Building a Time Server with Node.js

A Detailed Guide

By
Shruti Kavishwar
San Francisco Bay University

Guided By: Porf. Henry Chang

Agenda

- Introduction to NodeJS
- Project Setup
- Server Structure
- Helper Functions
- Creating the Server
- Handling API Requests
- Sending Response
- Running the Server
- Testing the API
- Conclusion and Learnings

Introduction

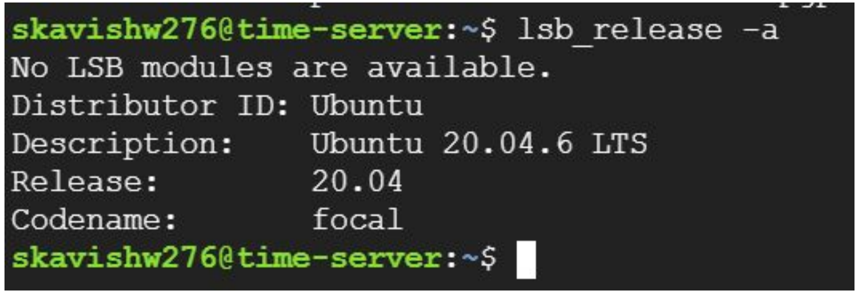
- **Node.js** is an **open source server environment** that uses JavaScript on server.
- A Node.js application runs within a single process, without generating a new thread for each request.
- Node.js includes **asynchronous I/O primitives** as a part of its standard library, which prevents JavaScript code from blocking and, in general, libraries in Node.js are developed using **non-blocking paradigms**.
- This makes blocking behaviour the exception instead of the rule.
- It's built on **Chrome's V8 JavaScript engine** and uses an **event-driven**, non-blocking I/O model, which makes it **lightweight and efficient**. It's perfect for data-intensive real-time applications that run across distributed devices.

Project Setup

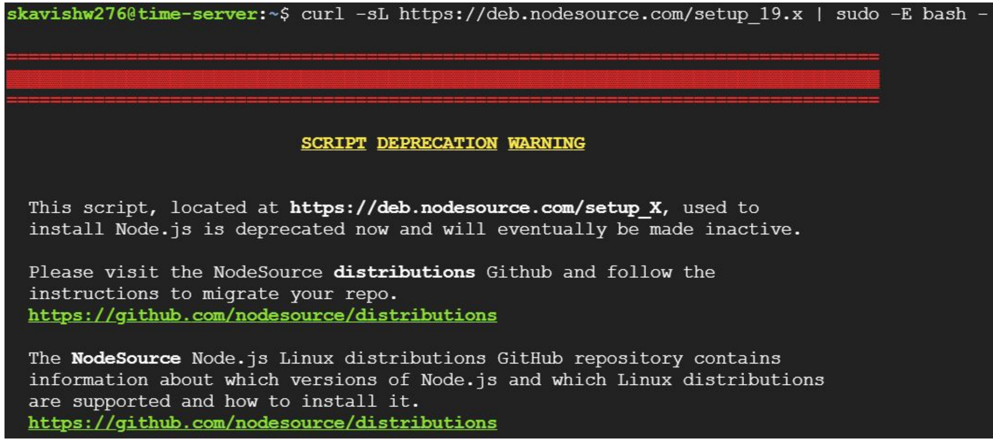
- Create a **Ubuntu VM** on your **Google Cloud Platform**
- Install **Node.js** and **npm** (Node Package Manager) on the VM
- Initialize a new Project using **$npm init** which creates a **package.json** file to manage your project's dependencies.
- The project directory typically includes an app.js (or server.js) file for your server code, along with other directories for routes, views, public assets etc.
- Enable **Firewall rule** on GCP to be able **to communicate on port 8000** in order to use the **external IP** address of the VM to connect from outside.

Screenshots of the Project Setup

- Check the ubuntu version by executing below command
  - `$ lsb_release -a`
- Enable NodeSource repository by executing below command on the Ubuntu VM.
  - `$ curl -sL https://deb.nodesource.com/setup_19.x | sudo -E bash -`



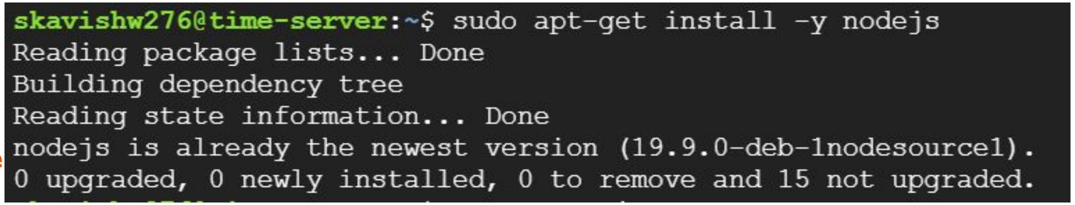
```
skavishw276@time-server:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.6 LTS
Release:        20.04
Codename:       focal
skavishw276@time-server:~$
```



```
skavishw276@time-server:~$ curl -sL https://deb.nodesource.com/setup_19.x | sudo -E bash -

                           SCRIPT DEPRECATION WARNING

  This script, located at https://deb.nodesource.com/setup_X, used to
  install Node.js is deprecated now and will eventually be made inactive.

  Please visit the NodeSource distributions Github and follow the
  instructions to migrate your repo.
  https://github.com/nodesource/distributions

  The NodeSource Node.js Linux distributions GitHub repository contains
  information about which versions of Node.js and which Linux distributions
  are supported and how to install it.
  https://github.com/nodesource/distributions
```

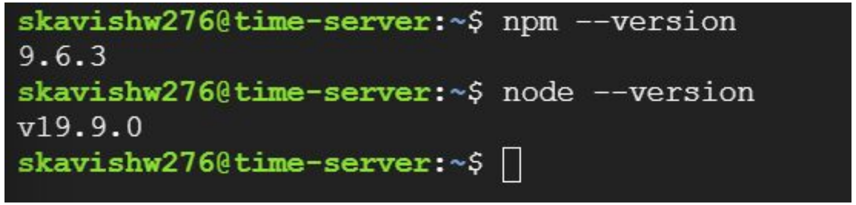
Screenshots of the Project Setup

- Install Node and npm using below commands:
  - `$sudo apt install nodejs`
  - `$sudo apt-get install -y node`

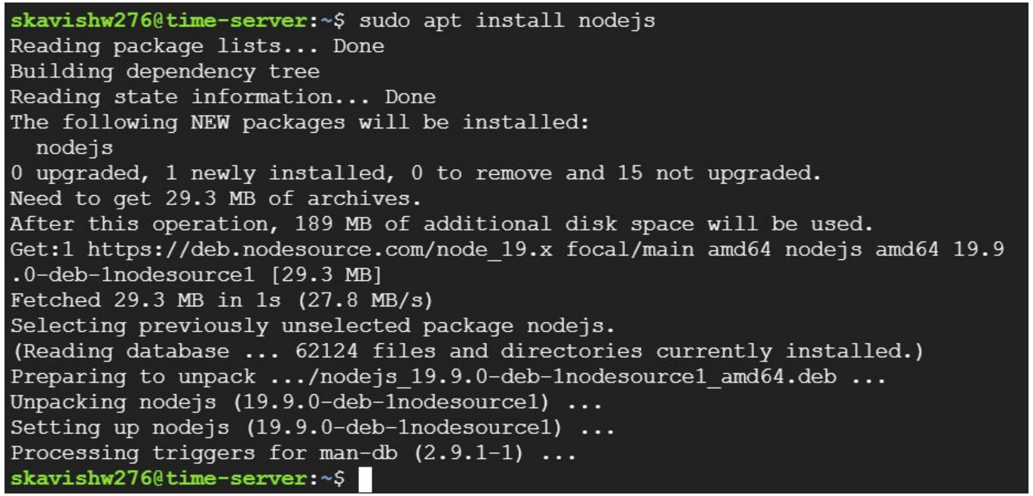


```
skavishw276@time-server:~$ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
nodejs is already the newest version (19.9.0-deb-1nodesource1).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
```

- Check the node and npm version using below commands:
  - `$ npm --version`
  - `$ node --version`



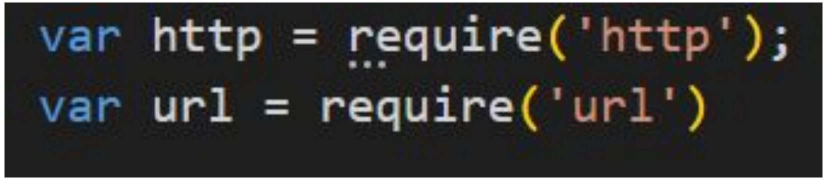
```
skavishw276@time-server:~$ npm --version
9.6.3
skavishw276@time-server:~$ node --version
v19.9.0
skavishw276@time-server:~$
```



```
skavishw276@time-server:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 15 not upgraded.
Need to get 29.3 MB of archives.
After this operation, 189 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_19.x focal/main amd64 nodejs amd64 19.9
.0-deb-1nodesource1 [29.3 MB]
Fetched 29.3 MB in 1s (27.8 MB/s)
Selecting previously unselected package nodejs.
(Reading database ... 62124 files and directories currently installed.)
Preparing to unpack .../nodejs_19.9.0-deb-1nodesource1_amd64.deb ...
Unpacking nodejs (19.9.0-deb-1nodesource1) ...
Setting up nodejs (19.9.0-deb-1nodesource1) ...
Processing triggers for man-db (2.9.1-1) ...
skavishw276@time-server:~$
```

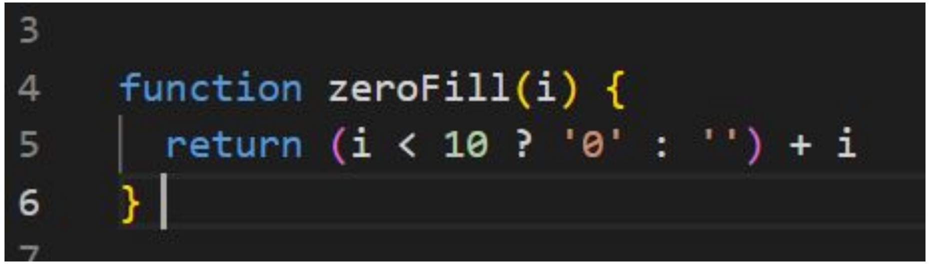
Server Structure

- A basic node.js server is created using the **http.createServer()** method from the **http module**.
- Returns a **new instance** of http.Server.
- The server is setup to listen for requests, which are handled by a **callback function**.
- This function takes two arguments **req** (the request object) and **res** (the response object).
- The **url module** is used to **parse** the request **URL**

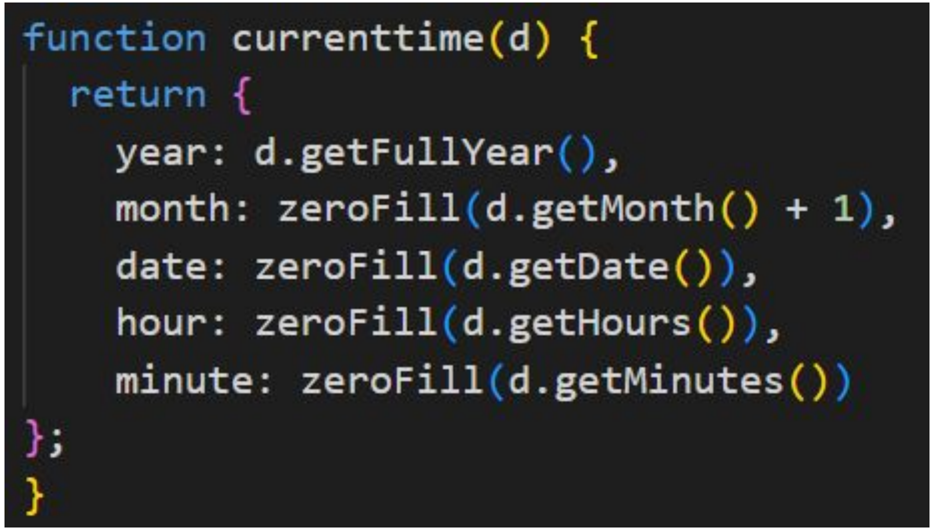


```
var http = require('http');
var url = require('url')
```

Helper Functions



```
function zeroFill(i) {
  return (i < 10 ? '0' : '') + i
}
```

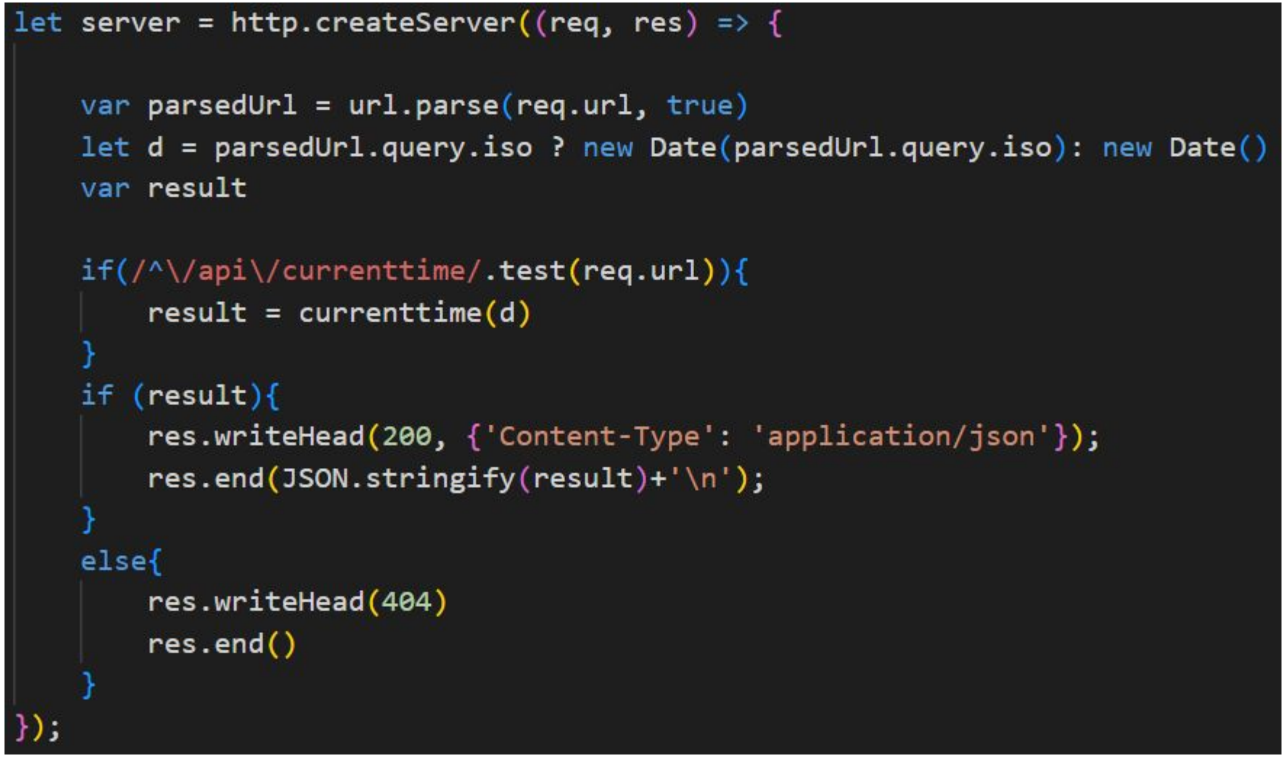


```
function currenttime(d) {
  return {
    year: d.getFullYear(),
    month: zeroFill(d.getMonth() + 1),
    date: zeroFill(d.getDate()),
    hour: zeroFill(d.getHours()),
    minute: zeroFill(d.getMinutes())
};
}
```

- The **zeroFill(i)** function takes a number as an argument and returns the number as a string, **prefixed with a zero if the number is less than 10**. This is used to ensure consistent formatting for the time.

- The **currenttime(d)** function takes a **Date object** and returns an object containing the **year, month, date, hour, and minute**, each formatted as a **two-digit string** using **zeroFill(i)**

Creating the Server

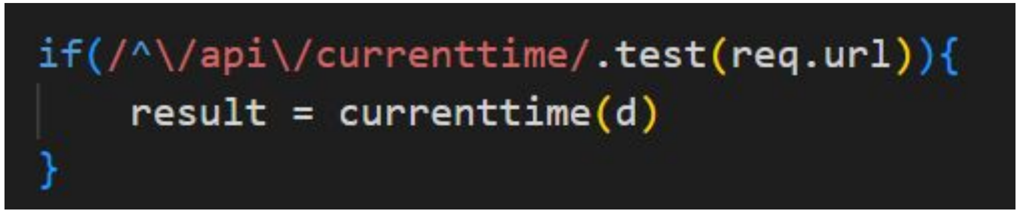


```
let server = http.createServer((req, res) => {

    var parsedUrl = url.parse(req.url, true)
    let d = parsedUrl.query.iso ? new Date(parsedUrl.query.iso): new Date()
    var result

    if(/^\/api\/currenttime/.test(req.url)){
        result = currenttime(d)
    }
    if (result){
        res.writeHead(200, {'Content-Type': 'application/json'});
        res.end(JSON.stringify(result)+'\n');
    }
    else{
        res.writeHead(404)
        res.end()
    }
});
```

- The server is created using **http.createServer()**, which takes a callback function as an argument. This function is called for every request made to the server. Inside this function, the request URL is parsed and the appropriate response is generated based on the URL path.

Handling API Requests

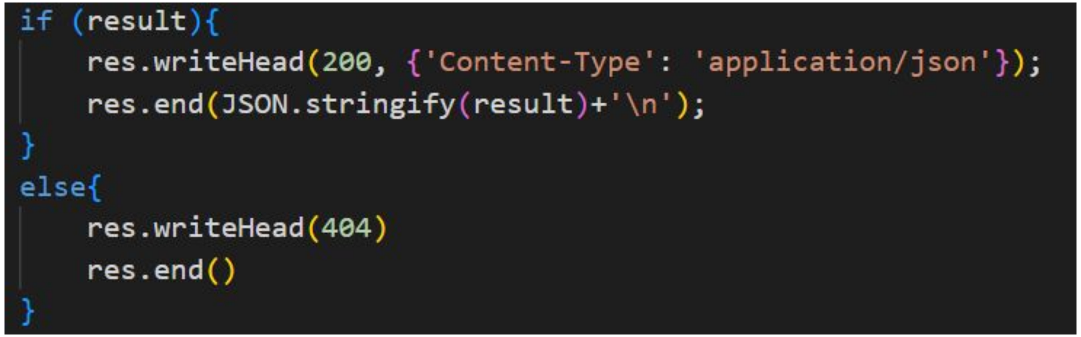
- The server handles **API requests** by checking the path of the request URL. If the path matches **/api/currenttime**, it calls the **currenttime(d)** function to generate the current time data.



```
if(/^\/api\/currenttime/.test(req.url)){
    result = currenttime(d)
}
```

Sending Response

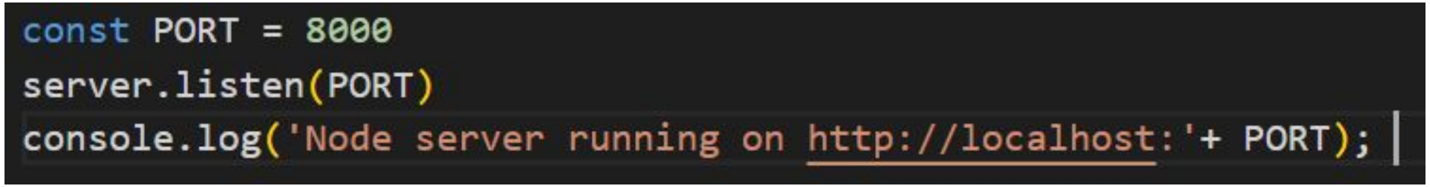
- The server sends a response using **res.writeHead()** to set the response status code and headers, and **res.end()** to end the response and send it to the client. The response data is converted to a JSON string using **JSON.stringify()**.



```
if (result){
    res.writeHead(200, {'Content-Type': 'application/json'});
    res.end(JSON.stringify(result)+'\n');
}
else{
    res.writeHead(404)
    res.end()
}
```

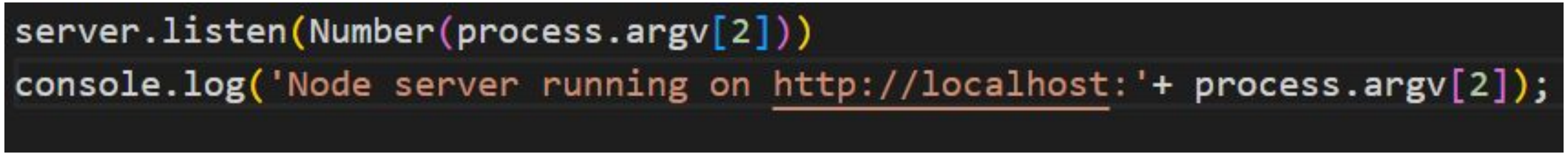
Running the Server

- The server is started using **server.listen()**, which takes the port number as an argument. The port number can also be provided from the command line arguments **(process.argv[2])**.



```
const PORT = 8000
server.listen(PORT)
console.log('Node server running on http://localhost:'+ PORT);
```

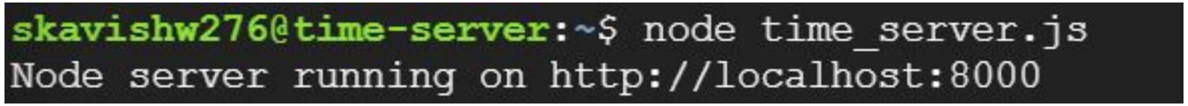
OR



```
server.listen(Number(process.argv[2]))
console.log('Node server running on http://localhost:'+ process.argv[2]);
```

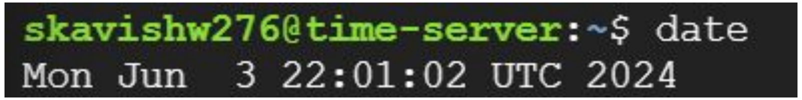
Testing the API

- You can test the API using tools like Postman or curl. Simply send a request to your **server's IP address and port number**, **followed by /api/currenttime**, and check the response.
- Execute the Script

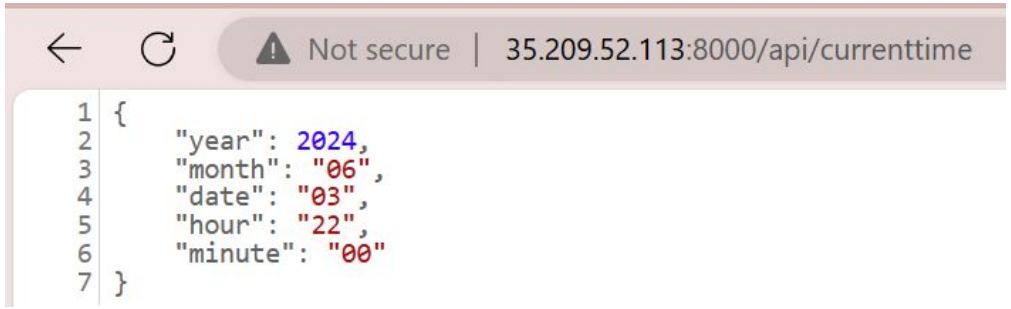


```
skavishw276@time-server:~$ node time_server.js
Node server running on http://localhost:8000
```

- Using the URL to access API: **http://<external-ip>:port-no/api/currenttime**
  - http://35.209.52.113:8000/api/currenttime (When using the browser outside the Ubuntu VM)
  - http://localhost:8000/api/currenttime (When using browser from the Ubuntu VM)
- Match the current time of your Ubuntu VM and the time displayed by the API.



```
skavishw276@time-server:~$ date
Mon Jun  3 22:01:02 UTC 2024
```



```
{
    "year": 2024,
    "month": "06",
    "date": "03",
    "hour": "22",
    "minute": "00"
}
```

Conclusion

- Successfully created the Ubuntu VM on Google Cloud Platform
- Successfully installed the necessary packages for running node.js
- The JavaScript executed correctly to show the current time of the machine it is being executed on.

Learnings

- Accessing the script output outside of the Ubuntu VM by enabling the firewall rules on GCP
- Error Handling: The server sends a 404 response when the requested URL path does not match any expected paths
- URL parsing: The url.parse() method is used to parse the request URL into its components
- Request Handling: The server is setup to handle requests using callback function. It uses two arguments "req" and "res"

GitHub: https://github.com/ShrutiK02/JavaScript/tree/main/TimeServer