Lab 1

Due Feb 18 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip
Available after Feb 4 at 12am

CS-554 Lab 1

Reviewing API Development

For this lab, you will submit a web server with the supplied routes and middlewares.

Verb	Route	Description
GET	/api/movies	Shows a list of movies. By default, it will show the first 20 movies in the collection. If a querystring (http://expressjs.com/en/api.html#req.query) variable `?skip=n` is provided, you will skip the first `n` movies. If a querystring variable `?take=y` is provided, it will show `y` number of movies. By default, the route will show up to `20` movies; at most, it will show `100` movies.
GET	/api/movies/:id	Shows the movie with the supplied ID
POST	/api/movies	Creates a movie with the supplied detail and returns created object; fails request if not all details supplied
PUT	/api/movies/:id	Updates the movie with the supplied ID and returns the new movie object; **Note**: PUT calls must provide all details of the new state of the object! **Note:** you cannot manipulate comments in this route!
PATCH	/api/movies/:id	Updates the movie with the supplied ID and returns the new movie object; **Note**: PATCH calls only provide deltas of the value to update! **Note:** you cannot manipulate comments in this route!
POST	/api/movies/:id/comments	Adds a new comment to the movie; ids must be generated by the server, and not supplied
DELETE	/api/movies/:movieId/:commentId	Deletes the comment with an id of `commentId` on the movie with an id of `movieId`

All PUT, POST, and PATCH routes expect their content to be in JSON format, supplied in the body.

All routes will return JSON.

Logging middleware

You will write and apply three middlewares:

1. The first will keep a count of the total number of requests to the server
2. The second will log all request bodies, as well as the url path they are requesting, and the HTTP verb they are using to make the request
3. The third will keep track of many times a particular URL has been requested, updating and logging with each request.

Database

You will use a module to abstract out the database calls.

You may find it helpful to reference the **CS-546 lecture 6 code (https://github.com/stevens-cs546-cs554/CS-546/tree/master/lecture_06/code)** .

You will store all data in a database named as such: `LastName-FirstName-CS554-Lab1` .

You may name the collection however you would like.

All ids must be generated by the server and be sufficiently random!

The movie object

```
{
  "title": string,
  "cast": [{firstName: string, lastName: string}, ........],
  "info": {director: string, yearReleased: number},
  "plot": string,
  "rating": number,
  "comments": [objects]
}
```

The comment object

```
{
  _id: new ObjectID(),
  "name": string,
  "comment": string
}
```

Example movie:

```
{
  "id": "956f8670-3bb3-460f-88cd-24ca7ee9091f",
  "title": "Bill and Ted Face the Music",
  "cast": [{firstName: "Keanu", lastName:"Reeves"},{firstName: "Alex", lastName:"Winter"}],
  "info": {director: "Dean Parisot", yearReleased: 2020},
  "plot": "Once told they'd save the universe during a time-traveling adventure, 2 would-be rockers from San Di
  "rating": 4.5,
  "comments": [
    {
        "id": "d0feb7c1-d4a8-4ca6-a8a7-2ed0a0189e28",
        "name": "Patrick",
        "comment": "Most Excellent Movie!"
    },
    {
        "id": "3b4e3555-77d3-42af-ae38-03e463e95292",
        "name": "Jason",
        "comment": "Bogus.. Most Heinous"
    },
    {
        "id": "294b9ef4-255b-4869-b6d1-105eece8add8",
        "name": "Mark",
        "comment": "Put them in the iron maiden"
    },
    {
        "id": "8372caad-0ec2-477a-a496-1b64534759ec",
        "name": "Bill & Ted",
        "comment": "Excellent!!"
    },
    {
        "id": "6f7098df-fede-45b5-9a20-5e85c107ada8,
        "name": "Mark",
        "comment": "Execute them!"
    },
    {
        "id": "4db0a278-41aa-4e90-add8-2671b48cdb2b",
        "name": "Bill & Ted",
        "comment": "Bogus!"
    }
  ]
}
```


Error Checking

1. You must error check all routes
2. You must error check all DB functions
3. You must fail with proper and valid HTTP status codes depending on the failure type

4. Do not forget to check for proper datatypes in the query string parameters for skip and take (they should be positive numbers, if they are not positive numbers, you should throw an error)

Notes

1. Remember to submit your `package.json` file but **not** your `node_modules` folder.