for invalid JSON data in the POST request
12. Test case: should respond with 200 OK for a valid PUT request
13. Test case: should respond with 400 Bad Request for invalid JSON data in the PUT request
14. Test case: should respond with 413 Request Entity Too Large for a large PUT request body
15. Test case: should respond with 400 Bad Request for an empty request body
16. Test case: should respond with a 404 status code for a non-existing resource in the PUT request
17. Test case: should respond with 413 Request Entity Too Large for a large JSON request body
18. Test case: should respond with 400 Bad Request for an empty request body
19. Test case: should respond with 400 Bad Request for invalid JSON data with JSON content-type
20. Test case: should respond with a 200 status code for a valid DELETE request with an existing ID
21. Test case: should respond with a 400 status code for a DELETE request without an ID parameter
22. Test case: should respond with a 404 status code for a DELETE request with a non-existing ID
23. Test case: should respond with a 405 status code for an unsupported DELETE request with invalid parameters
24. Test case: should respond with a 200 status code for a DELETE request with valid name, city, and school parameters
25. Test case: should respond with a 404 status code for a DELETE request with a combination of valid and invalid parameters
26. Test case: should respond with a 415 status code for a DELETE request with invalid content-type



```
.Server is running at http://127.0.0.1:3005/
method is PUT
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is GET
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is PUT
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is DELETE
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is GET
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is PUT
.Server is running at http://127.0.0.1:3005/
method is DELETE
Connected to MySQL database
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is POST
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is DELETE
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is PUT
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is POST
.Server is running at http://127.0.0.1:3005/
method is DELETE
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is DELETE
Connected to MySQL database
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is DELETE
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is POST
.Server is running at http://127.0.0.1:3005/
method is DELETE
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is POST
Connected to MySQL database
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is PUT
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is GET
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
method is POST
Connected to MySQL database
.Server is running at http://127.0.0.1:3005/
Connected to MySQL database
method is GET
.Server is running at http://127.0.0.1:3005/
method is GET
Connected to MySQL database
.


25 specs, 0 failures
Finished in 0.086 seconds
Randomized with seed 78218 (jasmine --random=true --seed=78218)
gayathri@nbp-123-189 se_5 %
```

Database:

```
CREATE DATABASE se5;
USE se5;
CREATE TABLE user (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   name VARCHAR(255) NOT NULL,
    ->   city VARCHAR(255),
    ->   school VARCHAR(255)
    -> );
INSERT INTO user (name, city, school) VALUES
    -> ('John', 'New York', 'High School A'),
    -> ('Alice', 'Los Angeles', 'High School B'),
    -> ('Bob', 'Chicago', 'High School C'),
    -> ('Eva', 'San Francisco', 'High School D'),
    -> ('Mike', 'Miami', 'High School E'),
    -> ('Sarah', 'Seattle', 'High School F'),
    -> ('David', 'Dallas', 'High School G'),
    -> ('Emily', 'Houston', 'High School H'),
    -> ('Kevin', 'Boston', 'High School I'),
    -> ('Olivia', 'Phoenix', 'High School J');

select * from user;
```