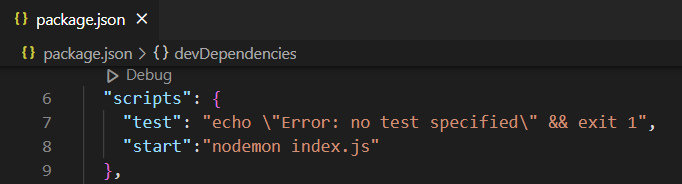
1. Status code
2. baseUrl : http://localhost:4000
3. resource : any data : Users/Products/Customers (we will go with users)
4. basic url formation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | GET | POST | PATCH | PUT | DELETE |
| http://localhost:400/users | y | y | n | n | n |
| http://localhost:400/users/{user-id} | y | n | y | y | y |
| http://localhost:400/users/search?name=raj | y | n | n | n | n |
| http://localhost:400/users/{user-id}/addresses | y | y | n | n | n |
| http://localhost:400/users/{user-id}/addresses/{address-id} | y | n | y | y | y |

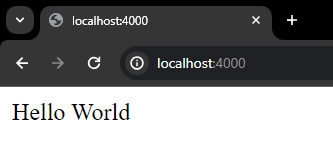
1. we will create new project
   1. create new folder (“node\_api”) and open in vs code
   2. >npm init --yes
   3. >npm i express
   4. >npm i –D nodemon
   5. Create index.js file
   6. Add nodemon script in package.json



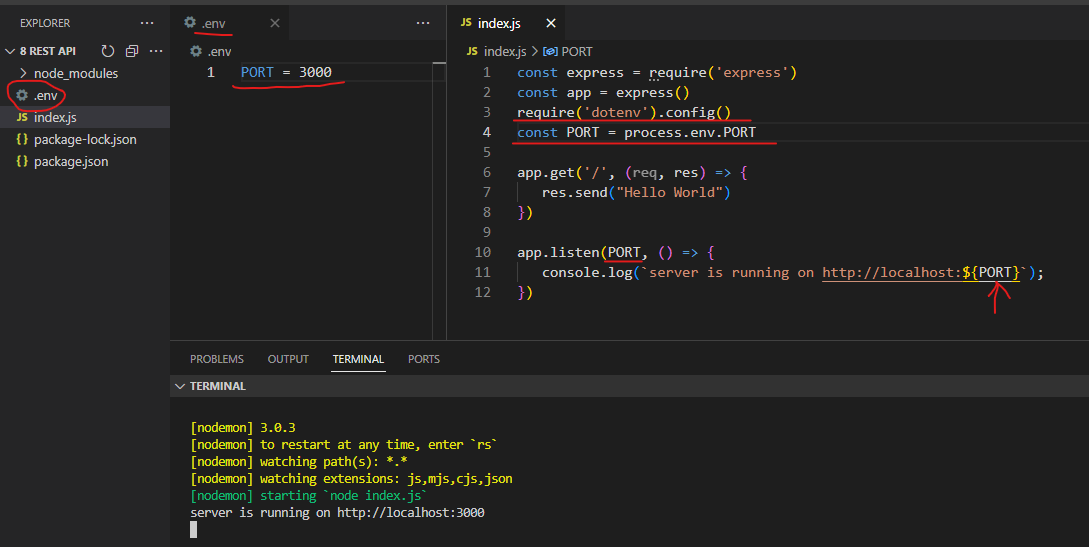
* 1. Write code to create server in index.js



* 1. Start server >npm start
  2. Output



1. Creating env variable at project level
   1. Import dotenv package
      1. >npm i dotenv
   2. Create .env file in your project (like we created index.js)

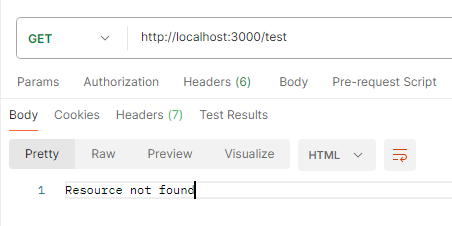


1. Create api to handle wild card routing
   1. Wild card routing always must be at the end

app.get('\*\*', (req, res) => {

   res.status(404).send("Resource not found")

})



1. Creating api to fetch all users

let users = [

   {

      id:1,

      name:"raj",

      age:25

   },

   {

      id:2,

      name:"rani",

      age:35

   }

]

app.get('/users', (req, res) => {

   // res.send(users)

   if(users.length!=0)

   {

      return res.status(200).send(

         {

            message:"following user found",

            data:users

         }

      )

   }

   else

   {

      return res.status(404).send(

         {

            message:`There are no users, please add some`,

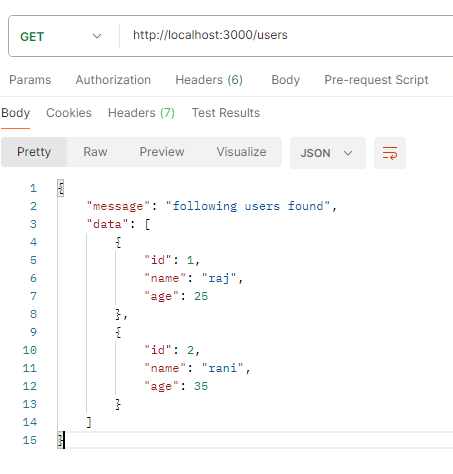
            data:[]

         }

      )

   }

})



1. Creating api to fetch user based on id

app.get('/users/:id', (req, res) => {

   const user\_id = req.params.id

   const user=users.filter(user=>user.id === +user\_id) //+user\_id means, converted to number

   if(user.length!=0)

   {

      return res.status(200).send(

         {

            message:"following user found",

            data:user

         }

      )

   }

   else

   {

      return res.status(404).send(

         {

            message:`user with id ${user\_id} does not exist`,

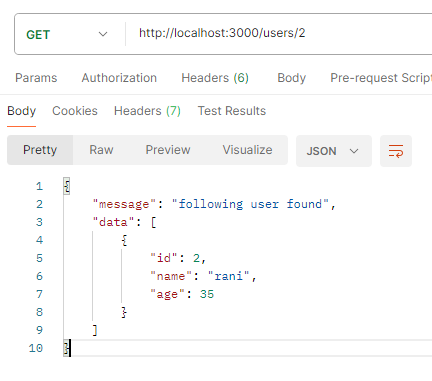
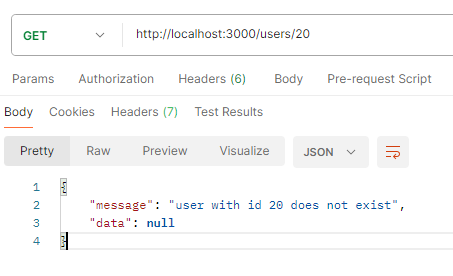
            data:null

         }

      )

   }

})

1. Creating api to search user based in his/her name
   1. We need to create search api : http://localhost:4000/users/search?name=raj
   2. Since we have query parameter in our url, we have to keep it before dynamic parameter (fetching user by id- here id is dynamic parameter because id can be 1234 or hello)

app.get('/users/search',(req,res)=>{

   const {name}=req.query

   const fetchedUsers=users.filter(user=>user.name.toLowerCase().includes(name.toLocaleLowerCase()))

   if(fetchedUsers.length!=0)

   {

      return res.status(200).send(

         {

            message:"following user found",

            data:fetchedUsers

         }

      )

   }

   else

   {

      return res.status(404).send(

         {

            message:`There are no users, with name ${name}`,

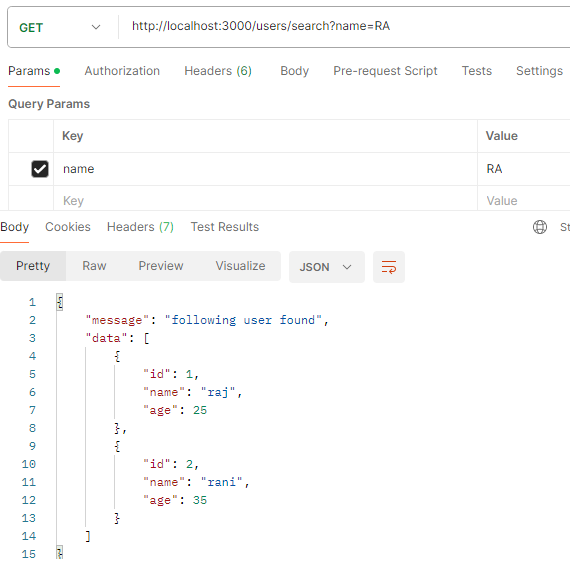
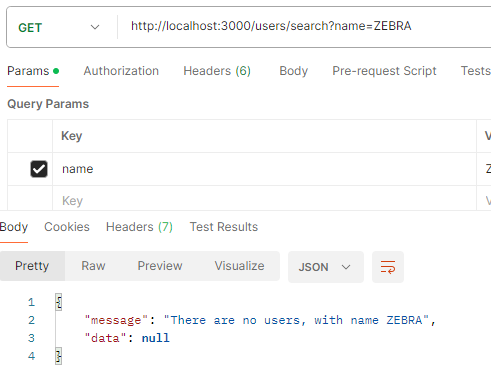
            data:null

         }

      )

   }

})

1. Creating post api
   1. Here we have to use express.json() middleware to read posted data. Since node doesn’t recognize req.body parameter

app.use(express.json())

app.post('/users',(req,res)=>{

   const {name,age}=req.body

   const user =

   {

      id:users.length+1,

      name,

      age

   }

   users.push(user)

   res.status(200).send(

      {

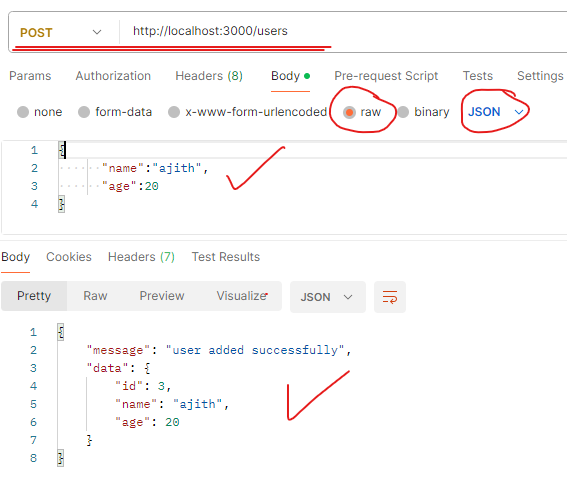
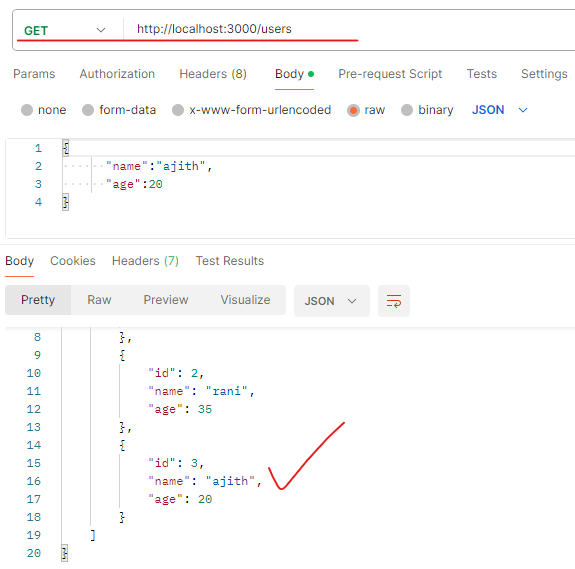
         message:`user added successfully`,

         data:user

      }

   )

})

1. Creating put api
   1. Code

app.put('/users/:id',(req,res)=>{

   const id=req.params.id

   const user = users.find(u=>u.id=== +id)

   if(user)

   {

      const {name,age}=req.body

      user.name = name

      user.age = age

      res.status(200).send(

         {

            message:`user with id ${id} updated with follwing details`,

            data:user

         })

   }

   else

   {

      res.status(404).send(

         {

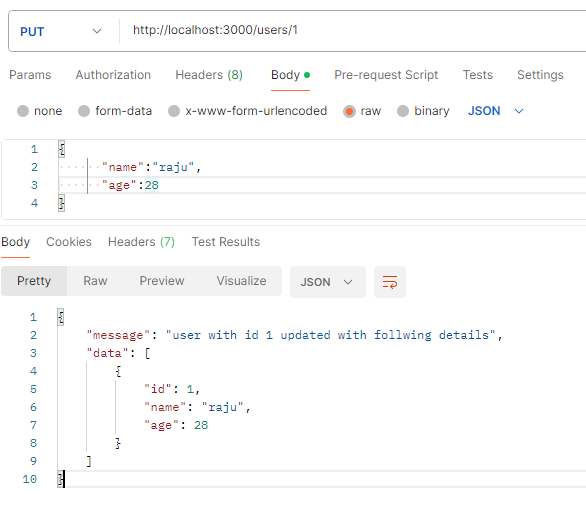
            message:`user with id ${id} doest not exists`,

            data:null

         })

   }

})



1. Creating patch api
   1. Code

app.patch('/users/:id',(req,res)=>{

   const id=req.params.id

   const user = users.find(u=>u.id=== +id)

   if(user)

   {

      const {name,age}=req.body

      if(name) {user.name = name}

      if(age) {user.age = age}

      res.status(200).send(

         {

            message:`user with id ${id} updated with follwing details`,

            data:user

         })

   }

   else

   {

      res.status(404).send(

         {

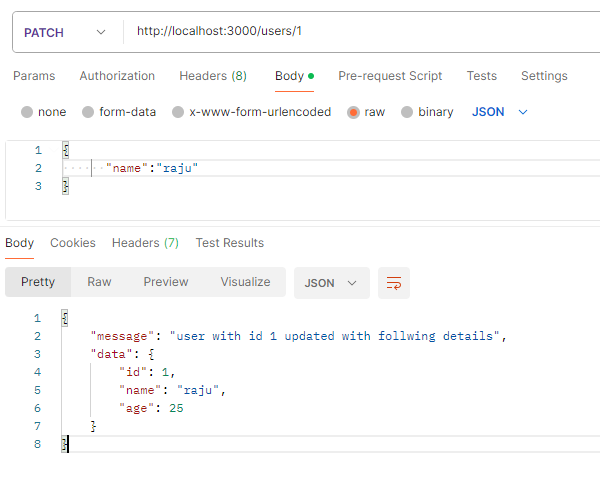
            message:`user with id ${id} doest not exists`,

            data:null

         })

   }

})



1. Creating delete api
   1. CODE

app.delete('/users/:id',(req,res)=>{

   const id=req.params.id

   const user = users.find(u=>u.id=== +id)

   if(user)

   {

      users = users.filter(u => u.id !== +id)

      res.status(200).send(

         {

            message:`user with id ${id} deleted`,

            data:null

         })

   }

   else

   {

      res.status(404).send(

         {

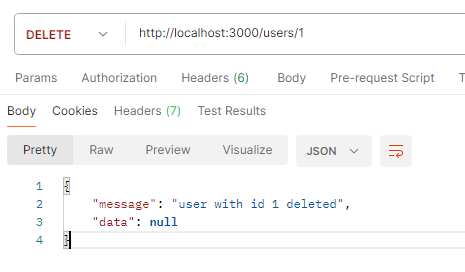
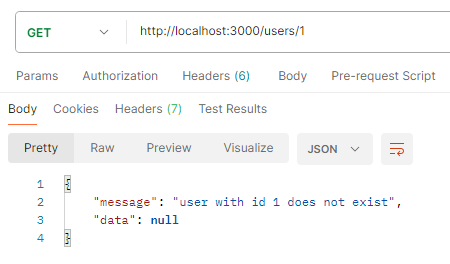
            message:`user with id ${id} doest not exists`,

            data:null

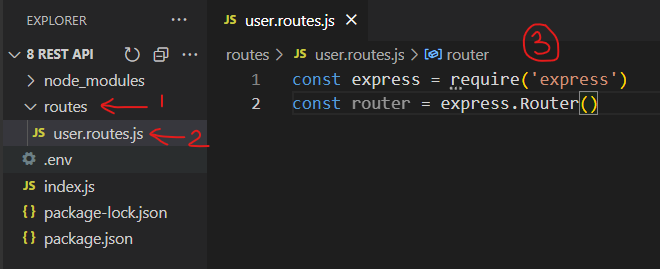
         })

   }

})

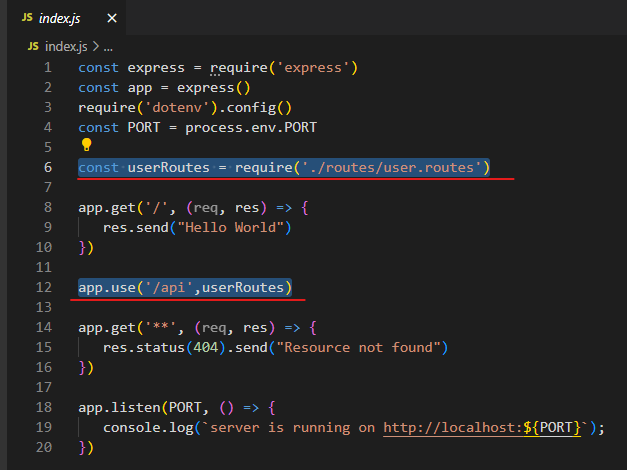
1. Creating separate file for each api resource
   1. Create routes folder
   2. Inside routes folder create user.routes.js
   3. Import Router() from express to manage the routes



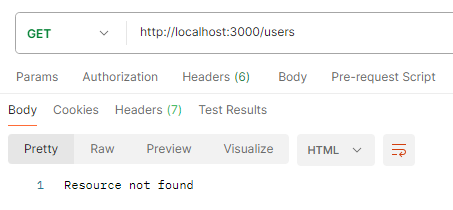
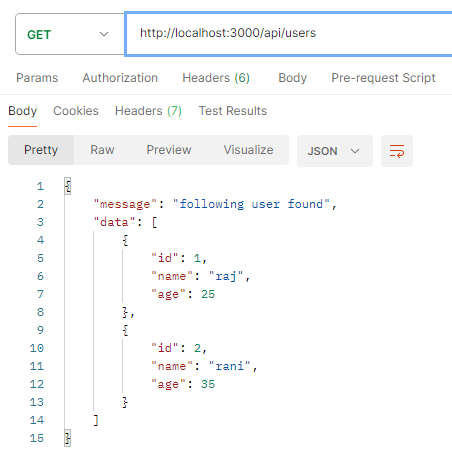
* 1. Add all user related apis from index.js to user.routes.js and change app.\* into router.\*
     1. Just change app to router and export it

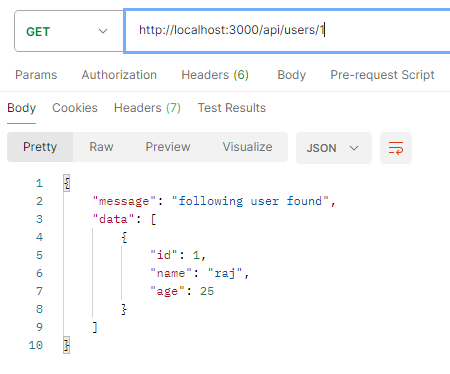
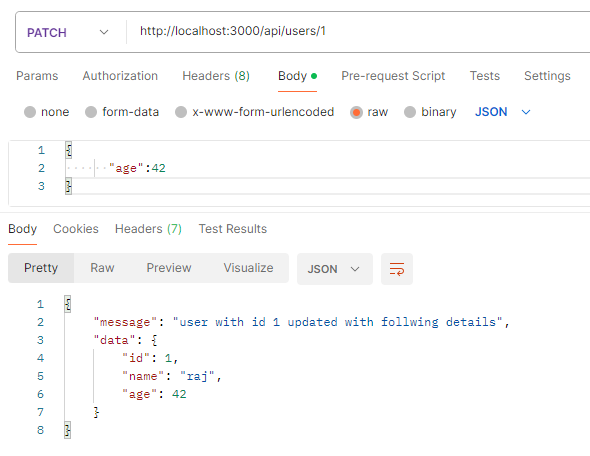


* 1. Import this file in index.js and use()



* 1. Now, our all users api will start from /api/users instead of /users

* 1. DONE !!!!!!!!!!!!!!!!!!

1. Sdasdas
2. Das
3. D
4. Asd
5. Sad
6. As
7. Ds
8. ada