**Dependencies:**

npm i express dotenv ejs multer cookie-parser express-validator mongoose jsonwebtoken bcrypt http-errors

in multer: upload.any() for all the form input field like text, image, etc. req.files and req.body we get data. When we will get an error inside uploader function by passing createError function. The error will go the default express error middleware handler. But we want to handle this error inside avatars upload function. So we call the next middleware upload.any()()(req, res, err) => {} now here we can handle the error.

**Express-validator:** using express validator method check() by chaining

//  ADD A USER

router.post("/", avatarUpload, [

  check("name").isLength({ min: 1 }),

  check("email").isEmail(),

]);

We know that we can run single middleware by separating a command each other. or we can pass array of middleware. So in above code we pass array of check() middleware. This is express middleware. Express handle this middleware. By chaining this check() middleware function we can validate each input field or anything.

**Note:** if we have 10 or more input field and If we validate like above way then our router will be more large. It looks like bad. So we can do one thing is that we can make array of this middleware in others file and import this middleware into here. Then our route looks like awesome and cleaner.

**Find data using or operation:**

We need to use $or operator with array. Inside array I can pass multiple query as object

  try {

    // find a user who has this email/username

    const user = await User.findOne({

      $of: [{ email: req.body.email }, { mobile: req.body.mobile }],

    });

  } catch (error) {}

Here I have passed two query as object. If one of them is meet up with the database then it return the user. Else if return the false.

**After using express-validator we receive the result inside the next middleware and we can handle this error and called the next process**

// Dependencies

const { check, validationResult } = require("express-validator");

const { unlink } = require("fs");

const path = require("path");

const createError = require("http-errors");

//Schema models imports

const User = require("../../models/UserModel");

// add user

const addUserValidator = [

  check("name")

    .isLength({ min: 1 })

    .withMessage("Name is required")

    .isAlpha("en-US", { ignore: " -" })

    .withMessage("Name must not contain other than alphabet")

    .trim(), //which method modify is called sanitization

  check("email")

    .isEmail()

    .withMessage("Invalid email address")

    .trim()

    .custom(async (value) => {

      try {

        const user = await User.findOne({ email: value });

        if (user) {

          throw createError("Email already is use");

        }

      } catch (error) {

        throw createError(error.message);

      }

    }),

  check("mobile")

    .isMobilePhone("bn-BD", {

      strictMode: true,

    })

    .withMessage("Mobile number must be bangladeshi")

    .custom(async (value) => {

      try {

        const user = await User.findOne({ mobile: value });

        if (user) {

          throw createError("Mobile already is use!");

        }

      } catch (error) {

        throw createError(error.message);

      }

    }),

  check("password")

    .isStrongPassword()

    .withMessage(

      "Password must be at least 8 characters long & should contain at least 1 lowecase, 1 uppercase, 1 number & 1 symbol"

    ),

];

// above middleware gives a result that we can receive into another function

const addUserValidatorHandler = function (req, res, next) {

  console.log("array of validation per korche");

  const errors = validationResult(req);

  const mappedErrors = errors.mapped();

  if (Object.keys(mappedErrors).length === 0) {

    console.log("kono error hoini clinet side a tai controller a jabe");

    next();

  } else {

    console.log("error hoice client side tai unlink korbe");

    // remove uploaded files

    if (req.files.length > 0) {

      const { filename } = req.files[0];

      unlink(

        path.join(\_\_dirname, `/../../public/uploads/avatars/${filename}`),

        (err) => {

          if (err) console.log("Error while unlink: ", err);

        }

      );

    }

    // response the errors

    res.status(500).json({ errors: mappedErrors });

  }

};

module.exports = {

  addUserValidator,

  addUserValidatorHandler,

};

If we use mappedErrors then it return a errors object with nice pattern.