Without middleware doing authentication

const express= require('express');

const app= express();

const port= 3000;

app.use(express.json());

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

  let  name=req.headers.username;

  let pass=req.headers.password;

  let id=req.query.kidneyId;

  if(name!="santhosh" || pass!="123"){

    res.status(400).json({message:"you are not fine"});

    return

  }

  if(id!=1&&id!=2){

    res.status(400).json({message:"wromg id"});

    return ;

  }

  res.json({message:"you are fine"});

});

app.listen(port, () => {

  console.log(`Server is running on http://localhost:${port}`);

} );

With middlewaresconst express = require('express');

const app = express();

const port = 3000;

app.use(express.json());

// Middleware to check username and password from headers

function userNameMiddleware(req, res, next) {

  const { username, password } = req.headers;

  if (username !== 'santhosh' || password !== '1234') {

    return res.status(400).json({ error: 'Wrong input: invalid credentials' });

  } else {

    next();

  }

}

// Middleware to check kidneyId from query parameters

function kidneyMiddleware(req, res, next) {

  const { kidneyId } = req.query;

  const id = Number(kidneyId); // convert to number

  if (id !== 1 && id !== 2) {

    return res.status(400).json({ error: 'Wrong input: invalid kidneyId' });

  } else {

    next();

  }

}

// Route with both middlewares

app.get('/check', userNameMiddleware, kidneyMiddleware, (req, res) => {

  res.send('Everything is fine!');

});

// Start the server

app.listen(port, () => {

  console.log(`Server is running at http://localhost:${port}`);

});

Average time calculation to get response using middleware assignment

const express = require('express');

const app = express();

const port = 3000;

app.use(express.json());

// Middleware to check username and password from headers

let totalTime = 0;

let requestHandled = 0;

function responseTimeMiddleware(req, res, next) {

  const start = Date.now();

  res.on('finish', () => {

    const duration = Date.now() - start;

    totalTime += duration;

    requestHandled++;

    const avgTime = totalTime / requestHandled;

    console.log(`Request took ${duration}ms, Avg: ${avgTime.toFixed(2)}ms`);

  });

  next();

}

// Route with both middlewares

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

  res.send('Everything is fine!');

});

// Start the server

app.listen(port, () => {

  console.log(`Server is running at http://localhost:${port}`);

});

Global catch

const express = require('express');

const app = express();

const port = 3000;

app.use(express.json());

app.post("/healthcheck", (req, res) => {

  const kidneys = req.body.kidneys;

  const kidneyLength = kidneys.length;

  res.send("your kidney length is " + kidneyLength + " cm and your kidneys are " + kidneys);

});

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

  console.error(err.stack);

  res.status(500).send('Something broke!');

});

app.listen(port, () => {

  console.log(`Example app listening on port ${port}`);

});

Zod

const { count } = require('console');

const express = require('express');

const app = express();

const z = require('zod');

const port = 3000;

const schema = z.array(z.number());  // expecting array of numbers

// another example for zod schema

// const schema2 = z.object({

//   name: z.string(),

//   age: z.number().int().positive(),

//   email: z.string().email(),

//   kidneys: z.array(z.object({

//     id: z.string(),

//     function: z.number().min(0).max(100),

//   })),

//   country: z.literal("USA").or(z.literal("IN")),

// });

app.use(express.json());

app.post("/healthcheck", (req, res) => {

  const kidneys = req.body.kidneys;

  if (!kidneys) {

    return res.status(400).json({ error: "Missing 'kidneys' in request body" });

  }

  const response = schema.safeParse(kidneys);

  if (!response.success) {

    return res.status(400).json({ error: response.error.errors });

  }

  res.send({ message: "Kidney data is valid", data: response.data });

});

app.listen(port, () => {

  console.log(`Example app listening on port ${port}`);

});

const zod=require("zod");

function validateInput(obj) {

  const schema =zod.object({

    name: zod.string().min(1).max(50),

    age: zod.number().min(1).max(100),

    email: zod.string().email().min(5).max(50),

    phone: zod.string().regex(/^\d{10}$/),

    address: zod.string().min(5).max(100),

    city: zod.string().min(1).max(50),

    state: zod.string().min(1).max(50),

    country: zod.string().min(1).max(50),

    zip: zod.string().regex(/^\d{5}$/),

    dob: zod.string().regex(/^\d{4}-\d{2}-\d{2}$/),

    password: zod.string().min(8).max(20).regex(/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)[a-zA-Z\d]/),

    confirmPassword: zod.string().min(8).max(20).regex(/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)[a-zA-Z\d]/),

})

const response = schema.safeParse(obj);

console.log(response);

if(!response.success) {

  console.log("Validation failed:", response.error.format());

  return;

}

};

validateInput({

  name: "John Doe",

  age: 30,

  email: "marimuthusanthoshh@gmail.com",

  phone: "1234567890",

  address: "123 Main St",

  city: "New York",

  state: "NY",

  country: "USA",

  zip: "12345",

  dob: "1990-01-01",

  password: "Password123",

  confirmPassword: "Password123",

})