* product.controller.js

```
export const addProduct = asyncHandler(async (req, res) => {
    const form = formidable({ multiples: true, keepExtensions: true });

    form.parse(req, async function (err, fields, files){
        if (err) {
            throw new CustomError(err.message || "Something went wrong", 500)
        }
        to store photo uniquely
        let productId = new Mongoose.Types.ObjectId().toHexString()

        console.log(fields, files);

        if (
            !fields.name ||
            !fields.price ||
            !fields.description ||
            !fields.collectionId
        ) {
            throw new CustomError("Please fill all the fields", 500)
        }
```

① Handle the error
② Extract the files from the fields
③ files in an Object so we will loop through this
④ while uploading the files we receive promise so we have to resolve that

3)

will return an array and we form an array out of it is because we need to loop through it

* Object.keys(files).map(async(file, index)=>{

inside each file there is a key which we want to extract

```
    const element = file[filekey]   // = file.filekey
    console.log(element)
```

sometimes to extract something from file

import fs from "fs"; to read file (fileSystem=fs)

```
const data = fs.readFileSync(element.filepath)
```

New Upload file

```
const Upload = await s3FileUpload({
    bucketName: config.S3_BUCKET_NAME,
    key: `product/${productID}/photo_${index+1}.png`,
    body: data,
    contentType: element.mimetype
})

return {secure_url: Upload.Location}
```

})

```
let imgArrayResp = Promise.all(
    Object.keys(files).map( async (file, index) => {
        const element = file[fileKey]
        console.log(element);
        const data = fs.readFileSync(element.filepath)

        const upload = s3FileUpload({
            bucketName: config.S3_BUCKET_NAME,
            key: `product/${productId}/photo_${index + 1}.png`,
            body: data,
            contentType: element.mimetype
        })

        // productId = 123abc456
        // 1: products/123abc456/photo_1.png
        // 2: products/123abc456/photo_2.png

        console.log(upload);
        return {
            secure_url : upload.Location
        }
    })
)
```

* let imgArray = await imgArrayResp

Sending to database

```
const Product = await Product.create ({
      _id: product_id,
    photos: imgArray,
    ... fields // destructuring fields (getting all its values fro

if (!Product) {
    throw new CustomError ("Failed to be created", 400)
}
res.status (200). json ({
    success: true,
    product
})
```

* We cannot extract Symbol (JS) datastructure using dot Notation
File[Filekey] = File.Filekey

*
```
    let imgArray = await imgArrayResp

    const product = await Product.create({
        _id: productId,
        photos: imgArray,
        ...fields
    })

    if (!product) {
        throw new CustomError("Product failed to be created in DB", 400)
    }
    res.status(200).json({
        success: Body.json(): Promise<any>
        product,
    })


    // TODO: we will take this tomorrow
})
```

* Just for extra surety so that file upload does not fail you can wrap entire form.parse() in try and catch once more

#Summary of file Upload
① Need third party library : formidable
② Create form
const form = formidable({multiples: true, keepExtensions: true})
Now we have access to files & fields
③ parse the form so apart from request we also have files & fields
form.parse(req, asyncFunction(err, fields, files){ }
i) Handle error
ii) create ProductId
iii) check if all fields are there ?

```
if (
    !fields.name ||
    !fields.price ||
    !fields.description ||
    !fields.collectionId
) {
    throw new CustomError("Please fill all the fields", 500)
}
```

④ Handle files
i) extract keys out of files by looping through them
ii) extract data using fs (file path)
⑤ Upload and Send it to database

```
// to get all products
export
* const getAllProducts = asyncHandler (async (req, res) => {
    const products = await Product.find ({})

    if (!products){
        throw new CustomError ("Nothing found", 404)
    }
    res.status (200).json ({
        success: true,
        products,
    })
})

// to get single Product
* export const getProductByID = asyncHandler (async (req, res) => {
    // we get ID from URL
    const {id: productID} = req.params
    const product = await Product.FindByID (productID)
    if (!product) {
        throw new CustomError ("Nothing found", 404)
    }
    res.status (200).json ({
        success: true,
        product,
    })
})
```

// to find products based on collection (collection_ID)

```
export const getProductByCollectionId = asyncHandler(async(req, res) => {
    const {id: collectionId} = req.params

    const products = await Product.find({collectionId})

    if (!products) {
        throw new CustomError("No products found", 404)
    }

    res.status(200).json({
        success: true,
        products
    })
})
```

// deleteProduct

```
export const deleteProduct = asyncHandler(async(req, res) => {
    const {id: productId} = req.params

    const product = await Product.findById(productId)
    // Check if product exists
    if (!product) {
        throw new CustomError("No product found", 404)

    }


    await product.remove() // we cannot use this as photos will not be deleted of the product
})
```

* So ① resolve promise
② loop through photos array ⇒ delete each photo
③ Grab the key and with that you can delete or perform any functionalities
(key: product._id)

```
const deletePhotos = Promise.all(
    product.photos.map(async( elem, index) => {
        await s3deleteFile({
            bucketName: config.S3_BUCKET_NAME,
            key: `products/${product._id.toString()}/photo_${index + 1}.png`
        })
    })
)

await deletePhotos;

await product.remove()

res.status(200).json({
    success: true,
    message: "Product has been deleted successfully"
})
```

// Controller to Update the Product

```
import { Router } from "express";
import { getProfile, login, logout, signUp } from "../controllers/auth.controller";
import { authorize, isLoggedIn } from "../middlewares/auth.middleware";
import AuthRoles from "../utils/authRoles";


const router = Router()

router.post("/signup", signUp)
router.post("/login", login)
router.get("/logout", logout)

router.get("/profile", isLoggedIn, authorize(AuthRoles.ADMIN), getProfile)
                       Middleware  Middle ware
export default router;
```

only if User is logged in can see profiles

Only admin can see profiles

* We create index.js in routes so that as we will be creating multiple files So we will just import this index.js single file and dont need to import multiple files (central file for all the exports)

```
import { Router } from "express";
import authRoutes from "./auth.route.js"

const router = Router()
router.use("/auth", authRoutes)


export default router
```

* We want now that all the access of the site should be as follows /auth / login
signup

* But we have to make our app aware of this setup

* So in app.js

```
import express from "express"
import cors from "cors"
import cookieParser from "cookie-parser"
import routes from "./routes/index.js"

const app = express()

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

app.use("/api/v1/", routes)

app.get("/", (_req, res) => {//just to check on home route
    res.send("Hello there hitesh - API")
})

app.all("*", (_req, res) => { // Handling wild card
    return res.status(404).json({ route ( route which is not defined by me
        success: false,
        message: "Route not found"
    })
})

export default app;

// http://localhost:4000/api/v1/auth/login
// http://localhost:4000/api/v1/auth/logout
```

Middle Ware should be used first

* _req : Syntax that req or _ is not being used here

* // Create a Controller so that Coupon can be added, deleted, edited, disabled

* Visit the file : Code > local > Download Zip