NAMASTE NODE.JS SEASON 3

Episode-6

Hand Written Notes

-By Shanmuga Priya

www.linkedin.com/in/shanmuga-priya-e-tech2

Episode-6

Scheduling cron Jobs

what is Scheduling cron Jobs?

If we need to run a particular task at certain intervals daily we schedule those task with cron to run those tasks automatically after certain periods.

```
npm i node-cron
```

Step 1: create a New file to run cron job

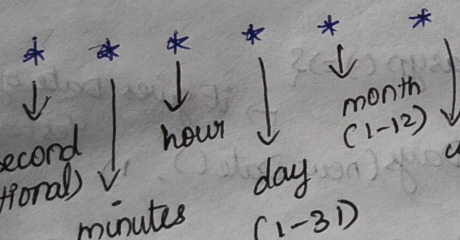
In utils folder create a new file called "cronjobs.js" (it can be anything) and inside it require a node-cron module that we installed.

eg: //cronjob.js

```
const cron = require ("node-cron")

cron.schedule ("* * * * * *", () => {
    console.log ("Hello world" + new Date())
})
```

→ these star represents time



→ if we put 6 stars means the task will run for every sec

→ if 5 stars, the task will run for every min & so on.

Step 2: Place it in app.js

inorder to run this cron jobs as soon as application starts we need to place it in the index file (i.e) app.js file. we just need to require this file in app.js tats it.

crontab.guru -> is a website where we can experiment with this star strings. eg: for 8'o clock → 0 8 * * * this will schedule a task to run at 8 A.M everyday.

min hr day month week

code for sending Email to users who recieved connection request yesterday:

Our task is to send a email everyday at 8:00 AM for the users who have received the connection request previous day.

eg code:

```
const cron = require("node-cron")
const { subDays, startOfDay, endOfDay } = require("date-fns")
                    ↳ we use "date-fns" package to calculate yesterday.
const sendEmail = require("./sendEmail")
const connectReq = require("../models/connectionRequest")

//Scheduling cron Job
cron.schedule("0 8 * * *", asyn()=>{
    try{
        const yesterday = subDays(new Date(), 1)   → it gives Date of yesterday
        const start = startOfDay(yesterday)
        const End = endOfDay(yesterday)            → it gives time stamp
        //querying Collection
        const pendingReq = await connectReq.find({
            status: "interested",
            createdAt: { $gte: start, $lt: End }
        }).populate("fromUserId toUserId")
```

//extracting the email Id of the req receivers.

```
const listOfEmails = [...new Set(pendingReq.map((req) =>
                                    req.toUserId.emailId))]
```

we are using Set to find a unique email Id becoz one emailId can recieve so much req.

```
for(const emails of listOfEmails){
    try{
        const res = await sendEmail.run(
            "New friend Requests are pending for" + email,
            "Please log in to DevTinder to accept the request"
        )

        console.log(res)

    }catch(err){
        console.log(err)
    }

}catch(err){
    console.log(err)
}
})
```

Note:

→ This is perfect for small application as we are looping the email and sending it blocks the code execution.

→ for larger app we should do either queueing (or) batching using some packages like "bee Queue" (or) "BullMQ" (or) simply can use Amazon SES Bulk email sending option.

— x —