

The background of the slide features a series of concentric, overlapping circles in a light gray color, creating a subtle, abstract pattern against the dark gray background.

▼ REST API, JSON ,
Postman

Objectives

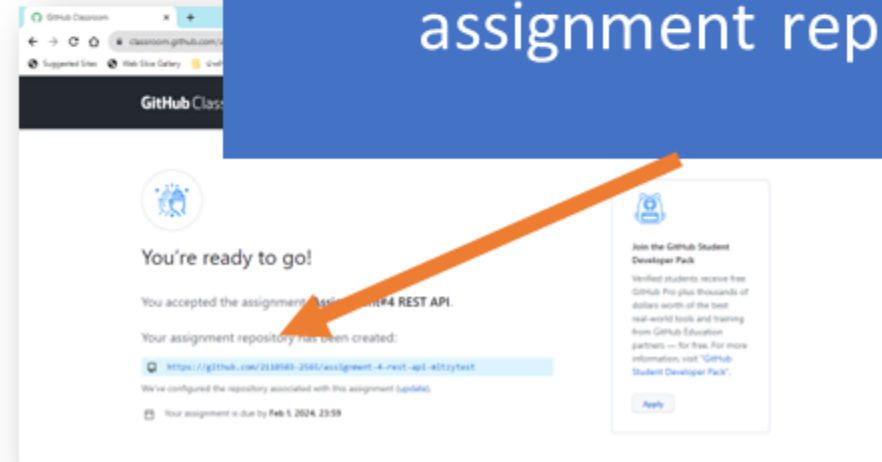
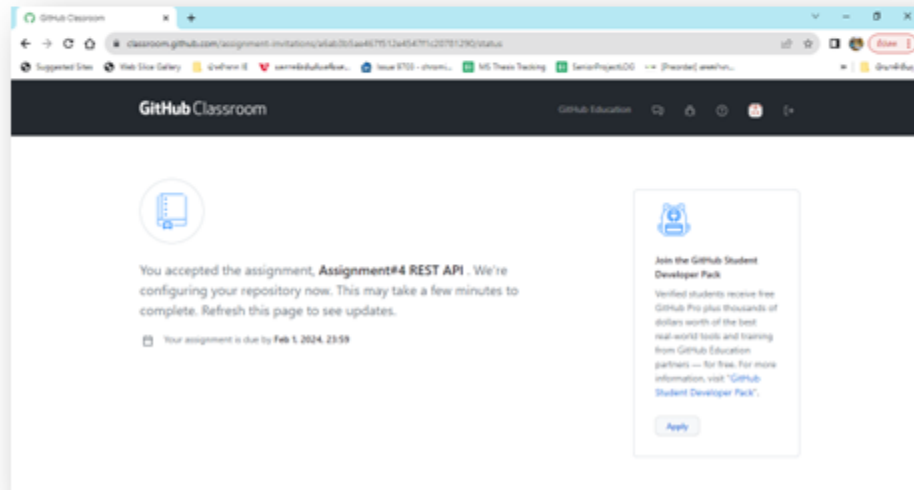
Develop basic REST APIs with Node.JS and MongoDB Atlas (on cloud)

Using Postman to test the APIs

Setup your assignment Github Repository

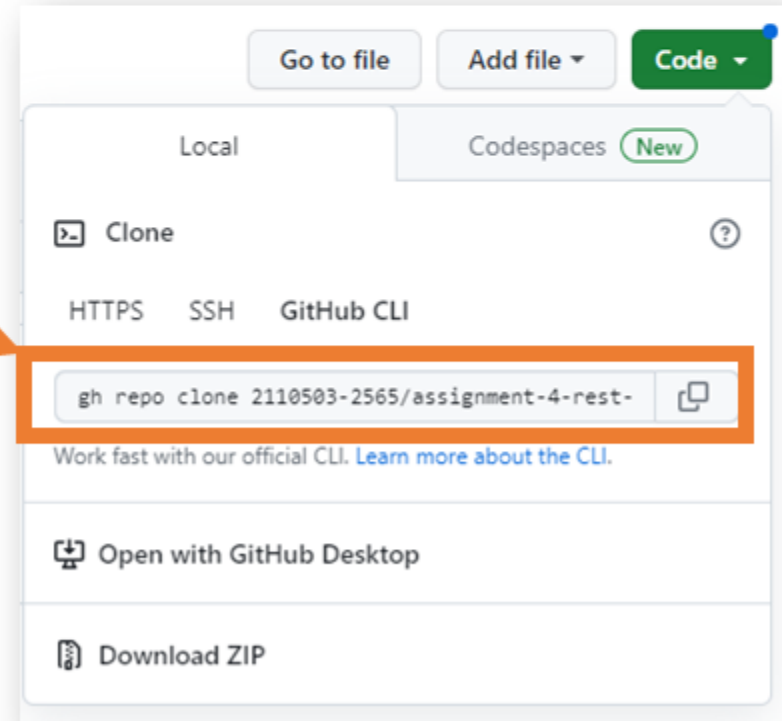


1. Goto your assignment link in MCV instruction:
 1. <https://classroom.github.com/a/XXXXXXX>
2. Log in to your Github account (or create one, if you don't have one).
3. Click on **Accept this assignment** and choose your ID.
4. Wait a minute and refresh this page.
5. Click the link to go to your assignment repo.



Setup the repository in your computer!

1. Open your destination folder in VS Code, i.e. C:\SW Practice\
 1. Tab: File >> Open folder >> Browse to your destination folder
2. Open Terminal
 1. Tab: Terminal >> New Terminal
3. Clone your github repo to your folder.
 - `winget install --id GitHub.cli` **OR** `brew install gh`
 - `gh extension install github/gh-classroom`
 - `gh repo clone 2110503-256X/your-assignment-folder`
4. change folder to your github assignment folder.
`cd your-assignment-folder`
5. Notice that there is a .git folder inside.



Install Github CLI

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

powershell

```
PS C:\autograde A6> winget install --id github.cli
```

```
Found GitHub CLI [GitHub.cli] Version 2.35.0
```

```
This application is licensed to you by its owner.
```

```
Downloading https://github.com/cli/cli/releases/download/v2.35.0/gh\_2.35.0\_windows\_amd64.msi
```

```
10.4 MB / 10.4 MB
```

```
Successfully verified installer hash
```

```
Starting package install...
```

```
Successfully installed
```

```
PS C:\autograde A6> gh extension install github/gh-classroom
```

```
gh : The term 'gh' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the  
or if a path was included, verify that the path is correct and try again.
```

```
At line:1 char:1
```

```
+ gh extension install github/gh-classroom
```

```
+ ~~
```

```
+ CategoryInfo          : ObjectNotFound: (gh:String) [], CommandNotFoundException
```

```
+ FullyQualifiedErrorId : CommandNotFoundException
```

Go Live Prettier

Install github classroom

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
```

⊗ PS C:\autograde A6> **gh** extension install github/gh-classroom
To get started with GitHub CLI, please run: **gh** auth login

⊗ Alternatively, populate the GH_TOKEN environment variable with a GitHub API authentication token.
PS C:\autograde A6> **gh** auth login

● ? What account do you want to log into? **GitHub.com**
? What is your preferred protocol for Git operations? **HTTPS**
? Authenticate Git with your GitHub credentials? **Yes**
? How would you like to authenticate GitHub CLI? **Login with a web browser**

! First copy your one-time code: **9280-3CB8**
Press Enter to open github.com in your browser...

✓ Authentication complete.
- **gh** config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as **nuengwong**

○ PS C:\autograde A6> █

Initialize your new project

```
> npm init
```

```
//entry point: server.js
```

```
> npm i express dotenv
```

```
> npm i -D nodemon
```

```
//to run the application in development mode
```


```
> npm run dev
```

```
//to run the application in production mode
```

```
> npm start
```


JS server.js U

{ } package.json M X

{ } package.json > { } scripts >  dev

```
1  {
2    "name": "vacq",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "dev": "nodemon server.js"
9    },
10   "author": "ohm",
11   "license": "MIT",
12   "dependencies": {
13     "dotenv": "^10.0.0",
14     "express": "^4.17.1"
15   },
16   "devDependencies": {
17     "nodemon": "^2.0.12"
18   }
19 }
20
```

Add this line to
package.json

.gitignore

```
# ignore all node_modules
node_modules/*
# ignore config file
config/config.env
```

config/config.env

```
PORT = 5000  
NODE_ENV = development
```

server.js: test a route

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

//Load env vars
dotenv.config({path: './config/config.env'});

const app=express();

app.get('/', (req,res) => {
  //1. res.send('<h1>Hello from express</h1>');
  //2. res.send({name:'Brad'});
  //3. res.json({name:'Brad'});
  //4. res.sendStatus(400);
  //5. res.status(400).json({success:false});
  res.status(200).json({success:true, data:{id:1}});
});

const PORT=process.env.PORT || 5000;
app.listen(PORT, console.log('Server running in ', process.env.NODE_ENV, ' mode on port ', PORT));
```

GET localhost:5000/ Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body 200 OK 16 ms 255 B Save Response

Pretty Raw Preview Visualize HTML

```
1 <h1>Hello from express</h1>
```

```
> npm run dev
```

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'Reports', and 'Explore'. The left sidebar shows 'My Workspace' with a tree view containing 'DevCamper API' and 'VacQ'. The main panel displays a GET request to 'localhost:5000/'. The response is a JSON object: `{ "success": true, "data": { "id": 1 } }`. The status bar at the bottom shows '200 OK', '13 ms', and '267 B'.

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

My Workspace New Import

VacQ GET Test google GET localhost:5...

No Environment

Collections

DevCamper API

VacQ

GET Test google

localhost:5000/

GET localhost:5000/

Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
-----	-------	-------------	-----------

Body

200 OK 13 ms 267 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "data": {
4     "id": 1
5   }
6 }
```

Find and Replace Console

Bootcamp Runner Trash

server.js: test a route

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

//Load env vars
dotenv.config({path: './config/config.env'});

const app=express();

app.get('/', (req,res) => {
  //1. res.send('<h1>Hello from express</h1>');
  //2. res.send({name:'Brad'});
  //3. res.json({name:'Brad'});
  //4. res.sendStatus(400);
  //5. res.status(400).json({success:false});
  res.status(200).json({success:true, data:{id:1}});
});

const PORT=process.env.PORT || 5000;
app.listen(PORT, console.log('Server running in ', process.env.NODE_ENV, ' mode on port ', PORT));
```

GET localhost:5000/ [Send](#)

Params Auth Headers (6) Body Pre-req. Tests Settings [Cookies](#)

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body [200 OK](#) [16 ms](#) [255 B](#) [Save Response](#)

Pretty Raw Preview Visualize [HTML](#)

```
1 <h1>Hello from express</h1>
```

Commit to your Git

```
git add .  
git commit -m 'Initial Express setup'
```

Route Structure

GET-POST-PUT-DELETE

/api/v1/hospitals

/api/v1/bookings

/api/v1/auth

/api/v1/users

server.js : add all routes (1/3)

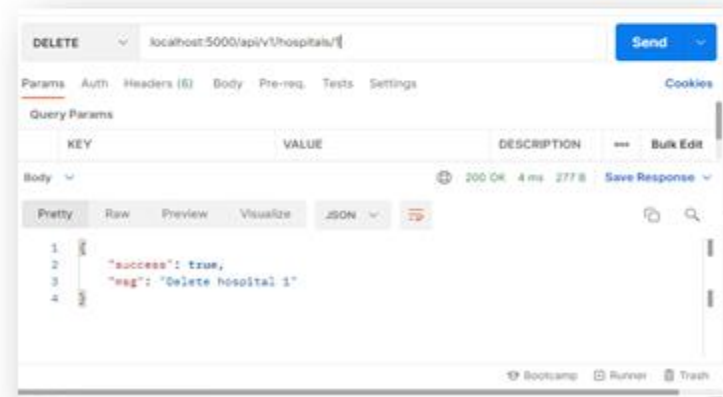
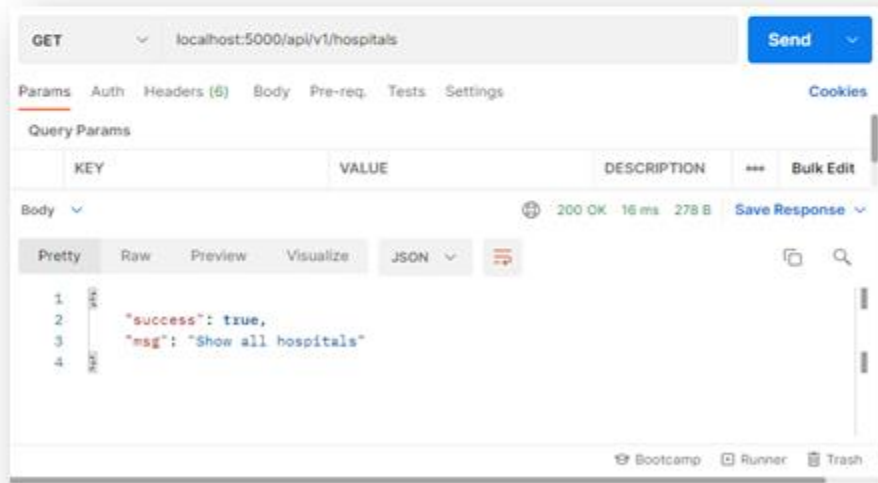
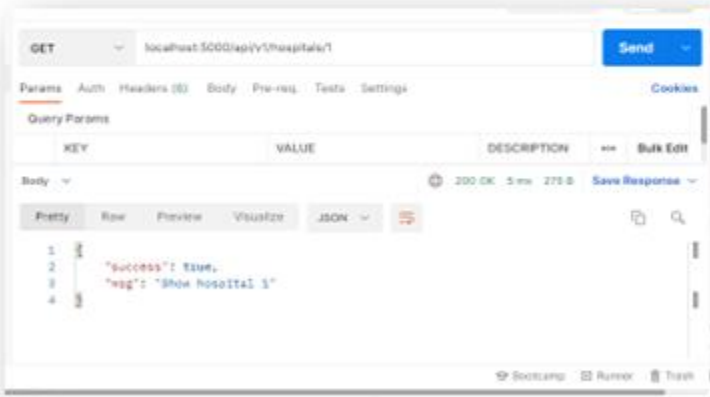
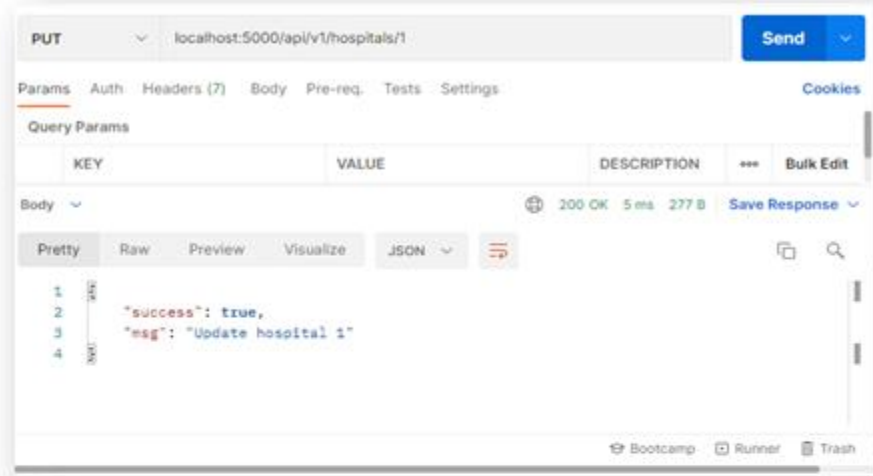
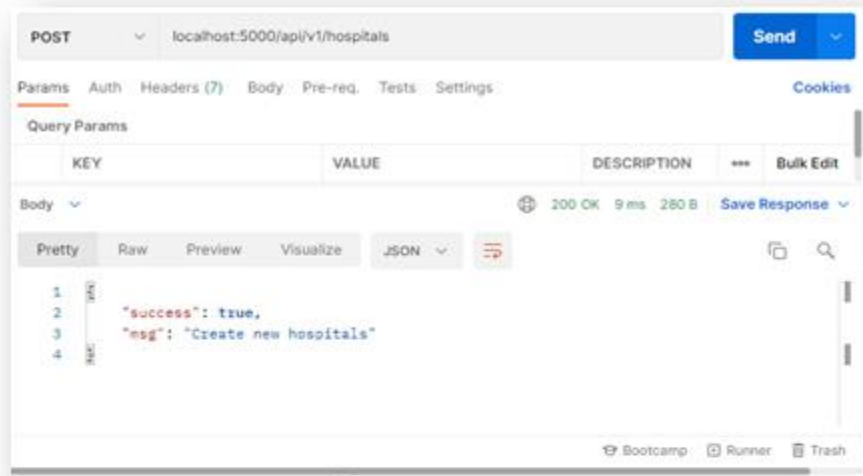
```
const express = require('express');  
const dotenv = require('dotenv');  
  
//Load env vars  
dotenv.config({path: './config/config.env'});  
  
const app=express();  
  
app.get('/api/v1/hospitals', (req,res) => {  
  res.status(200).json({success:true, msg:'Show all hospitals'});  
});
```

server.js : add all routes (2/3)

```
app.get('/api/v1/hospitals/:id', (req, res) => {  
  res.status(200).json({success:true, msg:`Show hospital  
${req.params.id}`});  
});  
  
app.post('/api/v1/hospitals', (req, res) => {  
  res.status(200).json({success:true, msg:'Create new hospitals'});  
});  
  
app.put('/api/v1/hospitals/:id', (req, res) => {  
  res.status(200).json({success:true, msg:`Update hospital  
${req.params.id}`});  
});
```

server.js : add all routes (3/3)

```
app.delete('/api/v1/hospitals/:id', (req, res) => {  
  res.status(200).json({success:true, msg:`Delete hospital  
${req.params.id}`});  
});  
  
const PORT=process.env.PORT || 5000;  
  
app.listen(PORT, console.log('Server running in ', process.env.NODE_ENV, '  
mode on port ', PORT));
```



Router Files

A close-up photograph of a network switch or router. The device has multiple rows of Ethernet ports. Numerous white Ethernet cables are plugged into the ports, some with labels attached. A few blue and yellow cables are also visible. Several yellow status lights are illuminated, indicating active connections. The text "Router Files" is overlaid in the center of the image.

Route Structure

GET-POST-PUT-DELETE

/api/v1/hospitals

/api/v1/bookings

/api/v1/auth

/api/v1/users

create Express Router

1. create a folder : **routes**
2. create a file under the folder: **hospitals.js**
3. move all **app.get/post/put/delete** from **server.js** to **hospitals.js**
4. at **hospitals.js** : require express & define **router = express.Router()**
5. at **hospitals.js**:
 1. replace **app-> router**
 2. select a word and hit **CTL+D** to replace all for the word
 3. Modify all **'/api/v1/hospitals'** to **'/'** (and **'api/v1/hospitals/:id'** to **'/:id'**)
 4. Add this line at the end of file:

```
module.exports=router;
```

6. at **server.js**: add the following codes:

```
const hospitals = require ('../routes/hospitals');
```

```
app.use ('/api/v1/hospitals',hospitals)
```


server.js

```
const express = require('express');
const dotenv = require('dotenv');
//Route files
const hospitals = require('./routes/hospitals');

//Load env vars
dotenv.config({path: './config/config.env'});

const app=express();

//Mount routers
app.use('/api/v1/hospitals', hospitals);

const PORT=process.env.PORT || 5000;

app.listen(PORT, console.log('Server running in ', process.env.NODE_ENV, ' mode on port ', PORT));
```

./routes/hospitals.js

```
const express = require('express');
const router = express.Router();

const app=express();

router.get('/', (req,res) => {
  res.status(200).json({success:true, msg:'Show all hospitals'});
});

router.get('/:id', (req,res) => {
  res.status(200).json({success:true, msg:`Show hospital ${req.params.id}`});
});

router.post('/', (req,res) => {
  res.status(200).json({success:true, msg:'Create new hospitals'});
});

router.put('/:id', (req,res) => {
  res.status(200).json({success:true, msg:`Update hospital ${req.params.id}`});
});

router.delete('/:id', (req,res) => {
  res.status(200).json({success:true, msg:`Delete hospital ${req.params.id}`});
});

module.exports=router;
```

POST localhost:5000/api/v1/hospitals Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body 200 OK 9 ms 280 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "msg": "Create new hospitals"
4 }
```

Bootcamp Runner Trash

GET localhost:5000/api/v1/hospitals Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body 200 OK 16 ms 278 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "msg": "Show all hospitals"
4 }
```

Bootcamp Runner Trash

PUT localhost:5000/api/v1/hospitals/1 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body 200 OK 5 ms 277 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "msg": "Update hospital 1"
4 }
```

Bootcamp Runner Trash

GET localhost:5000/api/v1/hospitals/1 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body 200 OK 5 ms 275 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "msg": "Show hospital 1"
4 }
```

Bootcamp Runner Trash

DELETE localhost:5000/api/v1/hospitals/1 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

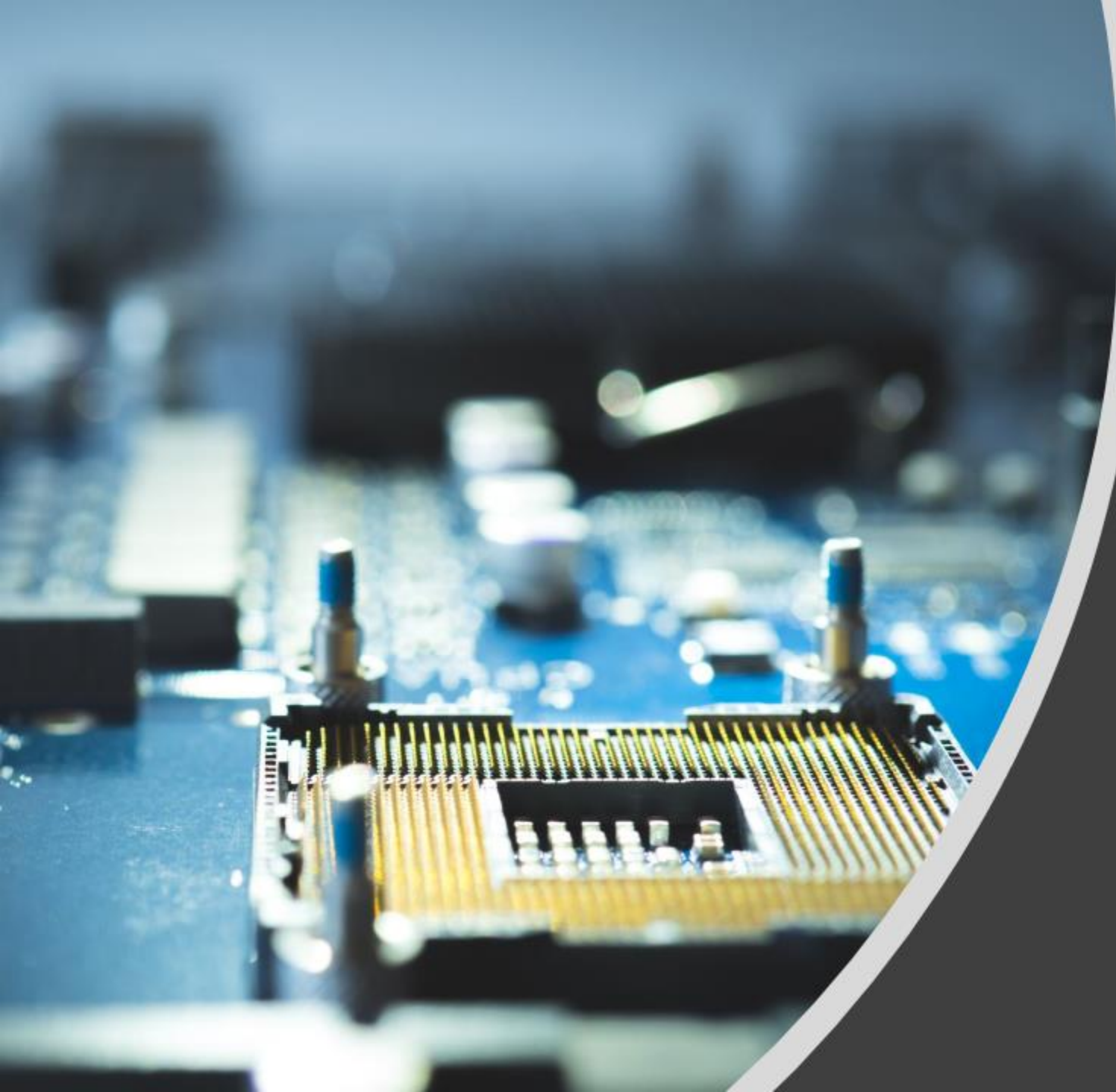
Body 200 OK 4 ms 277 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "msg": "Delete hospital 1"
4 }
```

Bootcamp Runner Trash

Let's test our REST methods:



The Controller Files

create Controller Methods

> create a folder :
controllers

> create a file under
the folder: **hospitals.js**

controllers/hospitals.js

```
//@desc    Get all hospitals
//@route    GET /api/v1/hospitals
//@access    Public
exports.getHospitals=(req,res,next)=>{
  res.status(200).json({success:true, msg:'Show all hospitals'});
};

//@desc    Get sigle hospital
//@route    GET /api/v1/hospitals/:id
//@access    Public
exports.getHospital=(req,res,next)=>{
  res.status(200).json({success:true, msg:`Show hospital ${req.params.id}`});
};

//@desc    Create new hospital
//@route    POST /api/v1/hospitals
//@access    Private
exports.createHospital=(req,res,next)=>{
  res.status(200).json({success:true, msg:'Create new hospitals'});
};

//@desc    Update hospital
//@route    PUT /api/v1/hospitals/:id
//@access    Private
exports.updateHospital=(req,res,next)=>{
  res.status(200).json({success:true, msg:`Update hospital ${req.params.id}`});
};

//@desc    Delete hospital
//@route    DELETE /api/v1/hospitals/:id
//@access    Private
exports.deleteHospital=(req,res,next)=>{
  res.status(200).json({success:true, msg:`Delete hospital ${req.params.id}`});
};
```

Create & export
method for each
REST verb.

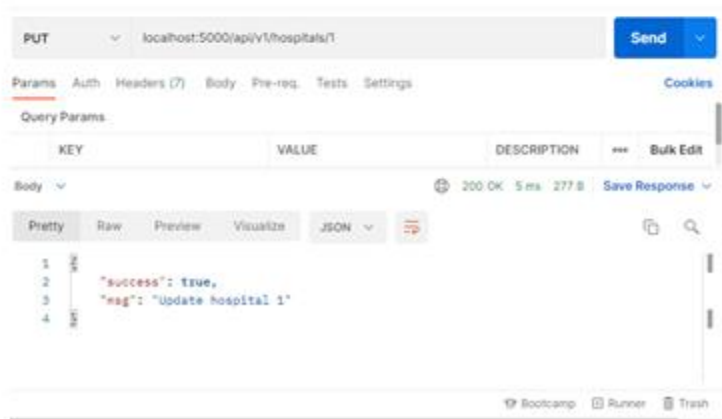
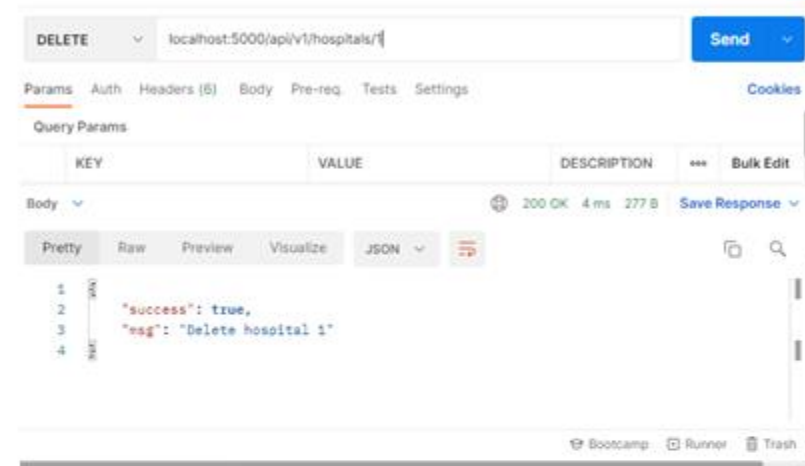
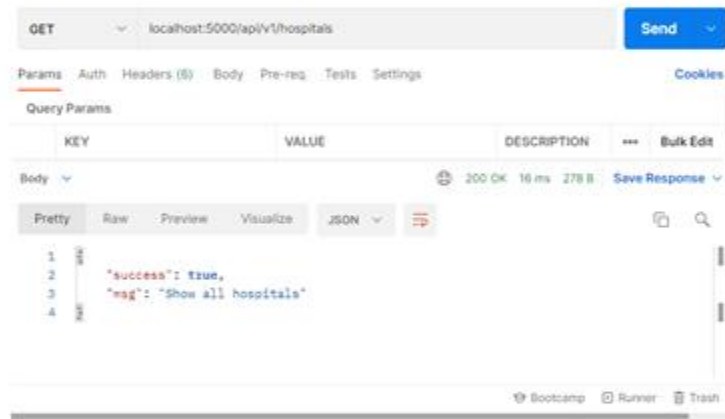
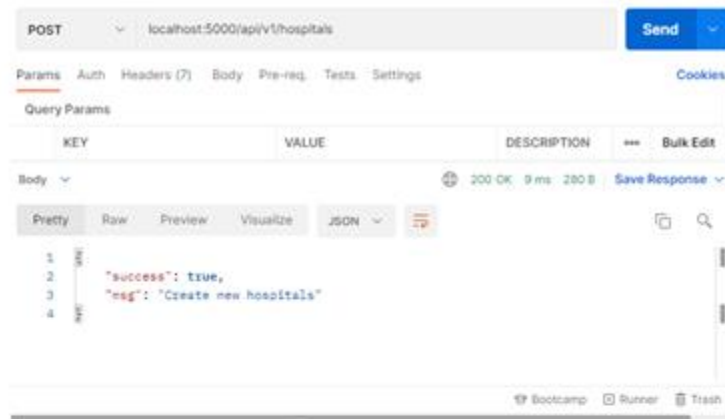
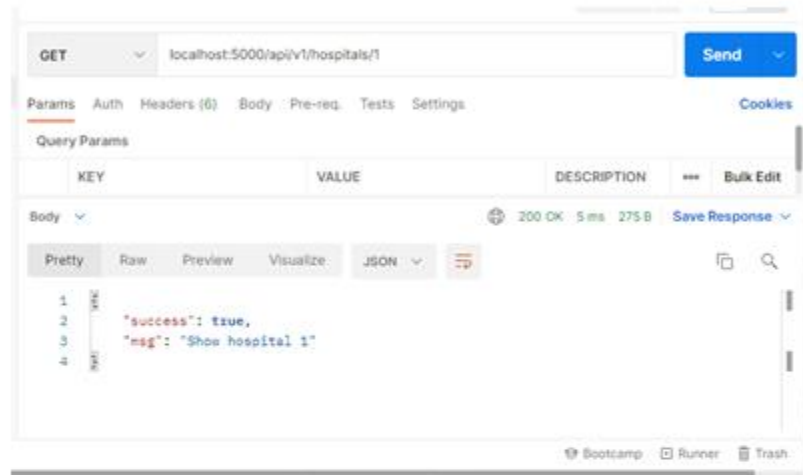
routes/hospitals.js

```
const express = require('express');
const {getHospitals, getHospital, createHospital, updateHospital, deleteHospital} = require('../controllers/hospitals');

const router = express.Router();

router.route('/').get(getHospitals).post(createHospital);
router.route('/:id').get(getHospital).put(updateHospital).delete(deleteHospital);

module.exports=router;
```

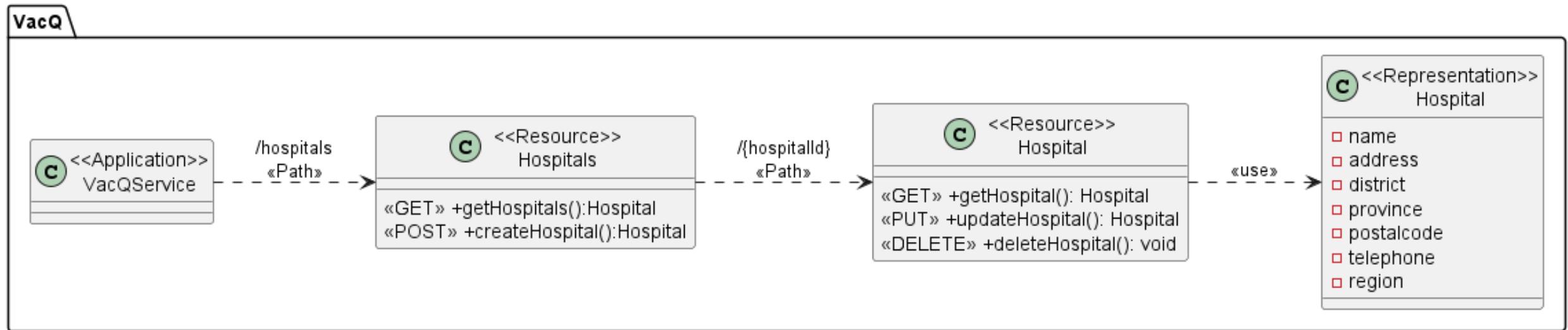



Let's test our REST methods:

to submit your assignment via Github Classroom

```
git add .  
git commit -m 'Your comment'  
git push -u origin main
```

Class Diagram: UML Profile



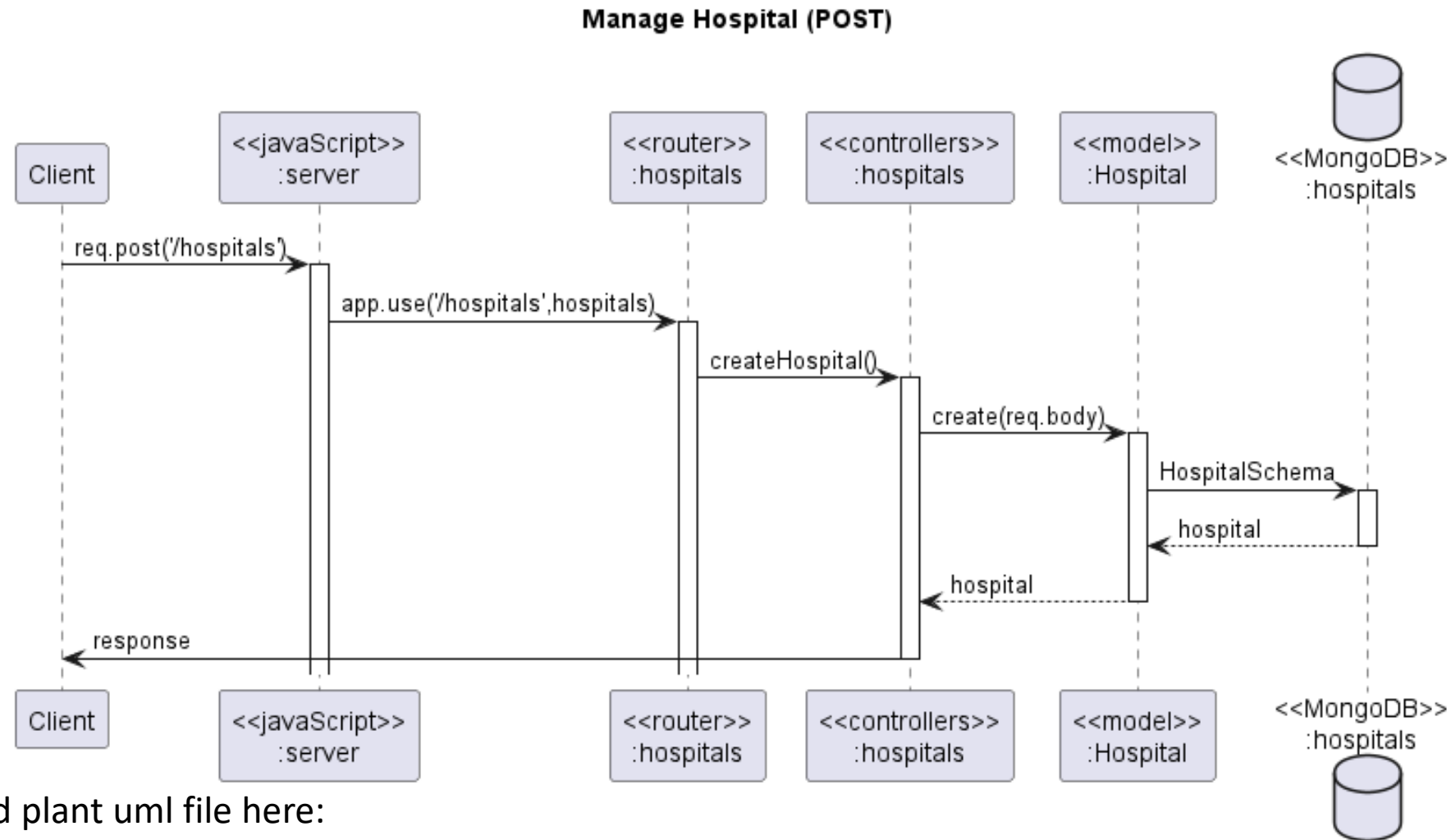
download plant uml file here:

https://drive.google.com/file/d/1sDnxMFs3TE_AQjQJgQEG6CmArm6dzpU/view?usp=sharing

Ref: <https://sparxsystems.us/go/restapi/>

Sequence Diagram: Manage Hospital (post)

Sample VacQ Sequence Diagram



download plant uml file here:

<https://drive.google.com/file/d/1sHwX0LViAwycw7ydAy2M8uFRO5KrUA9H/view?usp=sharing>

Conclusions

Basic REST APIs

- GET
- POST
- PUT
- DELETE

Develop APIs with Node.JS

- Router Files
- Controller Files
- (Model Files)

Test APIs with Postman