Server Programming Project 3

Write a RESTful service in Node.js for a company to allow them to track timecards for employees. Required to use the provided Data Layer. There is a separate zip file for the Data Layer (see below on how to include it in your app). Need to create the Service as per below, including any validation mentioned which **should be in** your Business Layer. Put other things in the Business Layer if the developer wishes. **Need to use your RIT user ID for the company name whenever it is asked for**. For error output, return an appropriate error message (**not** the String "An appropriate error message.")

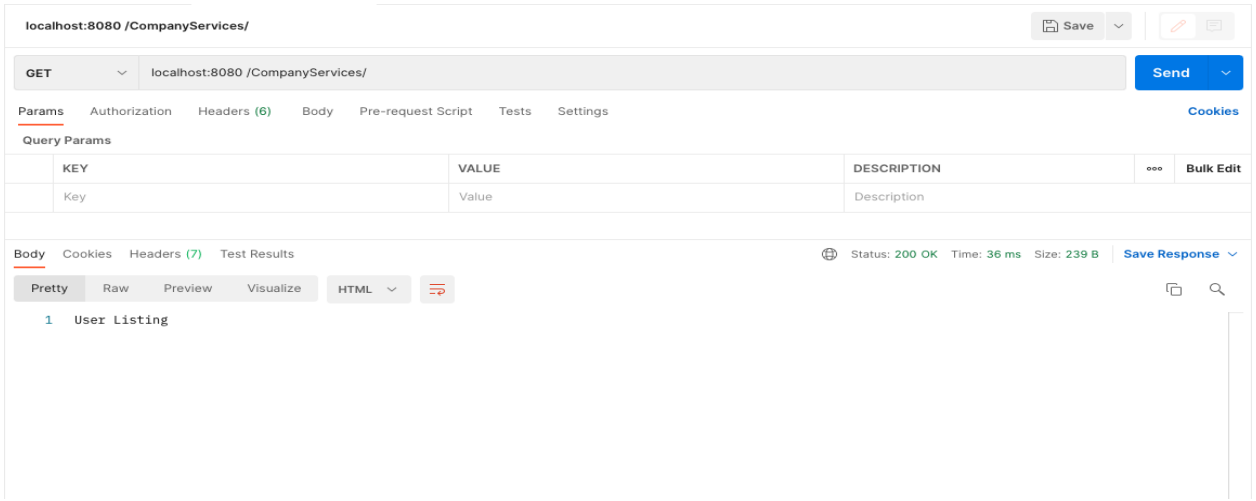
Service Layer:

All methods must return a JSON String which doesn't have to be **formatted** as in the samples below (in other words, with no carriage returns/line feeds/tabs) but must contain the same information. Some methods take JSON as input, others take Query Parameters or Form Parameters.

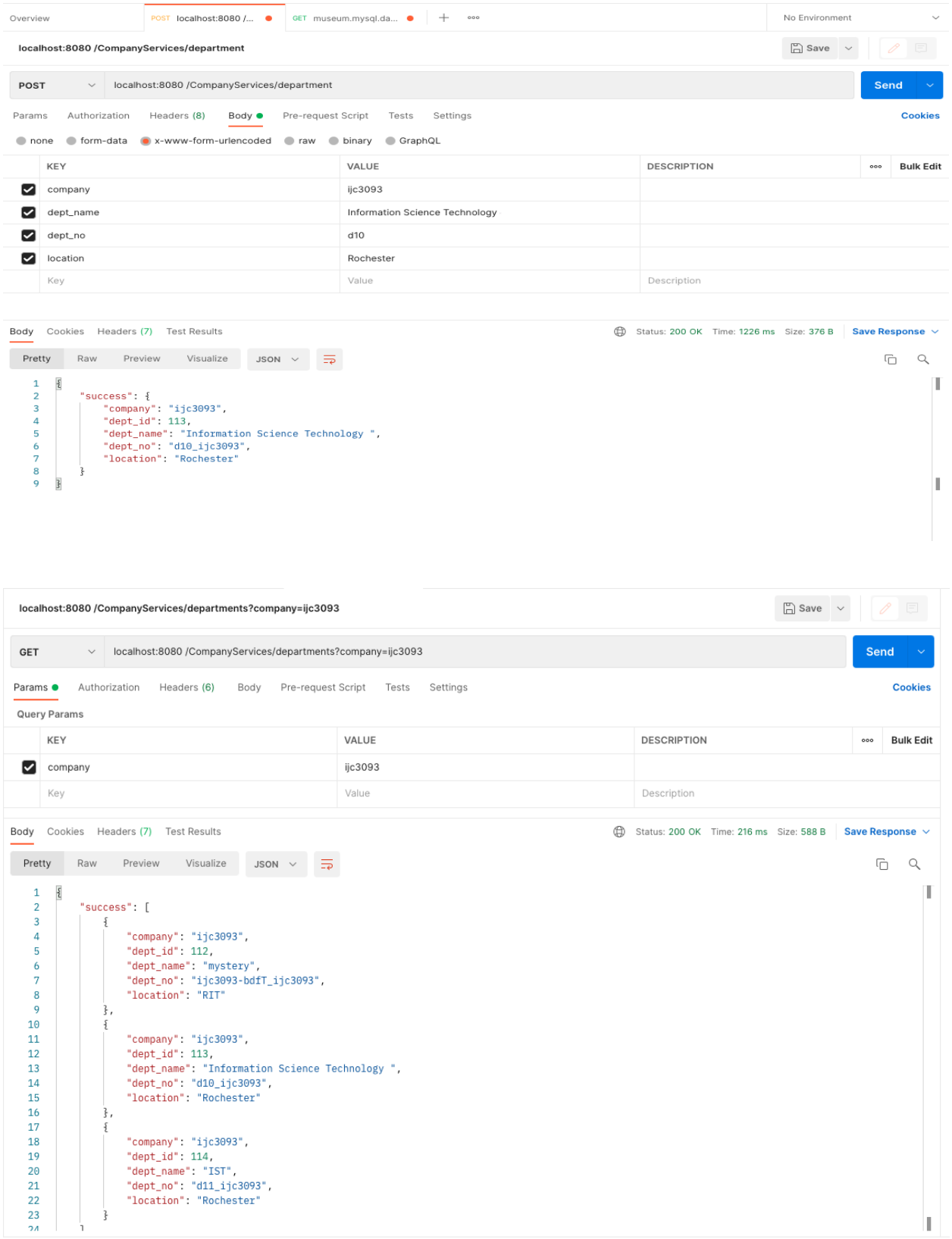
Open terminal or cmd

- Drag the **Nodejs-RestFul-Postman-IST341** into visual studio code.
- Click the terminal at the top tab and type: "node server.js"
- Your server.js app should now be running on localhost:8080.
- If you want to see all data on browse, you can see at http://localhost:8080/department
- Or open the postman and type in the url: localhost:8080 /CompanyServices/?company=ijc3093, it will be displayed: "user listening" meaning it is now running.
- Then please look at screens folder of the postman below and follow example:

COMPANY



DEPARTMENT


Overview
POST localhost:8080 /...
GET museum.mysql.da...
No Environment
localhost:8080 /CompanyServices/department
Save
POST
localhost:8080 /CompanyServices/department
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
KEY VALUE DESCRIPTION Bulk Edit
company ijc3093
dept_name Information Science Technology
dept_no d10
location Rochester
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 1226 ms Size: 376 B Save Response
Pretty Raw Preview Visualize JSON
{
"success": {
"company": "ijc3093",
"dept_id": 113,
"dept_name": "Information Science Technology ",
"dept_no": "d10_ijc3093",
"location": "Rochester"
}
}
localhost:8080 /CompanyServices/departments?company=ijc3093
Save
GET
localhost:8080 /CompanyServices/departments?company=ijc3093
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
company ijc3093
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 216 ms Size: 588 B Save Response
Pretty Raw Preview Visualize JSON
{
"success": [
{
"company": "ijc3093",
"dept_id": 112,
"dept_name": "mystery",
"dept_no": "ijc3093-bdfT_ijc3093",
"location": "RIT"
},
{
"company": "ijc3093",
"dept_id": 113,
"dept_name": "Information Science Technology ",
"dept_no": "d10_ijc3093",
"location": "Rochester"
},
{
"company": "ijc3093",
"dept_id": 114,
"dept_name": "IST",
"dept_no": "d11_ijc3093",
"location": "Rochester"
}
]

GET localhost:8080 /CompanyServices/department?company=ijc3093&dept_id=112 Send

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

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
company	ijc3093		
dept_id	112		
Key	Value	Description	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 215 ms Size: 355 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "success": {
        "company": "ijc3093",
        "dept_id": 112,
        "dept_name": "mystery",
        "dept_no": "ijc3093-bdfT_ijc3093",
        "location": "RIT"
    }
}
```


localhost:8080 /CompanyServices/department Save

PUT localhost:8080 /CompanyServices/department Send

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

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

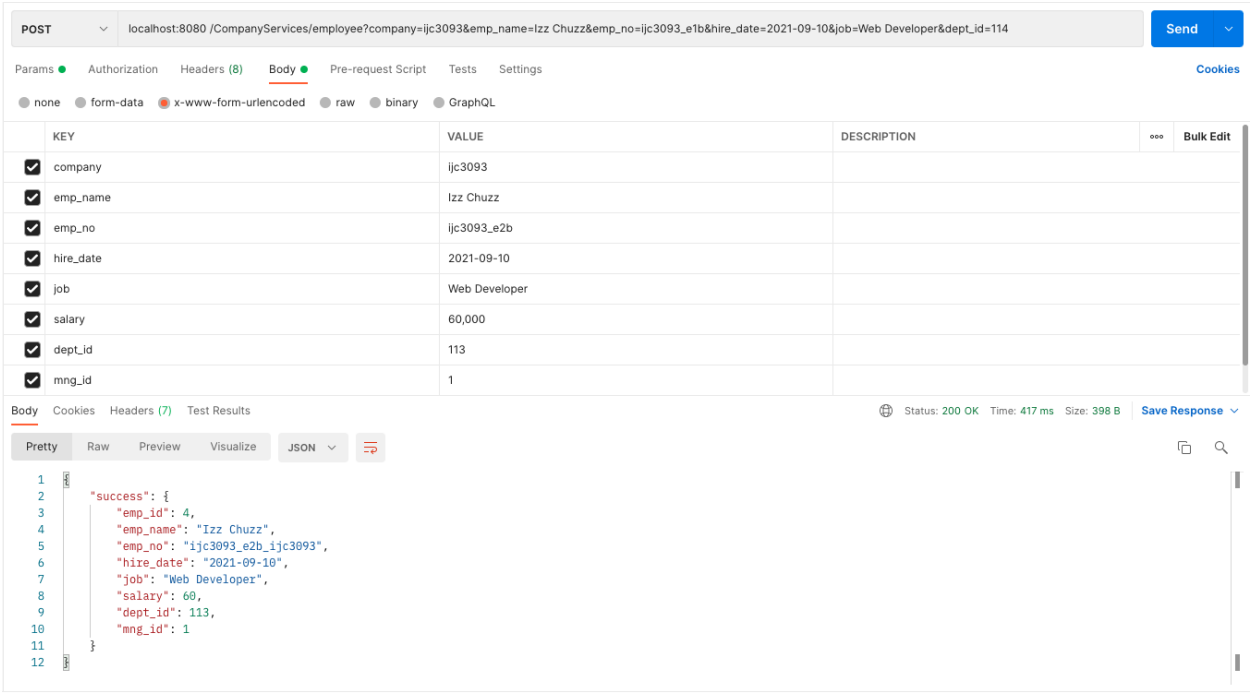

```
{
    "company": "ijc3093",
    "dept_id": 114,
    "dept_name": "Information Science Technology",
    "dept_no": "d11_ijc3093",
    "location": "Rochester"
}
```

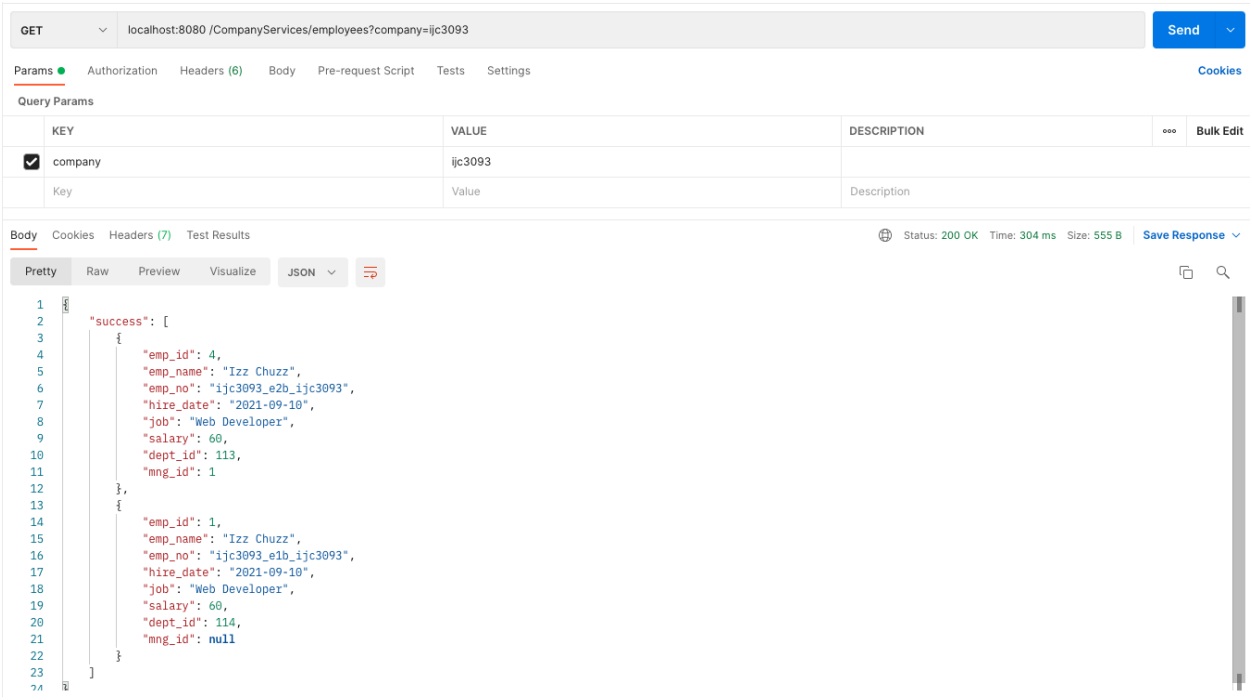

Body Cookies Headers (7) Test Results Status: 200 OK Time: 504 ms Size: 383 B Save Response

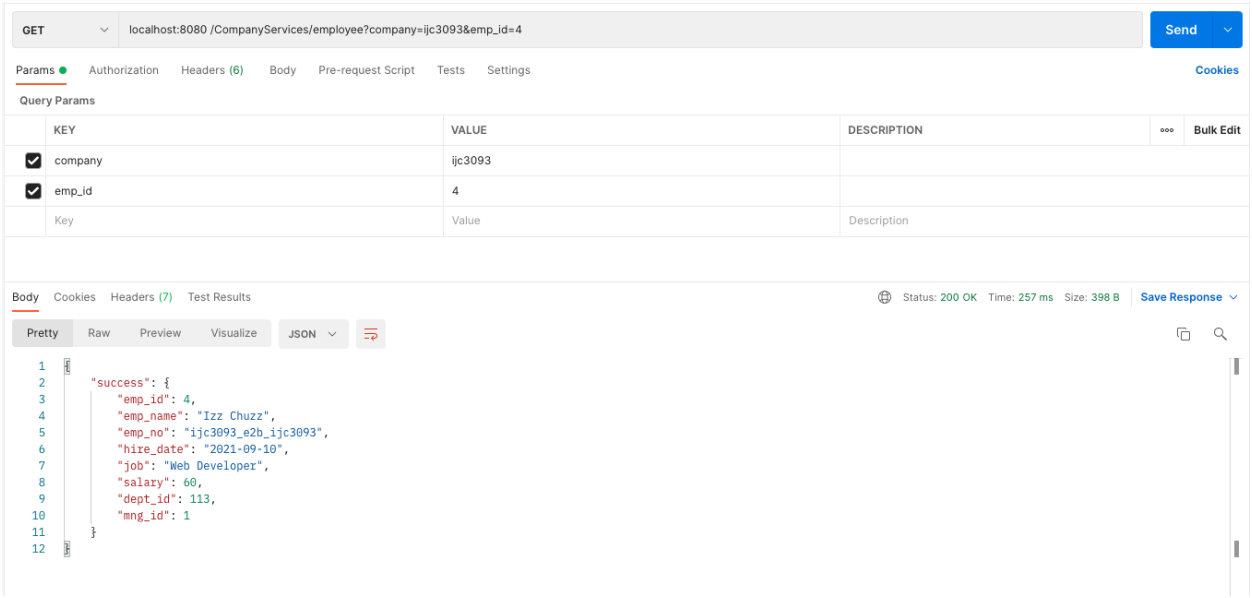
Pretty Raw Preview Visualize JSON

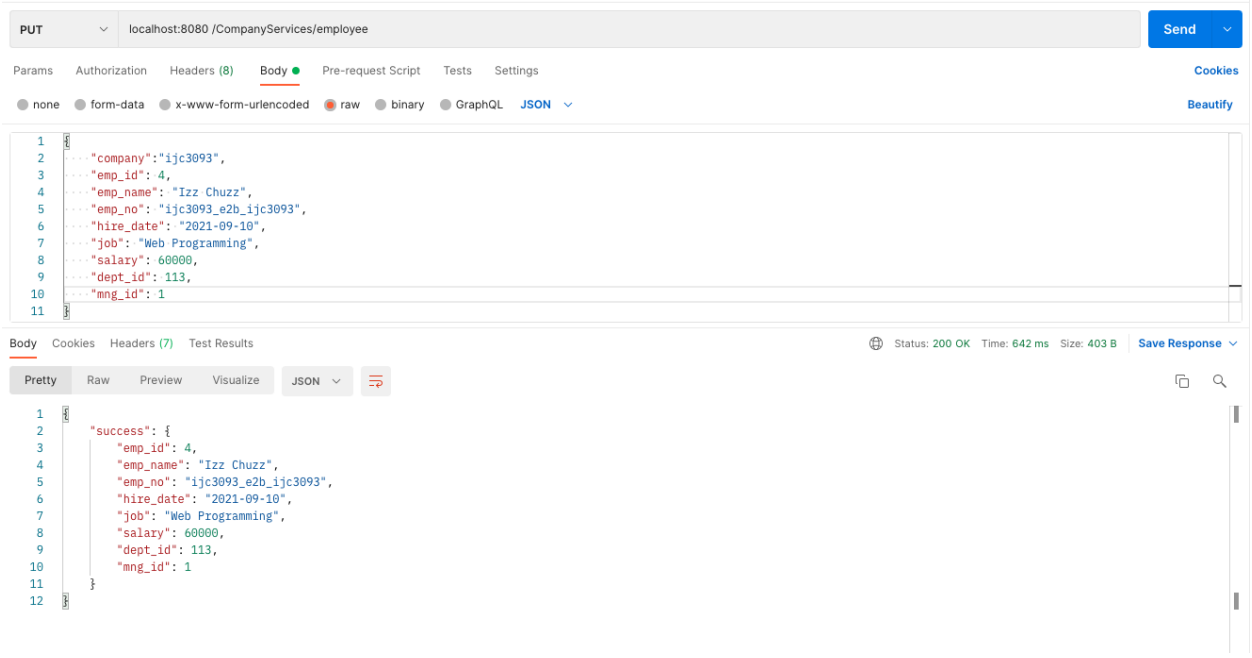

```
{
    "success": {
        "company": "ijc3093",
        "dept_id": 114,
        "dept_name": "Information Science Technology",
        "dept_no": "d11_ijc3093-ijc3093",
        "location": "Rochester"
    }
}
```

EMPLOYEE

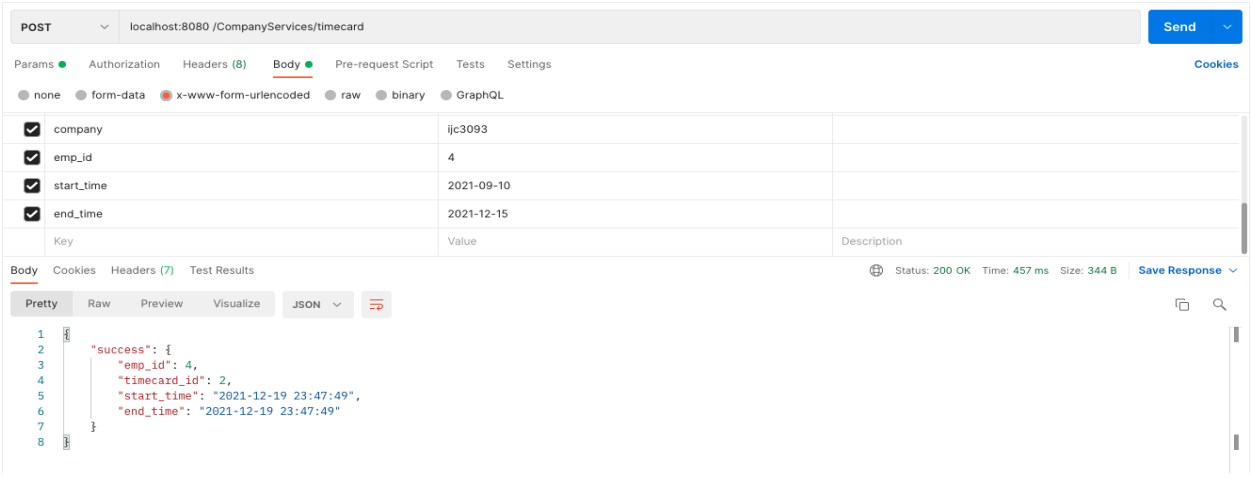

POST
localhost:8080 /CompanyServices/employee?company=ijc3093&emp_name=Izz Chuzz&emp_no=ijc3093_e1b&hire_date=2021-09-10&job=Web Developer&dept_id=114
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
company
ijc3093
emp_name
Izz Chuzz
emp_no
ijc3093_e2b
hire_date
2021-09-10
job
Web Developer
salary
60,000
dept_id
113
mng_id
1
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 417 ms
Size: 398 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": {
"emp_id": 4,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e2b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Developer",
"salary": 60,
"dept_id": 113,
"mng_id": 1
}
}

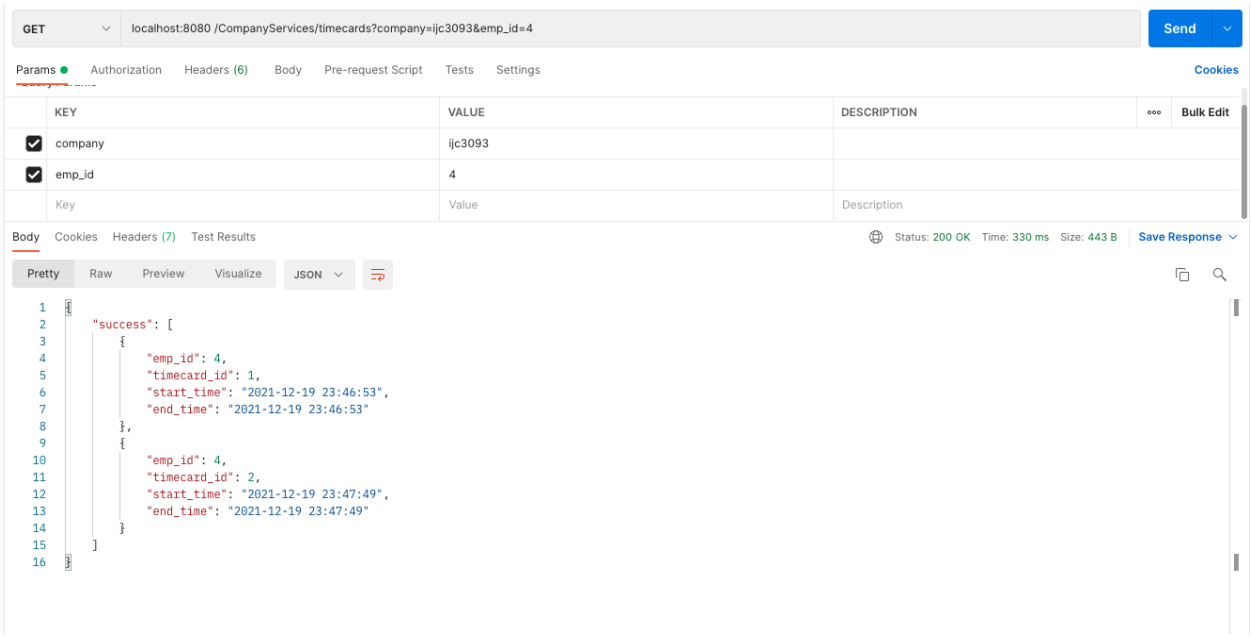

GET
localhost:8080 /CompanyServices/employees?company=ijc3093
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
company
ijc3093
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 304 ms
Size: 555 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": [
{
"emp_id": 4,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e2b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Developer",
"salary": 60,
"dept_id": 113,
"mng_id": 1
},
{
"emp_id": 1,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e1b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Developer",
"salary": 60,
"dept_id": 114,
"mng_id": null
}
]
}

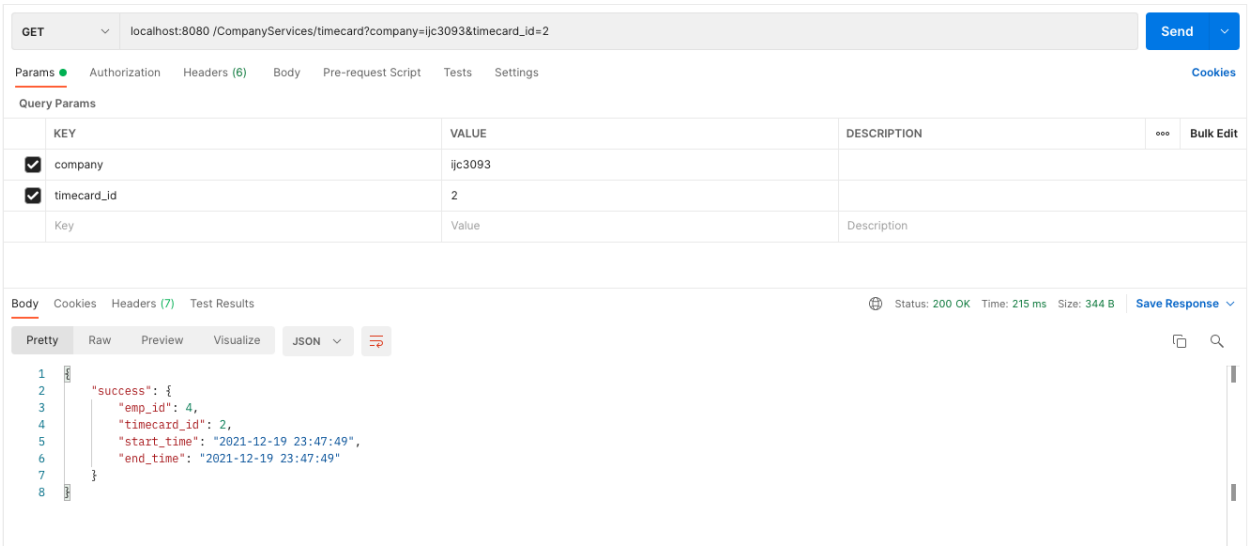

GET
localhost:8080 /CompanyServices/employee?company=ijc3093&emp_id=4
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
company
ijc3093
emp_id
4
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 257 ms
Size: 398 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": {
"emp_id": 4,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e2b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Developer",
"salary": 60,
"dept_id": 113,
"mng_id": 1
}
}


PUT
localhost:8080 /CompanyServices/employee
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"company":"ijc3093",
"emp_id": 4,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e2b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Programming",
"salary": 60000,
"dept_id": 113,
"mng_id": 1
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 642 ms
Size: 403 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": {
"emp_id": 4,
"emp_name": "Izz Chuzz",
"emp_no": "ijc3093_e2b_ijc3093",
"hire_date": "2021-09-10",
"job": "Web Programming",
"salary": 60000,
"dept_id": 113,
"mng_id": 1
}
}

TIMECARD


POST
localhost:8080 /CompanyServices/timecard
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
company
ijc3093
emp_id
4
start_time
2021-09-10
end_time
2021-12-15
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 457 ms
Size: 344 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": {
"emp_id": 4,
"timecard_id": 2,
"start_time": "2021-12-19 23:47:49",
"end_time": "2021-12-19 23:47:49"
}
}


GET
localhost:8080 /CompanyServices/timecards?company=ijc3093&emp_id=4
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
KEY
VALUE
DESCRIPTION
Bulk Edit
company
ijc3093
emp_id
4
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 330 ms
Size: 443 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"success": [
{
"emp_id": 4,
"timecard_id": 1,
"start_time": "2021-12-19 23:46:53",
"end_time": "2021-12-19 23:46:53"
},
{
"emp_id": 4,
"timecard_id": 2,
"start_time": "2021-12-19 23:47:49",
"end_time": "2021-12-19 23:47:49"
}
]
}



All method signatures must match the ones listed.

You may have to use multiple Data Layer methods to accomplish each Service Layer method.

General input validation: Refer to the EER Diagram for the database for datatypes and sizes. Any additional validation/business rules will be listed below in the appropriate method.

1) Root Path for Service Layer: "CompanyServices"

2) Server should listen on port 8080

3) The remaining paths will be appended to the above, e.g. localhost:8080 /CompanyServices/

4) Path: /company

Verb: DELETE

Produces: application/json

a. Deletes all Department, Employee and Timecard records in the database for the given company. You will need to pay attention to the Foreign Key Constraints.

b. Input is your RIT user ID as a String passed as **QueryParam**

i. company=company+name

where "company+name" is your RIT user ID

c. Output:

i. Success:

1. A JSON String:

{"success":"companyName's information deleted."}

2. A JSON String:

{"error":"An appropriate error message."}

5) Path: /department
Verb: GET
Produces: application/json

a. Returns the requested Department as a JSON String.
b. Input as **QueryParams**:

company=company+name&dept_id= id

where “company+name” is your RIT user ID and
“dept_id” is the record id of the department to retrieve.

c. Output:

i. Success:

1. A JSON String:


```
{
    "dept_id":1,
    "company":"rituserid",
  "dept_name":"accounting",
    "dept_no":"d10",
    "location":"new york"
}
```


2. A JSON String:

{"error":"An appropriate error message."}

6) Path: /departments
Verb: GET
Produces: application/json

a. Returns the requested list of Departments.

b. Input is your RIT user ID as a String in a **QueryParam**.

i. company=company+name

where “company+name” is your RIT user ID

c. Output:

i. Success:

1. A JSON String:


```
[
  {
      "dept_id":1,
      "company":"rituserid",
      "dept_name":"accounting",
      "dept_no":"d10",
      "location":"new York”
  },
  {
      "dept_id":2,
      "company":"rituserid",
      "dept_name":"research",
      "dept_no":"d20",
      "location":"dallas"
  },
  {
      "dept_id":3,
      "company":"rituserid",
      "dept_name":"sales",
      "dept_no":"d30",
      "location":"chicago"   },
  {
      "dept_id":4,
      "company":"rituserid",
      "dept_name":"operations",
      "dept_no":"d40",
      "location":"boston"   }
]
```

2. A JSON String:

{"error":"An appropriate error message."}

7) Path: /department
Verb: PUT
Consumes: application/json
Produces: application/json

a. Additional Validation:
   i. dept_no must be unique among all companies, Suggestion: include company name as part of id.
   ii. dept_id must be an existing record number for a department

b. Returns the updated Department as a JSON String.
c. Input: **JSON String (**Input any values you want to change plus the record id for the Department)

```
{
  "company":"rituserid",
  "dept_id":5,
  "dept_name":"IT",
  "dept_no":"d11",
  "location":"rochester"
}
```

where “company” is your RIT user ID.

d. Output:
   i. Success:
      1. A JSON String:

```
{
  "success":{
      "dept_id":5,
```

"company":"rituserid",
"dept_name":"IT",
"dept_no":"d11",
"location":"rochester" }
}

2. A JSON String:

{"error":"An appropriate error message."}

8) Path: /department
Verb: POST
Produces: application/json

a. Additional Validation:
i. dept_no must be unique among all companies, Suggestion: include company name as part of id.
b. Returns the new Department as a JSON String.
c. Input as FormParam:

"company" ="rituserid"
"dept_name" = "mystery"
"dept_no" = "d10"
"location" = "buffalo"

where “company” is your RIT user ID.

c. Output:
i. Success:
1. A JSON String:

{
"success":{
"dept_id":1,
"company":"rituserid",
"dept_name":"mystery",

```
    "dept_no":"d10",
    "location":"buffalo"   }
}
```

2. A JSON String:

{"error":"An appropriate error message."}

9) Path: /department
Verb: DELETE
Produces: application/json

a. Returns the number of rows deleted.
b. Input as **QueryParam**:

"company" = "company name"
"dept_id" = id

where “company name” is your RIT user ID and
“id” is the record id of the department to delete.

c. Output:

i. Success:

1. A JSON String:

```
{
  "success":"Department 5 from rituserid deleted."
}
```

2. A JSON String:

{"error":"An appropriate error message."}

10) Path: /employee
Verb: GET
Produces: application/json

a. Returns the requested Employee as a JSON String.
b. Input: the record id of the desired Employee as a **QueryParam**
    i. company=company+name
    where “company+name” is your RIT user ID
    ii. emp_id=#
c. Output:
    i. Success:
    1. A JSON String:

```
{
    "emp_id":2,
    "emp_name":"jones",
    "emp_no":"e2",
    "hire_date":"1981-04-01",
    "job":"manager",
    "salary":2975.0,
    "dept_id":2,
    "mng_id":1}
```

    2. A JSON String:

```
{"error":"An appropriate error message."}
```

11) Path: /employees
Verb: GET
Produces: application/json

a. Returns the requested list of Employees.
b. Input is your RIT user ID as a String as a **QueryParam**.
    i. company=company+name
    where “company+name” is your RIT user ID
c. Output:
    i. Success:
    1. A JSON String:

```
[
  {
```

```
    "emp_id":1,
    "emp_name":"king",
    "emp_no":"e1",
    "hire_date":"1981-11-16",
    "job":"president",
    "salary":5000.0,
    "dept_id":1,
    "mng_id":0
},
{
    "emp_id":2,
    "emp_name":"jones",
    "emp_no":"e2",
    "hire_date":"1981-04-01",
    "job":"manager",
    "salary":2975.0,
    "dept_id":2,
    "mng_id":1
},
{
    "emp_id":3,
    "emp_name":"ford",
    "emp_no":"e3",
    "hire_date":"1981-12-02",
    "job":"analyst",
    "salary":3000.0,
    "dept_id":2,
    "mng_id":2
},
{
    "emp_id":4,
    "emp_name":"smith",
    "emp_no":"e4",
    "hire_date":"1980-12-16",
    "job":"clerk",
    "salary":800.0,
    "dept_id":2,
    "mng_id":2
```

```
},
{
    "emp_id":5,
    "emp_name":"blake",
    "emp_no":"e5",
    "hire_date":"1981-04-30",
    "job":"manager",
    "salary":2850.0,
    "dept_id":3,
    "mng_id":1   },
{
    "emp_id":6,
    "emp_name":"allen",
    "emp_no":"e6",
    "hire_date":"1981-02-19",
    "job":"salesman",
    "salary":1600.0,
    "dept_id":3,
    "mng_id":5   },
{
    "emp_id":7,
    "emp_name":"ward",
    "emp_no":"e7",
    "hire_date":"1981-02-21",
    "job":"salesman",
    "salary":1250.0,
    "dept_id":3,
    "mng_id":5
},
{
    "emp_id":8,
    "emp_name":"martin",
    "emp_no":"e8",
    "hire_date":"1981-09-27",
    "job":"salesman",
    "salary":1250.0,
    "dept_id":3,
    "mng_id":5
```

```
  },
  {
      "emp_id":9,
      "emp_name":"clark",
      "emp_no":"e9",
      "hire_date":"1981-06-08",
      "job":"manager",
      "salary":2450.0,
      "dept_id":3,
      "mng_id":1   }
]
```


2. A JSON String:


```
{"error":"An appropriate error message."}
```


12) Path: /employee
Verb: POST
Consumes: Form Parameters
Produces: application/json

a. Additional validations:
   i. company – must be your RIT username
   ii. dept_id must exist as a Department in your company
   iii. mng_id must be the record id of an existing Employee in your company. Use 0 if the first employee or any other employee that doesn't have a manager.
   iv. hire_date must be a valid date equal to the current date or earlier (e.g. current date or in the past)
   v. hire_date must be a Monday, Tuesday, Wednesday, Thursday or a Friday. It **cannot** be Saturday or Sunday.
   vi. emp_no must be unique amongst all employees in the database, **including** those of other companies. You may wish to include your RIT user ID in the employee number somehow.
b. Returns the new Employee as a JSON String.

c. Input as FormParam:

“company”=”yourRITid”,
"emp_name"="french",
"emp_no"="rituserid-e1b",
"hire_date"="2018-06-16",
"job"="programmer",
"salary"=5000.0,
"dept_id"=1,
"mng_id"=2

d. Output:

i. Success:

1. A JSON String:

```
{
  "success":{
      "emp_id":15,
      "emp_name":"french",
      "emp_no":"rituserid-e1b",
      "hire_date":"2018-06-16",
      "job":"programmer",
      "salary":5000.0,
      "dept_id":1,
      "mng_id":2   }
}
```

2. A JSON String:

```
{"error":"An appropriate error message."}
```

13) Path: /employee
Verb: PUT
Consumes: application/json
Produces: application/json

a. Additional validations same as inserting an Employee plus emp_id must be a valid record id in the database.

b. Returns the updated Employee as a JSON String.

c. Input(any values you want to change plus the record id for the Employee) as **JSON string (company+name is your RIT username)**:

```
{
  "company ":company+name,
  "emp_id":15,
  "emp_name":"french",
  "emp_no":"rituserid-e1b",
  "hire_date":"2018-06-16",
  "job":"programmer",
  "salary":6000.0,
  "dept_id":1,
  "mng_id":2
}
```

d. Output:

i. Success:

1. A JSON String:

```
{
  "success":{
      "emp_id":15,
      "emp_name":"french",
      "emp_no":"rituserid-e1b",
      "hire_date":"2018-06-16",
      "job":"programmer",
      "salary":6000.0,
      "dept_id":1,
      "mng_id":2   }
}
```

2. A JSON String:

```
{"error":"An appropriate error message."}
```

14) Path: /employee
Verb: DELETE
Produces: application/json

a. Returns the that the employee deleted.
b. Input: the record id of the Employee to delete as a **QueryParam**.
    i. company=company+name
    where “company+name” is your RIT user ID
    ii. emp_id=#

c. Output:
    i. Success:
    1. A JSON String:

```
{
  "success":"Employee 15 deleted."
}
```

    2. A JSON String:

```
{"error":"An appropriate error message."}
```

15) Path: /timecard
Verb: GET
Produces: application/json

a. Returns the requested Timecard as a JSON String.
b. Input: the record id of the desired Timecard as a **QueryParam**
    i. company=company+name
    where “company+name” is your RIT user ID
    ii. timecard_id=#

c. Output:
    i. Success:
    1. A JSON String:

```
{
  "timecard":{
    "timecard_id":1,
    "start_time":"2018-06-14 11:30:00",
    "end_time":"2018-06-14 15:30:00",
    "emp_id":2
  }
}
```


2. A JSON String:


```
{"error":"An appropriate error message."}
```


16) Path: /timecards
Verb: GET
Produces: application/json

a. Returns the requested list of Timecards.
b. Input is the record id of the employee you want to see the Timecards for as a **QueryParam**.
    i. company=company+name
where “company+name” is your RIT user ID
    ii. emp_id=#
c. Output:
    i. Success:
1. A JSON String:


```
[
  {
    "timecard_id":3,
    "start_time":"2018-06-14 11:30:00",
    "end_time":"2018-06-14 15:30:00",
    "emp_id":4
  },
  {
    "timecard_id":4,
    "start_time":"2018-06-13 11:30:00",
```

```
        "end_time":"2018-06-13 15:30:00",
        "emp_id":4
    },
    {
        "timecard_id":6,
        "start_time":"2018-06-12 11:30:00",
        "end_time":"2018-06-12 15:30:00",
        "emp_id":4   }
]
```


2. A JSON String:


```
{"error":"An appropriate error message."}
```


17) Path: /timecard
Verb: POST
Consumes: form parameters
Produces: application/json

a. Additional validations:
   i. company must be your RIT id
   ii. emp_id must exist as the record id of an Employee in your company.
   iii. start_time must be a valid date and time equal to the current date or back to the Monday prior to the current date if the current date is not a Monday.
   iv. end_time must be a valid date and time at least 1 hour greater than the start_time and be on the same day as the start_time.
   v. start_time and end_time must be a Monday, Tuesday, Wednesday, Thursday or a Friday. They **cannot** be Saturday or Sunday.
   vi. start_time and end_time must be between the hours (in 24 hour format) of 08:00:00 and 18:00:00 inclusive.
   vii. start_time must not be on the same day as any other start_time for that employee.

b. Returns the new Timecard as a JSON String.

c. Input all Timecard values as **FormParam**s:

“company=”your RIT ID”,
”"emp_id"=1,
"start_time"="2018-06-15 12:30:00",
"end_time"="2018-06-15 15:30:00"

d. Output:

i. Success:

1. A JSON String:

```
{
  "success":{
     "timecard_id":1,
     "start_time":"2018-06-14 11:30:00",
     "end_time":"2018-06-14 15:30:00",
     "emp_id":2
  }
}
```

2. A JSON String:

{"error":"An appropriate error message."}

18) Path: /timecard
Verb: PUT
Consumes: application/json
Produces: application/json

a. Additional validations same as inserting a Timecard plus timecard_id must be a valid record id in the database.

b. Returns the updated Timecard as a JSON String.

c. Input(any values you want to change plus the record id for the Timecard) as **JSON string (company is your RIT username)**:

```
{
  “company”:”your RIT ID",
        timecard_id":2,
  "start_time":"2018-06-14 11:30:00",
  "end_time":"2018-06-14 15:30:00",
  "emp_id":1
}
```

d. Output:

i. Success:

1. A JSON String:

```
{
  "success":{
      "timecard_id":0,
      "start_time":"2018-06-15 12:30:00",
      "end_time":"2018-06-15 15:30:00",
      "emp_id":2

  }
}
```

2. A JSON String:

```
{"error":"An appropriate error message."}
```

19) Path: /timecard
Verb: DELETE
Produces: application/json

a. Returns the number of rows deleted.
b. Input: the record id of the Timecard to delete as a **QueryParam**.

i. company=company+name

where “company+name” is your RIT user ID

ii. timecard_id=#

c. Output:

i. Success:

1. A JSON String:

```
{
  "success":"Timecard 1 deleted."
}
```

2. A JSON String:

```
{"error":"An appropriate error message."}
```

Deliverables:

Put the following in the dropbox for Project 3 by the due date on the dropbox:

1) A zip file of all of your source files. (don't zip up your node_modules folder)

Hints:

1) In addition to the Data Layer, use npm install –save when installing.

2) To convert from Timestamp to String (for putting in JSON String), take a look at the date-fns module (format method) or the moment.js module.

3) When creating a Timecard, use the string representation of the date as a parameter to the constructor in the formt of yyyy-mm-dd hh:mm:ss

4) Use: app.use(express.json()) for processing JSON body input to get the fields from req.body

5) Use: app.use(express.urlencoded({extended:false})) for processing POST form input to get the fields from req.body

6) App structure for using the provided Data Layer:

- Unzip the companydata.zip file
- Make a directory for your node project

- "touch server.js"
- Copy the companydata folder created by the unzip above into this folder
- npm init
- npm install --save <path to the unzipped company folder>
- in server.js add:
  - var DataLayer = require("./companydata/index.js");
  - var dl = new DataLayer("yourusername");
  - Continue with the rest of your code.
  - To create a Department/Employee/Timecard: new dl.Department(…), new dl.Employee(…), new Timecard(…)

7) To test using Postman:

a. For DELETE method, make sure any text under raw/body is deleted, all form fields are unchecked and uncheck any header fields and the correct id is set as a parameter.

b. For POST method, select x-www-form-urlencoded with fields for each item you want to pass in.

c. For PUT, make sure Content-Type header = application/json

Rubric:

	Possible Points	Actual Points
All required methods with correct inputs and outputs:	40	
All validations in Business Layer:	25	
Appropriate error messages:	5	
Correct Node.js structure and it runs:	15	

Good code structure (DRY, etc):	15	
Total:	100	