CS 310 : Scalable Software Architectures

Class session on Thursday, October 10th

October 2024

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

www.a-printable-calendar.com

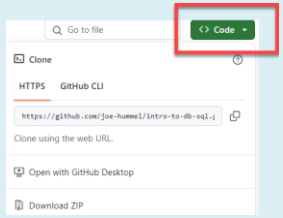
Notes:

- ***Focus this week:***
  - *Web services*
- ***Class sessions *are* being recorded this week***
  - *Will be available under Panopto on Canvas*
- ***Project 01** was due yesterday (Wednesday)*
  - *Can submit as late as Friday @ 11:59pm*

Northwestern University

Getting the necessary software

1. **Make sure Docker Desktop is running**
2. **Download files you need for today**
   - https://github.com/joe-hummel/intro-web-services-server
   - https://github.com/joe-hummel/intro-web-services-client



3. **You want to open 2 terminal windows, one based on server repo and one based on client repo:**

Linux/Mac/Windows WSL:

1) Open terminal, navigate to repo folder
2) `chmod 755 *.bash`
3) `./docker-build.bash`
4) `./docker-run.bash`

Windows:

1) Open Powershell, navigate to repo folder
2) `.\docker-build.bat`
3) `.\docker-run.bat`

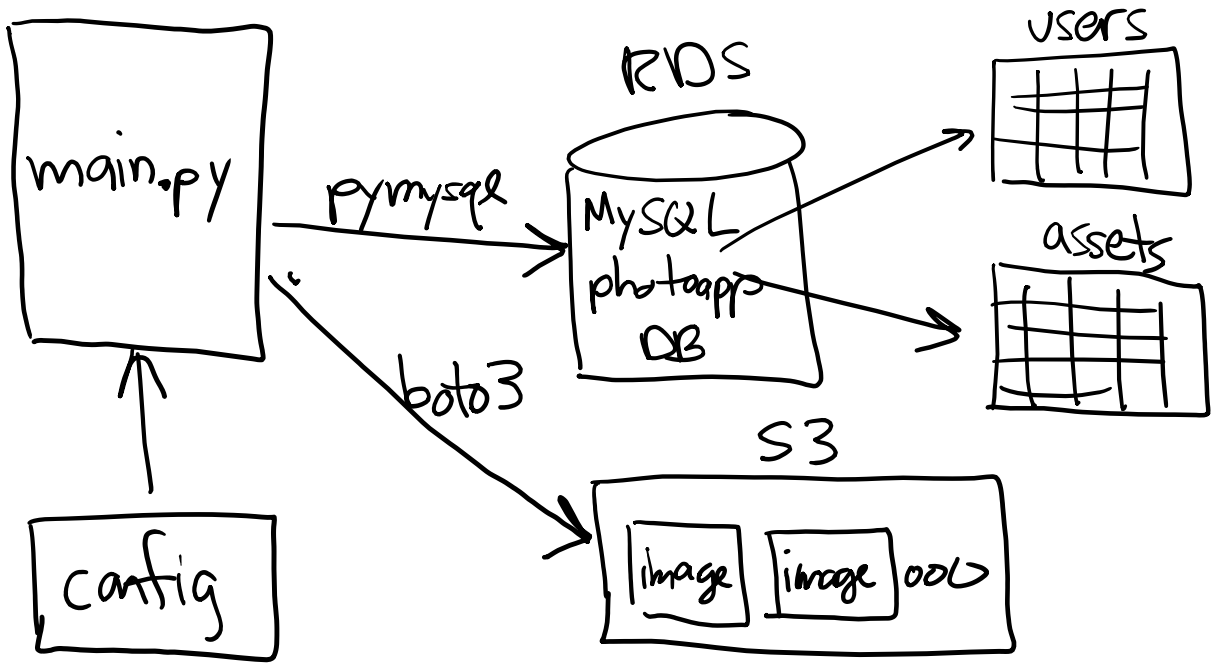
Common docker errors

1. **"docker" command not found**
   - *Uninstall and reinstall Docker Desktop*

2. **When you try to build, you are not authorized**
   - *`docker login -u docker-username`*

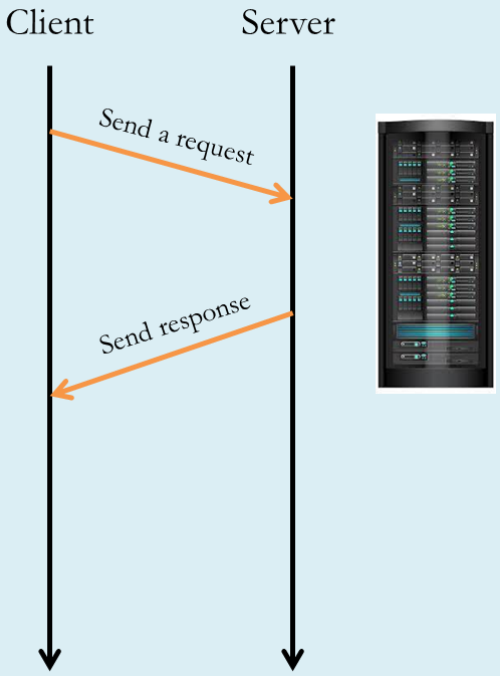
3. **When you try to run, you get errors like "bash: $\r: command not found"**
   1. *If you see the **docker>** prompt, type **exit***
   2. `((Get-Content .bashrc) -join "`n") + "`n" | Set-Content -NoNewLine .bashrc`

Project 01 => Project 02



Goals for today

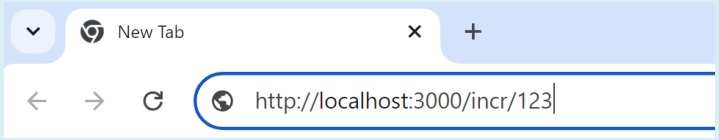
- Web service examples
- First a simple calculator service
- Then a more realistic one based on movielens DB



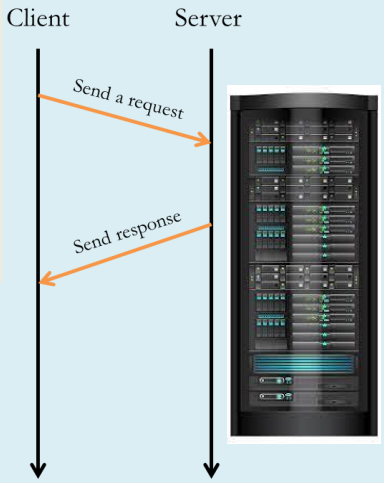

EXAMPLE

Example #1

- **A simple calculator web service:**



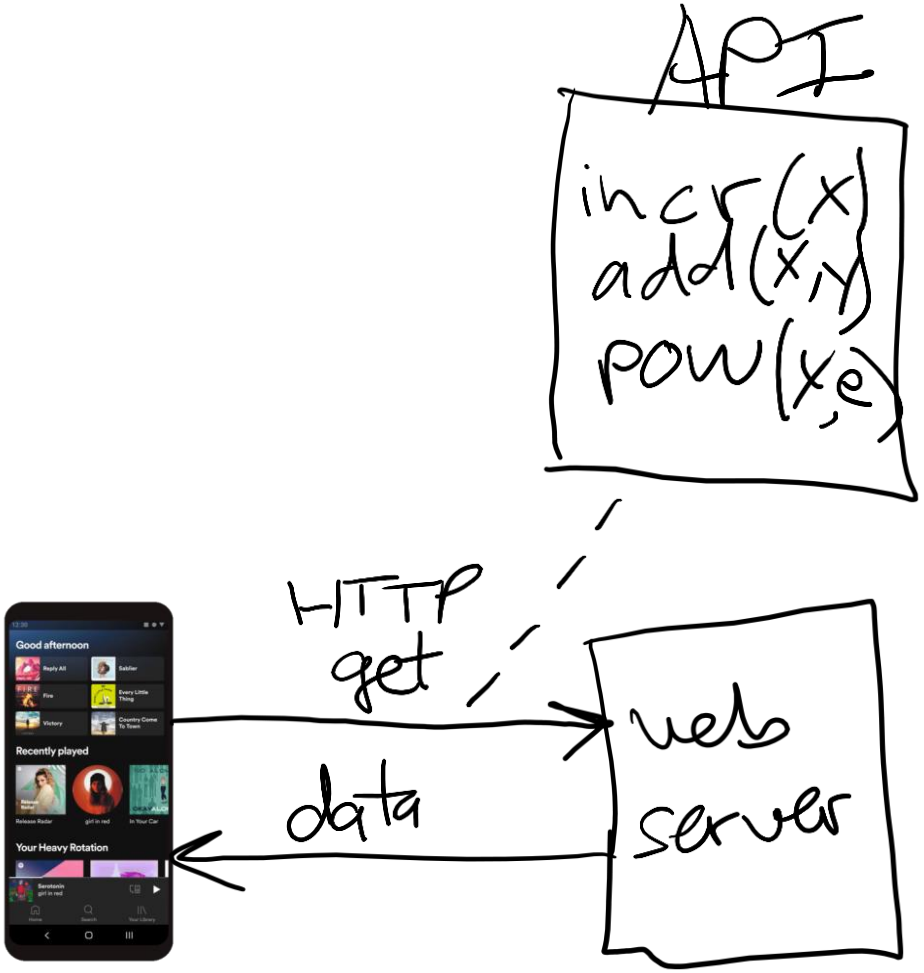

```
import requests

url = 'http://localhost:3000/'

x = input('Enter integer x> ')
url = url + 'incr/' + x
response = requests.get(url)
print(response.text)
```




```
const express = require('express');
const app = express();

// main():
app.listen(3000, () => {
  console.log('**SERVER: running...');
});

// increment x:
app.get('/incr/:x', (req, res) => {

  try {
    console.log('**call to /incr');
    let x = parseInt(req.params.x);
    if (isNaN(x))
      throw new Error('x not a number');
    let y = x + 1;
    res.send(y.toString());
    return;
  }
  catch(err) {
    res.status(400).send(err.message);
  }

});
```

Simple SOA example

- **Simple calculator web service…**




```
// increment x:
app.get('/incr/:x', (req, res) => {…});

// add x and y:
app.get('/add/:x/:y', (req, res) => {…});

// raise x to the exponent e:
app.get('/pow/:x/:e', (req, res) => {…});
```

add(x, y) and pow(x, e)

- **Edit "server.js" and implement the two functions…**

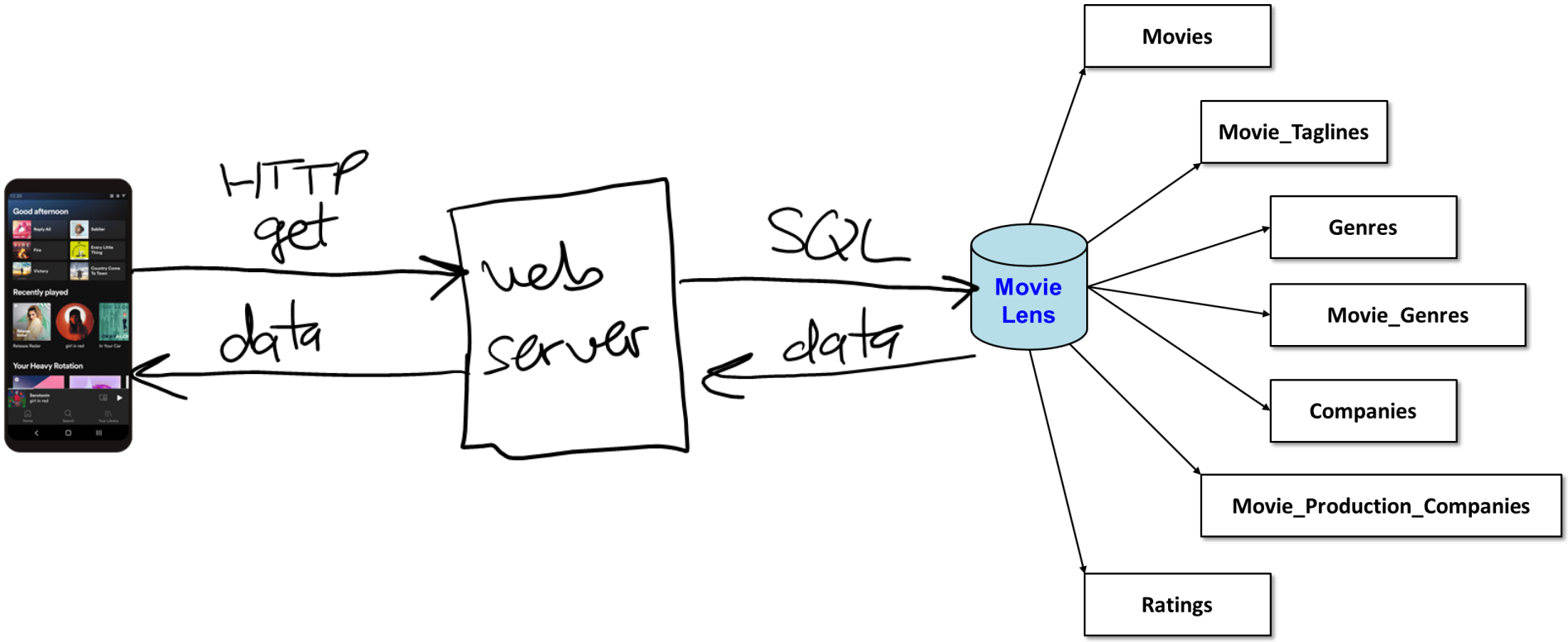
```
45  //
46  // add(x, y)
47  //
48  app.get('/add/:x/:y', (req, res) => {
49
50    //
51    // TODO: assume integers x and y (use parseInt)
52    //
53
54  });
55
```

```
56  //
57  // pow(x, e)
58  //
59  app.get('/pow/:x/:e', (req, res) => {
60
61    //
62    // TODO: assume x and e can be floating-point (use parseFloat)
63    //
64
65  });
66
```

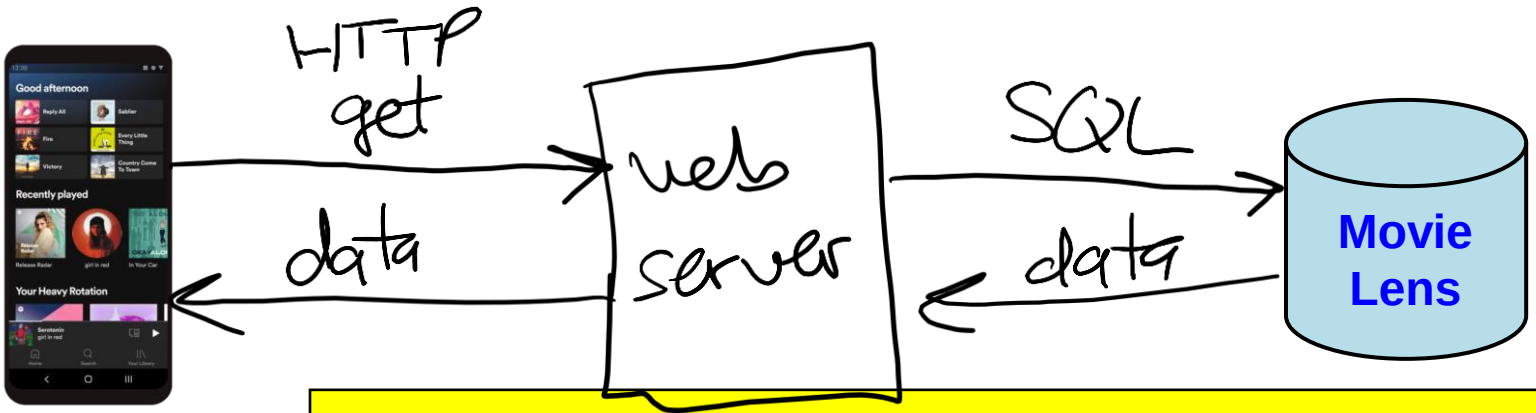

EXAMPLE

Example #2

- **Web service for the MovieLens database**

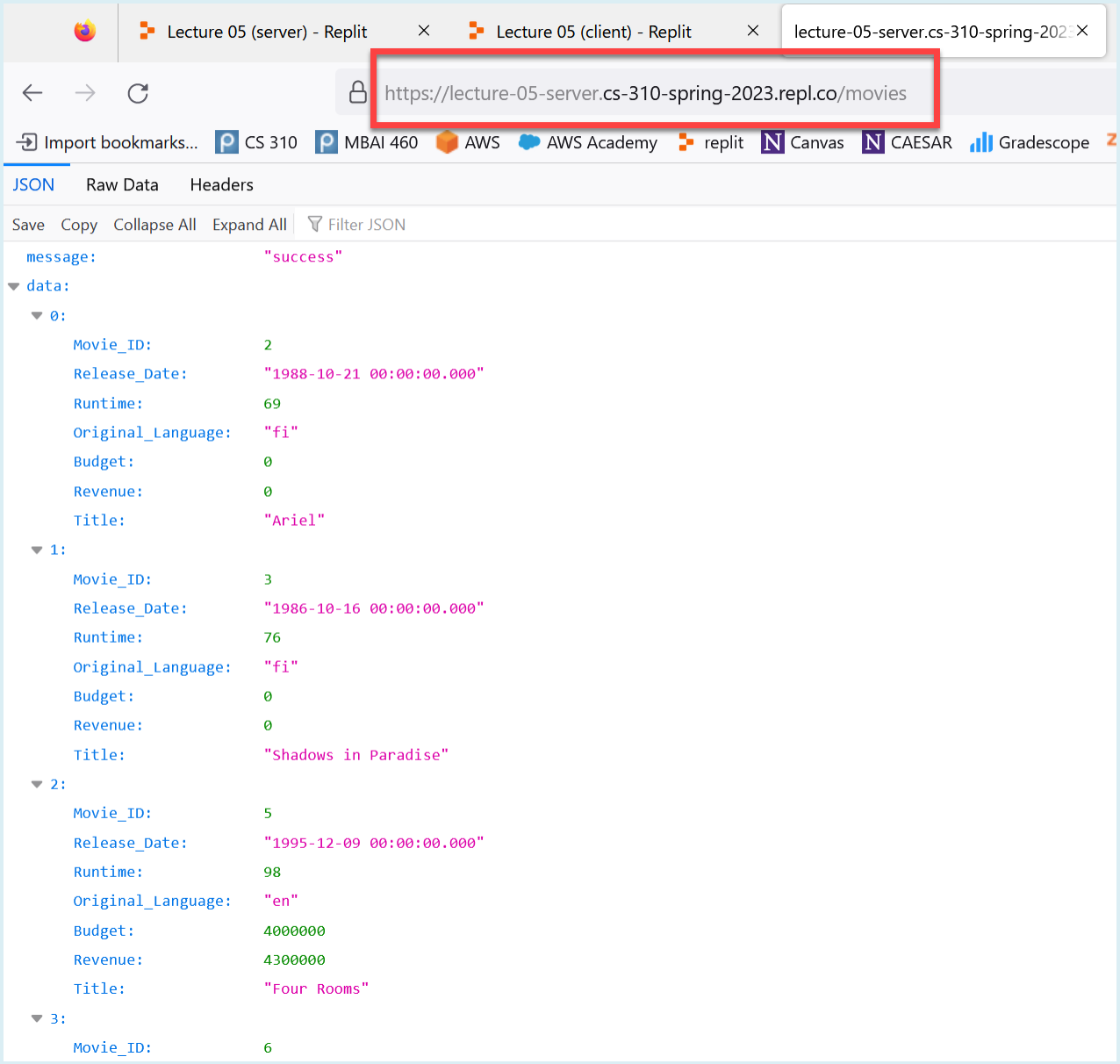


MovieLens web service




```
app.get('/movies', (req, res) => {…});

app.get('/movies/top10', (req, res) => {…});

// top N with at least M reviews (defaults to N=10, M=100):
app.get('/movies/topNwithM', (req, res) => {…});

// top N with at least M reviews, in given genre:
app.get('/movies/topNwithM/:genre', (req, res) => {…});

.
.
.
```

Demo using browser as client



/movies


```
//
// Retrieve all movies in the database:
//
app.get('/movies', (req, res) => {
  try {
     console.log("**call to /movies");

     let sql = "Select * From Movies Order By Movie_ID;";
     let params = [];

     // execute the SQL:
     movielens.all(sql, params, (err, rows) => {
         if (err) {
           res.status(500).json( {"message": err.message, "data": []} );
           return;
         }

         // send response in JSON format:
         console.log("sending response");
         res.json( {"message": "success", "data": rows} );
     });

     console.log("about to return");
     return;
   }
   catch(err) { res.status(500).json({"message": err.message, "data": []}); }
});
```

/movies/:id

- **Let's write a function to retrieve a movie by id:**
  - *Edit "server2.js", add to bottom…*
  - *Return the data as a dictionary in JSON format…*

```
//
// Retrieve the specified movie based on movie id:
//
app.get('/movies/:id', (req, res) => {
  try {
     console.log("**call to /movies/:id");

     let id  = ???;
     let sql = "Select * from ...";
     let params = [id];

     // execute the SQL and return the result (or an error):
     movielens.get(???

     });

     return;
   }
   catch(err) { res.status(500).json({ "message": err.message, "data": {} }); }
});
```

Client testing

- **Use a browser to test…**
  - *Test a movie id that exists: 862 or 603*
  - *Test a movie id that doesn't exist: 123456789*

- **Status code should be 200 in both cases --- they both executed successfully, one just didn't find a movie**
  - *If the movie exists, data be { "Movie_ID":…, … }*
  - *If the movie doesn't exists, change msg, data should be { }*
  - *[or should we return status code 400 if movie doesn't exist?]*

Solution /movies/:id

```
//
// Retrieve the specified movie based on movie id:
//
app.get('/movies/:id', (req, res) => {
  try {
     console.log("**call to /movies/:id");

     id = req.params.id;

     let sql = 'select * from movies where movie_id = ?;'
     let params = [id];

     // execute the SQL and return the result (or an error):
     movielens.get(sql, params, (err, row) => {
       if (err) {
         res.status(500).json({ message: err.message, data: {} });
         return;
       }

       if (row == undefined) {  // no such movie:
         res.json({ message: "no such movie", data: {} });
         return;
       }

       res.json({ message: "success", data: row });
     });

     return;
   }
   catch(err) { res.status(500).json({ "message": err.message, "data": {} }); }
});
```

Client-side in Python

- **Call your function using Python client**
  - *Edit "client4.py"*


```
{'message': …,
 'data': …   }
```

```
import requests

baseurl = 'http://localhost:3000'  ## no / at the end

id = input('enter a movie id (e.g. 862)> ')

url = baseurl + "/movies/" + id

response = requests.get(url)

# print status_code
# if status_code == 200:
#   deserialize the response
#   print the message
#   print the data
```

```
print(response.status_code)

if response.status_code == 200:
  body = response.json()
  message = body['message']
  data = body['data']
  print(message)
  print(data)
```

That's it, thank you!