**Model define for study Notion**

**User Model ➖**

FirstName = {string, required,trim }

lastName = {string,trim,required}

Email = {string , required , unique,,trim}

Password ={string, required maxlength:20 minLength:8}

Role = {String, [Student, Admin , Instructor, required }

active={boolean, default-true} (this is used for user , user set active or inactive by admin)

approved:{boolean, default-true} (this is used for instructor , account approved or not by admin)

Image = {string, required }

Token = {string}

resetPasswordExpires = {date}

additionDetails = { connect with profile model object id, ref- profile , required }

Courses = {connect with courses model object id , ref - course}

CourseProgress = {connect with courseProgress model object id , ref-courseProgress}

And also add time stamp

**Profile Model ➖**

Gender = {string,}

About = {string,trim}

DateOfbirth = {string}

contactNumber = {number , trim}

**OTP Model ➖**

Email = {string, required}

Otp = {string,required}

createdAt = {date , default - date.now(), expires = 60\*5 } (The document will be automatically deleted after 5 minutes of its creation time )

**Course Model ➖**

courseName = {string, required, trim}

courseDescription = {string, required, trim}

Price = {number, requires}

whatYouWillLearn = {string}

Thumbnail = {string}

Instructor = {connect with user model object id, ref-user, required}

Category = {connect with category model object id , ref-category }

Status = {String, [draft , published} (this is used for course is publish in website or not )

createdAt= { type:Date , default - date.now()}

courseContent = [in array {connect with section model object id, ref:Section}]

ratingAndReviews = [ in array {connect with ratingAndreview model object id , ref- ratingAndReviews}]

studentEnrolled = [ in array {connect with user model object id , ref:user, required}]

instruction = [in array {string}] (this is used for set the instruction of course)

Tag = [in array {string}] (this is used for se the tag of the course )

**Section Model ➖**

sectionName = {string, required}

subSection = [ In array {connect with subSection model object id , ref-subsection , required}]

**Sub Section Model ➖**

Title = {string}

timeDuration = {string}

Description = {string}

videoUrl = {string}

**CourseProgress Model ➖**

courseId = {connect with course model object id , ref- course}

userId = {connect with user model object id , ref- user}

Completed Video = [in array {connect with subSection model object id , ref-subsection}]

**RatingAndReviews Model ➖**

User = {connect with user model object id , ref- user}

Rating = {number, required}

Review = {String , required }

Course = {connect with user model object id , ref-course,index-true}

**Category Model ➖**

Name = {string , trim, required}

Description = {string}

Course = [in array {connect with course model object id , ref-course }]

**Controllers (logic )**

1. create model of schema

-> user

-> profile

-> courses

-> courseProgress

-> category

-> review

-> otp

-> Section

-> subsection

-> invoices

2. initialise the npm (npm init)

3. install package & libraries that are used in project

-**> Express**

**-> mongoose**

**-> cors**

**-> Jsonwebtoken**

**-> Bcrypt**

**-> cookie-parser**

**-> otp-generator**

**-> fileupload**

**-> Cloudinary**

**-> Dotenv**

**-> nodemailer**

**-> nodemon**

4. make folder like models, routes , middlewares, utils , config, controllers

5. make mongodb atlas file and make cluster

6. connect database from mongoose

========================================================================

=>>>>>>> THIS IS CONFIG FUNCTION FOR CONNECT TO DATABASE

-----------------------------------------------

1. import mongoose from mongoose library

2. create an async function name is ConnectDB

3. write code for connection with db in try catch block

4. using await in mongoose having one method is connect method and using this code

code:

await mongoose.connect(dbUrl)

5. then export connectDB function

==================================================================================

=>>>>> THIS IS UTILITY FUNCTION FOR USING SEND MAIL

---------------------------------------------------------------------------------

1. send mail with otp before data save in database for verify your account using (pre save middleware) method in OTP model

In Register

USER ====> Data User ===> mail with otp ===> verify OTP ==> registered successfully

HOW TO SEND MAIL

-----------------------------

1. create a async function having argument like title , body , subject and also all code in try catch block

2. create a transporter by using nodemailer with createTransport

3. host name = which service is use (like : smtp.gmail.com)

4. auth :{

user : email of owner

pass : password of app from gmail

}

5. create info variable store

6. make sendMail function by using transporter

let info = tansporter.sendMail({

from : 'coder Camp || Md Ajhar Alam',

to : `${email}`,

subject :`${subject}`,

body:`${body}`

})

=========================================================================

==> \*\*\*\*\*\*\*\*\*\* THIS IS UTILITY FUNCTION FOR UPLOAD IMAGE ON CLOUDINARY

--------------------------------------------------------------------------------------------------------

1. import Cloudinary from Cloudinary

2. make a async function name UploadImageToCloudinary having parameters (file ,height, quality , folder)

3. make a try catch block

4. make options variable having contain folder parameter in object

5. if height is available then set height in options.height = height

6. if quality is available the set quality in options.quality = quality

7. also set options.resource\_type = 'auto' automatic check which type of file send

8. then call cloundinary.v2.uploader.upload function send parameter ('file.tempFilePath,options)

9. then send res for success message

10. if error then send res in catch with error message

=========================================================================

==> \*\*\*\* BEFORE USER REGISTER AN ACCOUNT SEND AN OTP ON USER MAIL

=============================================

\*\*\*\*

IN OTP MODELS SEND VERIFICATION MAIL FUNCTION

----------------------------------------------------------------------------

I. steps:-

---------

1. making a async function , name is sendVerificationMail having two argument like

(email , otp)

2. all code in try catch block

3. in that function call sendMail function from utils and send arguments like

(email , 'Email Verification',otp)

II. Steps:-

----------------

1. using OTPScheme and also use pre (save) middleware in that argument pass next()

2. in this call sendVerificationMail and pass two argument , like

(this.email , this.otp)

==================================================================================

\*\*\*\* Auth Controllers :-

====================>>>>>>>>

1. Otp generate API

---------------------

i. create an async function having name sendOtp

ii. find email from req body

iii. check email is valid or not then send response (error or success)

iv. check in user model if email is present or not

v. if email is present then send error message ('user account already exists')

vi. generate otp by otp-generator library and otp store in otp variable

code:

let otp = otpgenerator.generate({

upperCaseAlphabets:false,

lowerCaseAlphabets:false,

specialCharacter:false

})

vii. check otp in OTP model , otp present or not and store in result variable

viii. always generate unique otp by using this code

while(result){

otp = otpgenerator.generate({

upperCaseAlphabets:false,

lowerCaseAlphabets:false,

specialCharacter:false

})

result = await OTP.findOne({otp:otp})

}

ix. create otpPayload having {email,otp}

x. then save the otpPayload in otp model

xi. then send success response

-----------------------------------------------------------------------------------------------

2. createAccount API

------------------

i. find all {firstName,lastName,email,password,confirmPassword,role,otp} from req body

ii. check all if any missing then send error response

iii. check user in USER model by findOne method using email

iv. if user is present or not if user is present then send error response

v. check password or confirm password is equal or not if not then send error response

vi. find recent otp from OTP model by using this code

code:

const recentOtp = OTP.findOne({otp:otp}).sort({createdAt:-1}).limit(1)

vii. then check recent otp length if length is 0 then send error response

vii. check recent Otp and otp both are equal if not equal then send error response

viii. create profile in profile model all value of null

ix. then hash password by using Bcrypt library

x. then create all value save in user model with profile.\_id in userprofile

xi. in image set this link

image:`https://api.dicebear.com/5.x/initials/svg/seed=${firstName} ${lastName}`

xii. then send success response

-----------------------------------------------------------------------------------------------

3. loginAccount API

------------------------

i. find all {email,password} from req body

ii. check all if any missing then send error response

iii. check user in USER model by findOne method using email

then populate('additional Detail')

iv. if user is present or not if user is not present then send error response

v. check password and user password are equal or not by using bcrypt library compare

method

vi. if matchPassword is not then send error response

vii. create token payload having id:user\_id, email:user.email, role:user.role

viii. create jwt token by using jwt.sign method with secret\_key and with payload and

also set expiresIn:'1d'

ix. then set password is undefined

x. create cookies option payload having expires: 3\*24\*60\*60\*100 , httpOnly:true

x. set token in cookies having token option with cookies payload with send success

response login is Successfull

-----------------------------------------------------------------------------------------------

4. changePassword API

------------------------------

i. find all {oldPassword,newPassword,confirmPassword} from req body

ii. check all if one missing then send error response

iii. find the userId from the req.user.id

iv. check in User model user is found or not by userId

v. if user not found then send error response

vi. check password and confirm password is equal or not if not then send error response

vii. check oldPassword is equal to user.password by using bcrypt library using compare

method.

viii. update the new password and confirmPassword in User Model by using

findOneAndUpdate and use this code

code:

await User.findOneAndUpdate(

{\_id:userId},

{password:newPassword, confirmPassword:newPassword},

{new:true})

ix. then send success response to user

==================================================================================

\*\*\*\* resetPassword controllers:-

===================================>>>>>

1. resetPasswordToken API

----------------------

i. find all {email} from req body

ii. check all if one is missing then send error response

iii. check in User Model user is present or not

iv. if user is not present then send error response

v. 1.create a token by using crypto using this code

code:

const token = await crypto.randomUUID()

vi. update the token and resetPasswordExpires in User model and by using this code

code :

const updateDetails = await User.findOneAndUpdate(

{email:email},

{token:token, resetPasswordExpires:Date.now() + 5\*60\*1000},

{new:true}

)

vi. create frontend url and also send token with link

vii. send verification mail to user email by using this

await mailSender(email,'Password Reset Link',link)

viii. then send success response to user

-----------------------------------------------------------------------------------------------

2. resetPassword API

------------------

i. find all {token, newPassword, confirmPassword} from req body

ii. check all if one is missing then send error response

iii. check newPassword or confirmPassword are equal or not

iv. if password not equal then send error response

v. check in User Model user present or not by using findOne through token

vi. if user not found then send error response

vii. check resetPasswordExpires is expire or not by using this code

code:

if(resetPasswordExpires < date.now()){

then send error response to user reset link is expires

}

viii. then hash newPassword from Bcrypt library using hash method

ix. update the password in User Model by using findOneAndUpdate metho through token

x. in update method update password and confirm password

xi. then send success response to user

==================================================================================

Category Controllers:-

====================>>>>>

1. createCategory API

------------------

i. find all {name,description} for req body

ii. check if one is missing then send error response

ii. check in Category Model category name is already exist or not

iv. if exists then send error response

v. then save the data in Category Model

vi. then send success message to user

-----------------------------------------------------------------------------------------------

2. findAllCategory API

-------------------

i. find all category in Category Model using find method with parameter name:true and

description:true

ii. then send success message

-----------------------------------------------------------------------------------------------

3. findCategoryById API

---------------------

i. find id from req.parmas.id or req.body

ii. check id is true or not

iii. if not true then send error response

iv. then find category in Category Model by using findById method through id

v. if category not found then send error response

vi. then send success response to user

-----------------------------------------------------------------------------------------------

4. updateCatgoryDetailbyId API

---------------------------

i. find id from req.params or req.body

ii. find all {name, description} from req body

iii. check if one is missing then send error response

iv. check id is valid or not

v. if not valid then send error response

vi. then update the new data in Category Model by using findOneAndUpdate method with

parameter like id , name , description and also set new should be true

vii. then send success response to user

-----------------------------------------------------------------------------------------------

5. deleteCategoryById API

----------------------------

i. find id from req.params.id or req.body

ii. check id is valid or not

iii. if id is not valid then send error response

iv. then delete this category by using findOneAndDelete method through id

v. then send success response to user

==================================================================================

Course Controllers:-

===================>>>>>

1. createCourse API

-----------------

i. find all const{courseName,courseDescription,whatWillYouLearn,price,category,status, tag:\_tag,instruction: \_instruction} = req.body

‘ ii. find thumbnail or image from req.file.thumnailImage

iii. Convert tag into javascript object by using this code

Code:

Const tag = JSON.parse(\_tag) {// Parsing the JSON string into a JavaScript

object}

Iv. convert instruction into javascript object by using this code:

Code:

Const tag = JSON.parse(\_tag) {// Parsing the JSON string into a JavaScript

object}

v. check if one is missing and also check !tag.length and !instruction.length then send error response

Vi. check if status is empty or undefined then set status should be draft

Vii. find Userid from req.user.id

viii find a instructor detail from User Model

ix. check instructor detail exist or not by using this code

Code :

Const instructor = await user.findById(userId,{role:Instructor})

ix. if not exists then send error response

vii. find categoryDetail in Category Model by using findById method with category

vii. if not found then send error response

viii. upload image in Cloundinary by using this code:

code:

await uploadImageToCloudinary(thumbnail,process.env.FOLDER\_NAME)

ix. save the data in course model by using create method

x. update and push the course id in user model of instructor by using this code

code:

await User.findByIdAndUpdate(

{\_id:instructorDetail.id},

{

$push:{

courses:newCourse.id

}

},

{new:true}

)

xi. update the course detail in category schema by using this code

code:

await User.findByIdAndUpdate(

{\_id:category},

{

$push:{

courses:newCourse.id

}

},

{new:true}

)

xii. then send success response to user

----------------------------------------------------------------------------------------------------------

2. getAllCourses API

------------------

i. find all course from course model using find method having parameters:

And also check {status:published}

courseName:true,

courseDescription:true,

price:true,

thumbnail:true,

instructor:true,

ratingAndReviews:true,

studentEnrolled:true

ii. and populate the instructor detail and also execute

ii. then send success response