

eComm_class_live

1.configs/server.config.js

```
/**  
 * This file will contain the server configurations  
 */
```

```
module.exports = {  
  PORT : 8888  
}
```

2.configs/db.config.js

```
module.exports = {  
  DB_NAME : "ecomm_db",  
  DB_URL : "mongodb://localhost/ecomm_db" //0.0.0.0  
}
```

3.configs/auth.config.js

```
module.exports = {  
  secret : "This is my super private secret"  
}
```

MODELS

1.models/user.model.js

```
const mongoose = require("mongoose")  
const userSchema = new mongoose.Schema({  
  name : {  
    type : String,  
    required : true  
  },  
  userId : {  
    type : String,
```

```

        required : true,
        unique : true
    },
    password : {
        type : String,
        required : true
    },
    email : {
        type : String,
        required : true,
        lowercase : true,
        minLength : 10,
        unique : true
    },
    userType : {
        type : String,
        default : "CUSTOMER",
        enum : ["CUSTOMER", "ADMIN"]
    }
}, {timestamps : true, versionKey: false})

module.exports = mongoose.model("User", userSchema)

```

2. models/category.model.js

```

const mongoose = require("mongoose");
const categorySchema = new mongoose.Schema({
    name : {
        type : String,

```

```
      required : true,
      unique : true
    },
    description : {
      type : String,
      required : true
    }
  }, {timestamps : true , versionKey : false})
```

```
module.exports = mongoose.model("Category", categorySchema)
```

CONTROLLERS

```
1.controllers/auth.controller.js
```

```
/**
 * I need to write the controller / logic to register a user
 */
const bcrypt = require("bcryptjs")
const user_model = require("../models/user.model")
const jwt = require("jsonwebtoken")
const secret = require("../configs/auth.config")
const userModel = require("../models/user.model")
exports.signup = async (req, res)=>{
  /**
   * Logic to create the user
   */

  //1. Read the request body
  const request_body = req.body

  //2. Insert the data in the Users collection in MongoDB
```

```

const userObj = {
  name : request_body.name,
  userId : request_body.userId,
  email : request_body.email,
  userType : request_body.userType,
  password : bcrypt.hashSync(request_body.password,8)
}

try{

  const user_created = await user_model.create(userObj)
  /**
   * Return this user
   */

  const res_obj = {
    name : user_created.name,
    userId : user_created.userId,
    email : user_created.email,
    userType : user_created.userType,
    createdAt : user_created.createdAt,
    updatedAt : user_created.updatedAt
  }
  res.status(201).send(res_obj)

}catch(err){
  console.log("Error while registering the user", err)
  res.status(500).send({
    message : "Some error happened while registering the user"
  })
}

```

```

    })
  }
  //3. Return the response back to the user
}

exports.signin = async (req, res)=>{

  //Check if the user id is present in the system
  const user = await user_model.findOne({userId : req.body.userId})
  if(user == null){
    return res.status(400).send({
      message : "User id passed is not a valid user id"
    })
  }

  //Password is correct
  const isPasswordValid = bcrypt.compareSync(req.body.password, user.password)
  if(!isPasswordValid){
    return res.status(401).send({
      message : 'Wrong password passed'
    })
  }

  // using jwt we will create the acces token with a given TTL and return
  const token = jwt.sign({id : user.userId}, secret.secret,{
    expiresIn : 120
  })
  res.status(200).send({
    name : user.name,
    userId : user.userId,
    email : user.email,

```

```
    userType : user.userType,  
    accessToken : token  
  })  
}
```

2. controllers/category.controller.js

```
const category_model = require("../models/category.model")
```

```
/**  
 * Controller for creating the category  
 *  
 * POST localhost:8080/ecommm/api/v1/categories  
 *  
 * {  
   "name" : "Household",  
   "description" : "This will have all the household items "  
 }  
 */
```

```
exports.createNewCategory = async (req, res)=>{
```

```
  //Read the req body
```

```
  //Create the category object
```

```
  const cat_data = {  
    name : req.body.name,  
    description : req.body.description  
  }
```

```
  try{
```

```
    //Insert into mongodb
```

```
    const category = await category_model.create(cat_data)
```

```
    return res.status(201).send(category)
```

```

}catch(err){
    console.log("Error while creating the category", err)
    return res.status(500).send({
        message : "Error while creating the category"
    })
}

```

```

//return the response of the created category

```

```

}

```

Middlewares

middlewares/auth.mw.js

```

const user_model = require("../models/user.model")

```

```

const jwt = require("jsonwebtoken")

```

```

const auth_config = require("../configs/auth.config")

```

```

/**

```

```

 * Create a mw will check if the request body is proper and correct

```

```

 */

```

```

const verifySignUpBody = async (req, res, next)=>{

```

```

    try{

```

```

        //Check for the name

```

```

        if(!req.body.name){

```

```

            return res.status(400).send({

```

```

                message : "Failed ! Name was not provided in request body"

```

```

            })

```

```
}
```

```
//check for the email
```

```
if(!req.body.email){
```

```
    return res.status(400).send({
```

```
        message : "Failed ! Email was not provided in request body"
```

```
    })
```

```
}
```

```
//check for the userId
```

```
if(!req.body.userId){
```

```
    return res.status(400).send({
```

```
        message : "Failed ! userId was not provided in request body"
```

```
    })
```

```
}
```

```
//Check if the user with the same userId is already present
```

```
const user = await user_model.findOne({userId : req.body.userId})
```

```
if(user){
```

```
    return res.status(400).send({
```

```
        message : "Failed ! user with same userId is already present"
```

```
    })
```

```
}
```

```
next()
```

```
}catch(err){
```

```
    console.log("Error while validating the request object", err)
```

```
    res.status(500).send({
```



```
        message : "Error while validating the request body"
    })
}
}
```

```
const verifySignInBody = async (req, res, next)=>{
```

```
    if(!req.body.userId){
        return res.status(400).send({
            message : "userId is not provided"
        })
    }
    if(!req.body.password){
        return res.status(400).send({
            message : "password is not provided"
        })
    }
    next()
}
```

```
const verifyToken = (req , res, next)=>{
    //Check if the token is present in the header
    const token = req.headers['x-access-token']

    if(!token){
        return res.status(403).send({
            message : "No token found : Unauthorized"
        })
    }
}
```

```

//If it's the valid token
jwt.verify(token,auth_config.secret ,async (err, decoded)=>{
  if(err){
    return res.status(401).send({
      message : "Unauthorized !"
    })
  }
  const user = await user_model.findOne({userId : decoded.id})
  if(!user){
    return res.status(400).send({
      message : "Unauthorized, this user for this token doesn't exist"
    })
  }
  //Set the user info in the req body
  req.user = user
  next()
})
//Then move to the next step
}

```

```

const isAdmin = (req, res, next) => {
  const user = req.user
  if(user && user.userType == "ADMIN"){
    next()
  }else{
    return res.status(403).send({
      message : "Only ADMIN users are allowed to access this endpoint"
    })
  }
}

```

```

    }
}

module.exports = {
  verifySignUpBody : verifySignUpBody,
  verifySignInBody : verifySignInBody,
  verifyToken : verifyToken,
  isAdmin : isAdmin
}

```

ROUTES

routes/auth.routes.js

```

/**
 * POST localhost:8888/ecommm/api/v1/auth/signup
 *
 * I need to intercept this
 */

const authController = require("../controllers/auth.controller")
const authMW = require("../middlewares/auth.mw")

module.exports = (app)=>{
  app.post("/ecommm/api/v1/auth/signup",[authMW.verifySignUpBody], authController.signup)

  /**
   * route for
   * POST localhost:8888/ecommm/api/v1/auth/signin
   */
  app.post("/ecommm/api/v1/auth/signin",[authMW.verifySignInBody],authController.signin )
}

```

```
}
```

2.routes/category.routes.js

```
/**
```

```
 * POST localhost:8080/ecommm/api/v1/categories
```

```
 */
```

```
const authMw = require("../middlewares/auth.mw")
```

```
category_controller = require("../controllers/category.controller")
```

```
auth_mw = require("../middlewares/auth.mw")
```

```
module.exports = (app)=>{
```

```
  app.post("/ecommm/api/v1/categories",[auth_mw.verifyToken,  
  authMw.isAdmin],category_controller.createNewCategory)
```

```
}
```

server.js

```
/**
```

```
 * This will be the starting file of the project
```

```
 */
```

```
const express = require("express")
```

```
const mongoose = require("mongoose")
```

```
const app = express()
```

```
const server_config = require("../configs/server.config")
```

```
const db_config = require("../configs/db.config")
```

```
const user_model = require("../models/user.model")
```

```
const bcrypt = require("bcryptjs")
```

```
app.use(express.json())
```

```

/**
 * Create an admin user at the starting of the application
 * if not already present
 */
//Connection with mongodb
mongoose.connect(db_config.DB_URL)
const db = mongoose.connection
db.on("error" , ()=>{
  console.log('Error while connecting to the mongoDB')
})
db.once("open", ()=>{
  console.log("Connected to MongoDB")
  init()
})

async function init(){
  try{
    let user = await user_model.findOne({userId : "admin"})

    if(user){
      console.log("Admin is already present")
      return
    }

  }catch(err){
    console.log("Error while reading the data", err)
  }
  try{
    user = await user_model.create({

```

```

        name : "Vishwa",
        userId : "admin",
        email : "kankvish@gmail.com",
        userType : "ADMIN",
        password : bcrypt.hashSync("Welcome1",8)
    })
    console.log("Admin created ", user)
} catch(err){
    console.log("Error while create admin", err)
}
}
/**
 * Stich the route to the server
 */
require("./routes/auth.routes")(app)
require("./routes/category.routes")(app)
/**
 * Start the server
 */
app.listen(server_config.PORT, ()=>{
    console.log("Server started at port num : ", server_config.PORT)
})

```