Npm packages needed

1. Dotenv
2. Express
3. Mongoose
4. Multer

Dev dependencies

1. Nodemon

Scripts

Start : node app.js

Dev : nodemon app.js

Folders needed

1. Controllers
2. Models
3. Routes
4. Uploads

Files needed

1. App.js
2. .env

Steps

1. Npm init -y
2. Npm Install packages
3. Npm install -D packages

require('dotenv').config()

const PORT = process.env.PORT || 5000

const express = require('express')

const app = express()

app.get("/api/posts",  (req, res) => {

    res.send("hello")

})

app.listen(PORT, () => {

    console.log(`Server is running on port ${PORT}`)

} )

MOVE THE ROUTE TO THE ROUTES FOLDER

postRoute.js

const express = require('express')

const router = express.Router()

router.get('/api/posts', (req, res) => {

    res.send('helooo')

})

module.exports = router

UPDATE IN THE APP.JS

const routes = require('./routes/postRoute')

app.use('/', routes)

MOVE THE API INTO THE CONTROLLER

router.get('/api/posts', postController.indexPost)

const postController =  require('../controllers/postController')

postController.js

const indexPost = (req, res) => {

    res.send('helloo')

}

module.exports = {indexPost}

CONNECT TO DATABASE

Use the async and await with the try and catch

const dbConnection = async () => {

    try {

    const connected = await mongoose.connect(process.env.DB\_URI)

    console.log('Connected to Database')

    } catch (error) {

        console.log(error)

    }

}

dbConnection()

CREATE A MODEL THAT HELP YOU CREATE A DATABASE SCHEMA FOR INFORMATION IN YOUR DATABASE

Post.js

const mongoose = require('mongoose')

const postSchema = mongoose.Schema({

    title: String,

    catagory: String,

    content: String,

    image: String,

    created: {

        type: Date,

        default: Date.now()

    }

})

module.exports = mongoose.model('Post', postSchema)

Now go to your controller and access the data

 try {

        const posts = await Post.find({})

        res.status(200).json(posts)

    } catch (error) {

        res.status(404).json({

            message: error.message

        })

    }

And import sample data into your database

That the GET ALL REQUEST

POST REQUUEST

Insert these middlewares into your file

app.use(express.json())

app.use(express.urlencoded({ extended:true }))

app.use(express.static('uploads'))

Test the post route

const storePost = async (req, res) => {

    res.send("Hello")

}

Make sure you update the uploads routes

router.post('/api/posts', postController.storePost)

NOW POST DATA

How to handle images

const multer = require("multer")

// multer middleware

let storage = multer.diskStorage({

    destination: function(req, file, cb){

        cb(null, './uploads');

    },

    filename: function(req, file, cb){

        cb(null, file.fieldname+"\_"+Date.now()+"\_"+file.originalname)

    }

})

let upload = multer({

    storage: storage,

}).single("image")

router.post('/api/posts', upload, postController.storePost)

GET A POST

const showPost = async(req, res) => {

        const id  = req.params.id

        try {

            const post = await Post.findById(id)

            if(post){

                res.status(200).json(post)

            }else{

                res.status(404).json({

                    message: "Page not found"

                })

            }

        } catch (error) {

            res.status(404).json({

                message: error.message

            })

        }

    }

router.get('/api/posts/:id', postController.showPost)

PUT METHOD

const fs = require('fs')

const updatePost = async (req, res) => {

    const id = req.params.id

    let new\_image = ''

    if(req.file){

        new\_image  = req.file.filename

        try {

            fs.unlinkSync('./uploads/'+req.body.old\_image)

        } catch (error) {

            console.log(error);

        }

    }

    else{

        new\_image = req.body.old\_image

    }

    const newPost = req.body;

    newPost.image = new\_image

    try {

        await Post.findByIdAndUpdate(id, newPost)

        res.status(200).json({

            message: "Post updated successfully"

        })

    } catch (error) {

        res.status(404).json({

            message: error.message

        })

    }

}

router.put('/api/posts/:id', upload, postController.updatePost)

DELETE METHOD

router.delete('/api/posts/:id', upload, postController.destroyPost)

const destroyPost = async(req, res) => {

    const id = req.params.id

    try {

        const result = await Post.findByIdAndDelete(id)

        if(result.image !=  ''){

            try {

                fs.unlinkSync('./uploads/'+result.image);

            } catch (error) {

                console.log(error)

            }

        }

        res.status(200).json({

            message: "Successfully Deleted"

        })

    } catch (error) {

        res.status(404).json({

            message: error.message

        })

    }

}