

```
export async function getAllNotes(req, res) {
```

Alternative is

```
export async function getAllNotes([], res) {
```

MIDDLEWARE

Middleware is a function that runs in the middle between the **request** and the **response**.



We can do something with the response.

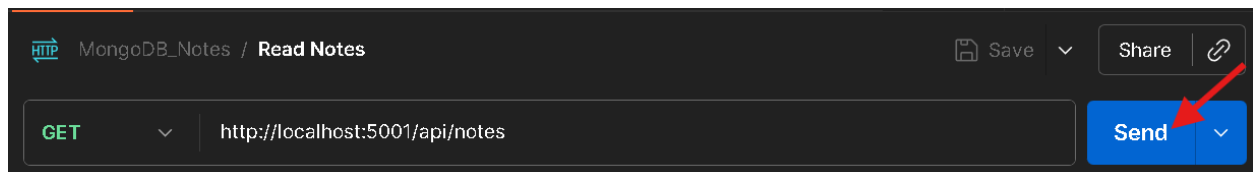
In **server.js**:

```
// middleware
app.use(express.json())

app.use((req, res, next) => {
  console.log("We just got the new req")
  next()
})

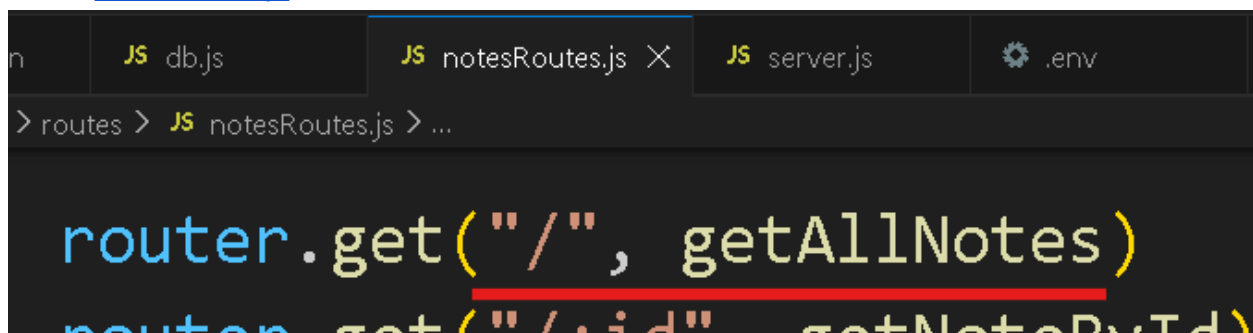
app.use("/api/notes", notesRoutes)
```

If in postman we want to get all the notes:



At first we console log it["We just got new request"], then next() function to call the **getAllNotes** function from get request as given below.

And in [notesRoutes.js](#) has:



In [server.js](#):

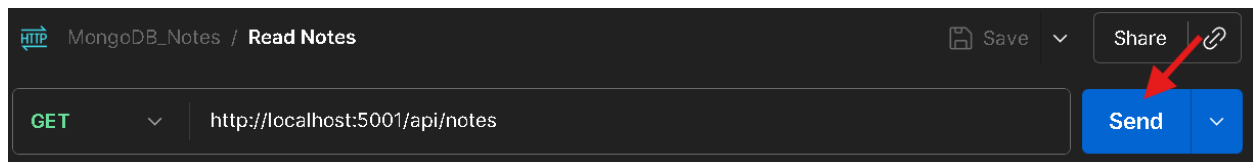
```
JS db.js JS notesRoutes.js JS server.js X .env JS notesController.js
> JS server.js > app.use() callback

// middleware
app.use(express.json())

app.use((req, res, next) => {
  console.log(`Req method: ${req.method}.\nReq URL: ${req.url}.`)
  next()
})

app.use("/api/notes", notesRoutes)
```

In postman: to get all notes

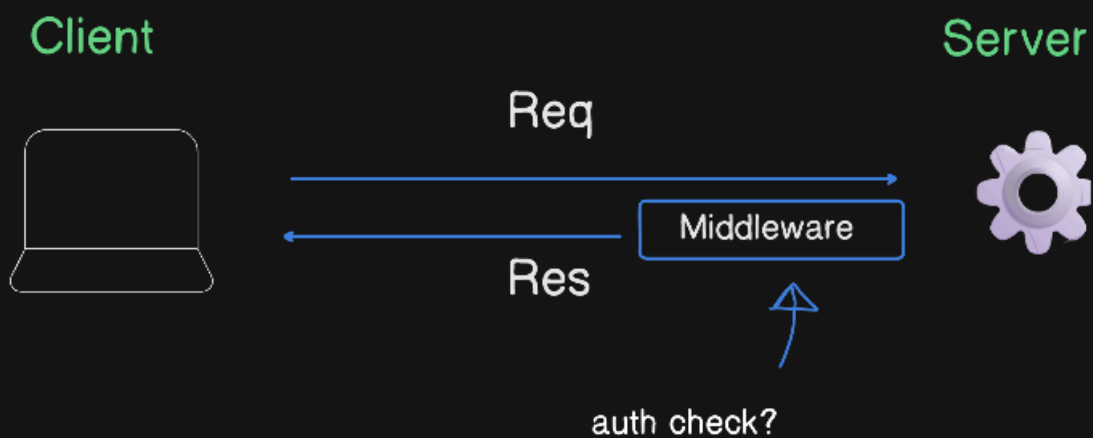


In terminal:

```
[nodemon] restarting due to changes...
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
Req method: GET.
Req URL: /api/notes.
[]
```

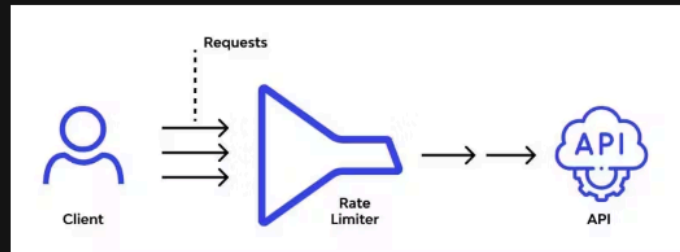
MIDDLEWARE

Middleware is a function that runs in the middle between the **request** and the **response**.



One of the most powerful use case of middle ware is authentication check.

RATE LIMITING



👉 **Rate limiting** is a way to control how often someone can do something on a website or app like how many times they can refresh a page, make a request to an API, or try to log in.

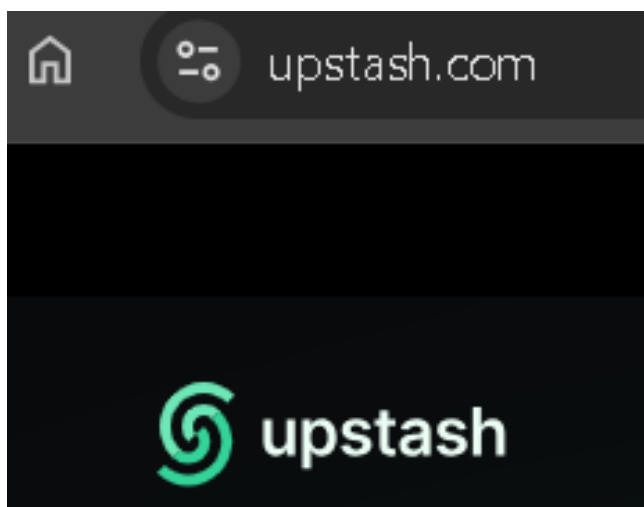
👉 Only 100 requests per user every 15 minutes

RATE LIMITING HELPS WITH

- 👉 Preventing abuse (e.g., stopping someone from making 1000 login attempts in a minute)
- 👉 Protecting servers from getting overwhelmed

429 Too Many Requests

To implement this **rate limiting**, we can use **upstash**:



Visit:

Has a free plan:

Free

\$0



Perfect for prototypes
and hobby projects.

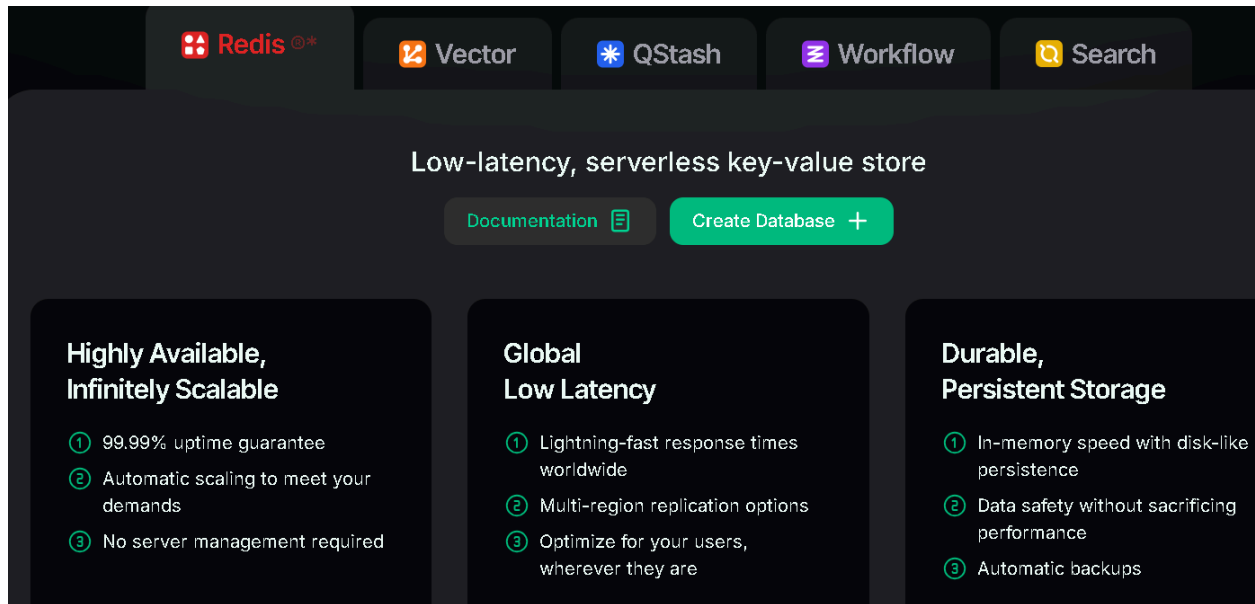
Data Size

256 MB

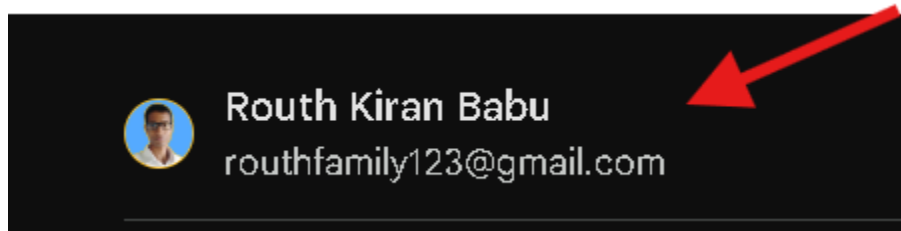
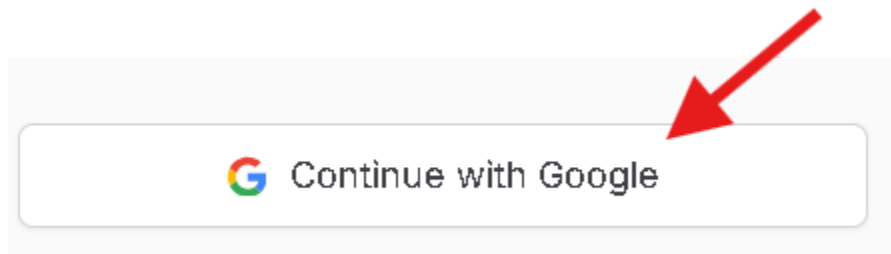
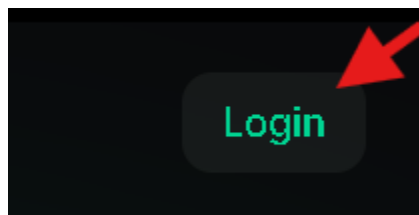
Monthly Commands

500 K

[Start Now](#)



Login to upstash:



Google will allow upstash.com to access this info about you



Routh Kiran Babu

Name and profile picture



routhfamily123@gmail.com

Email address

Review upstash.com's [Privacy Policy](#) and [Terms of Service](#) to understand how upstash.com will process and protect your data.

To make changes at any time, go to your [Google Account](#).

Learn more about [Sign in with Google](#).

Cancel

Continue



Personal

Redis

QStash

Workflow

Vector

...

4



Redis - Low-latency, serverless key-value store

COMMANDS

0

AVERAGE STORAGE

0 B

COST

\$0.00


Search...



Import...



Create Database

 Personal

Redis




QStash

Workflow

Vector

...

4

 **Redis** - Low-latency, serverless key-value store

COMMANDS
0

AVERAGE STORAGE
0 B

COST
\$0.00

 Search...

 Import...

 Create Database

Create Database

Select a Plan

Name

thinkboard

Primary Region

Mumbai, India (ap-south-1)

Choose the region where most of your writes will take place.

Read Regions

Select read regions (optional)

Read regions are only available for paid plans



Eviction

Enable to evict entries when max data size is reached.

 Add a payment method for paid plans.

Cancel

Next

Create Database

Select a Plan



Free

Free forever for hobbyists.

Max Data Size: **256 MB**

Max Monthly Bandwidth: **10 GB**



Pay as You Go

\$0.2 / 100K commands



Fixed 250 MB - 500 GB

Starting from **\$10**



Add a payment method for paid plans.

Learn more about [the plans](#) 

Back

Next

Create Database

Select a Plan



thinkboard

Mumbai, India (ap-south-1)

DEFAULT FEATURES



Persistence



REST API



TLS



Global

Monthly : \$0



Add a payment method for paid plans.

Back

Create

To connect to the DataBase.

For env file: copy it

Redis / thinkboard ☆ ...

Free Tier

AWS

Mumbai, India

ap-south-1

Global

Docs

SDK

Connect

Connect to your Redis database from anywhere

REST TCP

☐ Read-Only Token



```
1 UPSTASH_REDIS_REST_URL="https://fit-boan-42819.upstash.io"
2 UPSTASH_REDIS_REST_TOKEN="*****"
```

Paste it in .env file:

```
.env
1 MONGO_URI=mongodb+srv://routhfamily123_db_user:dRoCgH5M0Bb
2 PORT=5001
3 UPSTASH_REDIS_REST_URL="https://fit-boar-42819.upstash.io"
4 UPSTASH_REDIS_REST_TOKEN="AadDAAIncDI2YTMwOTQwZTA3MTM0ZDk1"
```

Install dependencies:

Stop the server running.

Use the command in backend:

`npm i @upstash/ratelimit@2.0.5 @upstash/redis@1.34.9`

```
l\backend> npm i @upstash/ratelimit@2.0.5 @upstash/redis@1.34.9
```

In package.json:

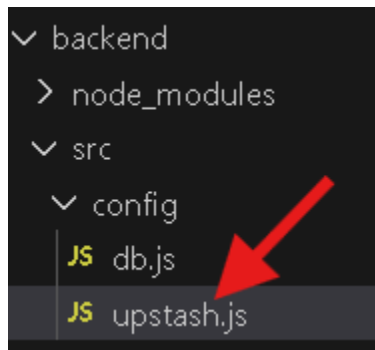
```
{ } package.json X JS db.js JS notesRoutes.js JS server.js .env JS notesCo
backend > { } package.json > abc main
13 "license": "ISC",
14 "dependencies": {
15   "@upstash/ratelimit": "^2.0.5",
16   "@upstash/redis": "^1.34.9",
17   "dotenv": "^16.5.0",
```

Run the server/app.

```
PS C:\Users\kiran\OneDrive\Desktop\mern-thinkboard\backend> npm run dev
> backend@1.0.0 dev
> nodemon src/server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
□
```

Create upstash under config:



Code in upstash.js:

```
import {Ratelimit} from "@upstash/ratelimit"
import {Redis} from "@upstash/redis"
```

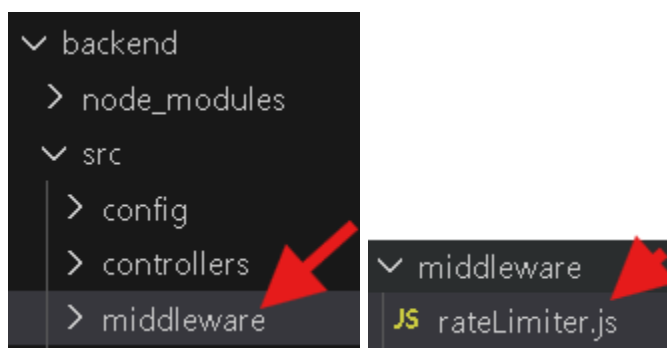
```
import dotenv from "dotenv"
dotenv.config()
```

```
// create a ratelimiter that allows 10 requests per 20 seconds(s)[.slidingWindow(10, "20 s")]
const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(10, "20 s")
})
```

```
export default ratelimit;
```

Best practices:

Create middleware folder:



folder where we can add custom

middlewares.

Code in rateLimiter.js:

```
const rateLimiter = async (req, res, next) => {
```

```
}
```

export default rateLimiter;

Code in [server.js](#):

```
import rateLimiter from "../middleware/rateLimiter.js"

connectDB()

// middleware
app.use(express.json()) // this mi
app.use(rateLimiter)
// Our simple custom middleware
// app.use((req, res, next) => {
//     console.log(`Req method: ${req.method}`)
//     next()
// })

app.use("/api/notes", notesRoutes)
```

Code in [rateLimiter.js](#):

```
import ratelimit from "../config/upstash.js";
```

```
const rateLimiter = async (req, res, next) => {
  try {
    // To check success rate
    const {success} = await ratelimit.limit("my-limit-key")
    if(!success){
      // 429=To many Request
      return res.status(429).json({message: "To many request, please try again later..."})
    }
  } catch (error) {
    console.log(error)
  }
  next()
}
```

```

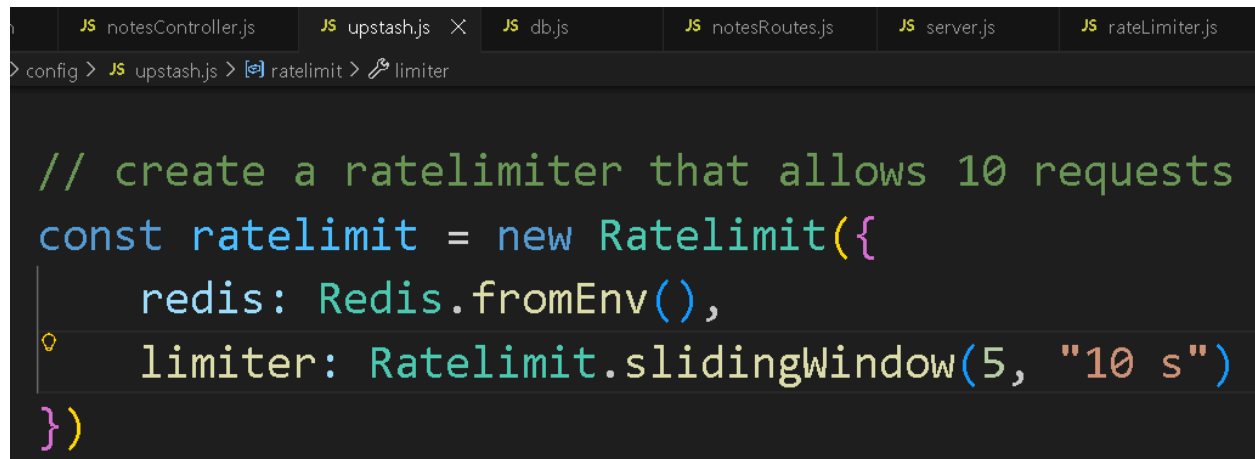
    }
    next() // if success then call the function using next()
  } catch (error) {
    console.log("Rate limit Error", error)
    // we can also add error within next
    next(error)
  }
}
}

```

export default rateLimiter;

For now let's make the limit as:

In [upstash.js](#): [5 request per 10 seconds]



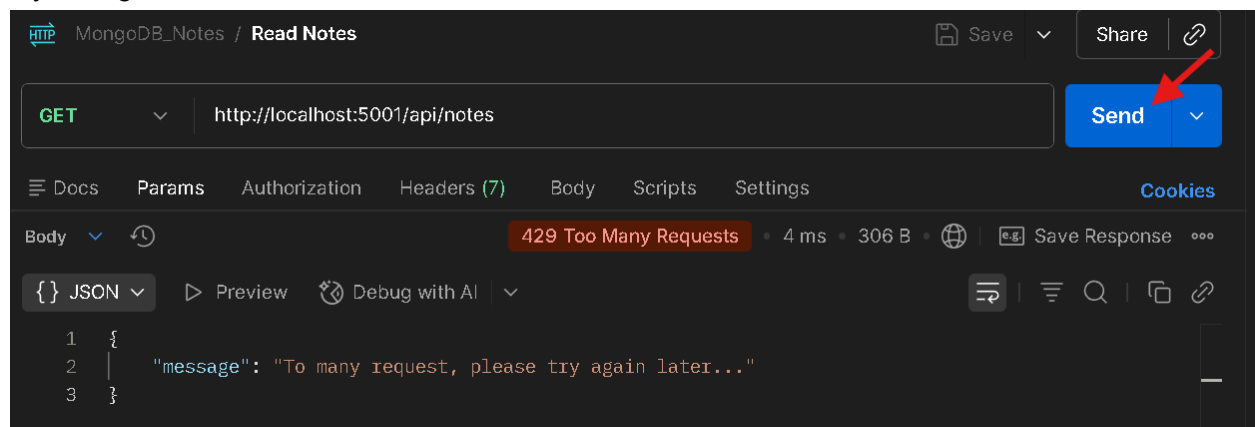
```

// create a ratelimiter that allows 10 requests
const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(5, "10 s")
})

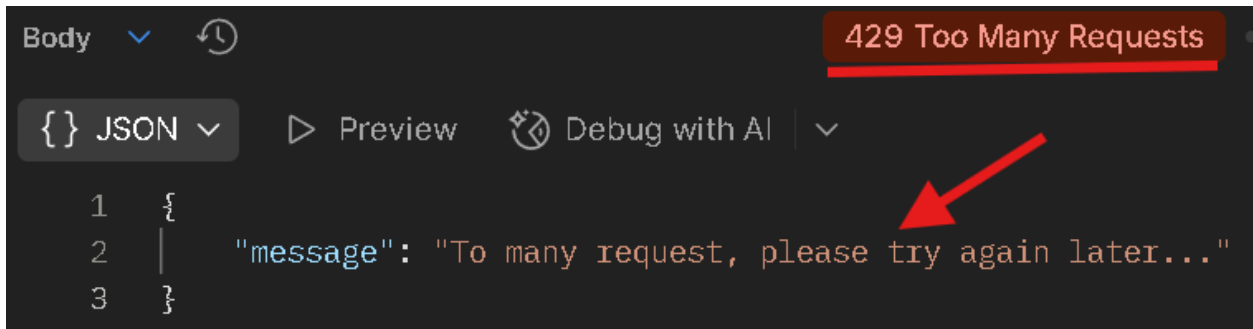
```

In postman:

Try hitting send button more than 5 within 10 Seconds.



Response:



As per the code:

```
rateLimiter = async (req, res, next) => {
  if (!success) {
    // 429=To many Request
    return res.status(429).json({message: "To many request, please try again later..."})
  }
  next() // if success then call the function using next()
}
```

In terminal:

```
[nodemon] restarting due to changes...
[nodemon] starting `node src/server.js`
Server started on PORT: 5001
MongoDB connected Successfully...
█
```

At first server is connected then database.

It's better to connected the database, if connected then start the server.

So in server.js , change the code **FROM:**

connectDB()

// middleware

app.use(express.json()) // this middleware will parse JSON bodies: req.body

app.use(rateLimiter)

// Our simple custom middleware

// app.use((req, res, next) => {

// console.log(`Req method: \${req.method}.\nReq URL: \${req.url}.`)

// next()

// })

app.use("/api/notes", notesRoutes)

app.listen(PORT, () => {

console.log("Server started on PORT:", PORT)

})

TO:

```
//connectDB()
```

```
// middleware
app.use(express.json()) // this middleware will parse JSON bodies: req.body
app.use(rateLimiter)
// Our simple custom middleware
// app.use((req, res, next) => {
//   console.log(`Req method: ${req.method}.\nReq URL: ${req.url}.`)
//   next()
// })
```

```
app.use("/api/notes", notesRoutes)
```

```
connectDB().then(() =>{
  app.listen(PORT, () => {
    console.log("Server started on PORT:", PORT)
  })
})
```

Code in [server.js](#):

```
import express from "express"
import notesRoutes from "../routes/notesRoutes.js"
import { connectDB } from "../config/db.js"
```

```
import dotenv from "dotenv"
import rateLimiter from "../middleware/rateLimiter.js"
dotenv.config()
```

```
//console.log(process.env.MONGO_URI)
```

```
const app = express()
// if process.env.PORT is undefined then PORT = 5001(by default value)
const PORT = process.env.PORT || 5001
```

```
//connectDB()
```

```
// middleware
app.use(express.json()) // this middleware will parse JSON bodies: req.body
app.use(rateLimiter)
// Our simple custom middleware
// app.use((req, res, next) => {
//   console.log(`Req method: ${req.method}.\nReq URL: ${req.url}.`)
//   next()
// })
```

```
// })

app.use("/api/notes", notesRoutes)

connectDB().then(() =>{
  app.listen(PORT, () => {
    console.log("Server started on PORT:", PORT)
  })
})
```

Code in upstash.js:

```
import {Ratelimit} from "@upstash/ratelimit"
import {Redis} from "@upstash/redis"

import dotenv from "dotenv"
dotenv.config()

// create a ratelimiter that allows 100 requests per minute("60 s")
const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(10, "20 s")
})

export default ratelimit;
```

Code in [rateLimiter.js](#):

```
import ratelimit from "../config/upstash.js";

const rateLimiter = async (req, res, next) => {
  try {
    // To check success rate
    // "my-rate-limit" <- can provide ip address
    // block based on ip address, not everyone should
    // be blocked if limit exceeds
    const {success} = await ratelimit.limit("my-rate-limit")
    if(!success){
      // 429=To many Request
      return res.status(429).json({message: "To many request, please try again later..."})
    }
    next() // if success then call the function using next()
  } catch (error) {
    console.log("Rate limit Error", error)
  }
}
```

```
    // we can also add error within next
    next(error)
  }
}
```

```
export default rateLimiter;
```

Code in .env:

```
MONGO_URI=mongodb+srv://routhfamily123_db_user:dRoCgH5MOBbEmJAW@cluster0.dxej
p0q.mongodb.net/notes_db?appName=Cluster0
PORT=5001
UPSTASH_REDIS_REST_URL="https://fit-boar-42819.upstash.io"
UPSTASH_REDIS_REST_TOKEN="AadDAAIncDI2YTMwOTQwZTA3MTM0ZDk1YjgwMWUzY
mUwN2RiOGU3YXAyNDI4MTk"
```