**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

==> \*\*\*\*\*\*\*\*\*\* THIS IS UTILITY FUNCTION FOR CONVERT TOTAL DURATION INTO SECOND

1. Make a function name is converSecondToDuration having parameter totalSecond
2. Make a varibale hours and find hour , minutes , second by using this code

unction convertSecondsToDuration(totalSeconds) {

const hours = Math.floor(totalSeconds / 3600);

const minutes = Math.floor((totalSeconds % 3600) / 60);

const seconds = Math.floor((totalSeconds % 3600) % 60);

if (hours > 0) {

return `${hours}h ${minutes}m`;

} else if (minutes > 0) {

return `${minutes}m ${seconds}s`;

} else {

return `${seconds}s`;

}

}

Export default convertSecondsToDuration,

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

==> \*\*\*\* 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

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

3. getCourseDetailByID API

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

1. Find course id from req body
2. Check course id not exits then send error response
3. Find the course detail form Course Model
4. If course found then populate this detail by using this code

findOne(courseId).populate({path:”Instructor”, populate:{path:”additionalDetail”},

}).populate(“category”).populate(“ratingAndReviews”).populate(

{

Path:”courseContent”,

Populate:{

Path:”subSection”,

Select:”-videoUrl”

},

}).exec().

1. If course detail not found then send error response
2. Let create totatl duration of course by using this code

Let totatCourseDuration = 0;

courseDetails.courseContent.foreach((content)=>{

content.subSection.foreach((subSection)=>{

const timeDurationInSeconde = parseInt(subsSection.timeDuration)

totalCourseDuration += timeDurationInSecond

})

})

1. Make a variable (TotalDuration) store and call utility function {convertSecondToDuration) and send totalDuration
2. Send success message to user and send also courseDetails , totalDuration

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

4. getFullCourseDetail API with course progress

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

1. Find courseId from req body
2. If courseId not exists then send error response
3. Find userId from req.user.id
4. Find courseDetails form Course Model
5. If course found then populate this details using this code

findOne(courseId).populate({path:”Instructor”, populate:{path:”additionalDetail”},

}).populate(“category”).populate(“ratingAndReviews”).populate(

{

Path:”courseContent”,

Populate:{

Path:”subSection”,

Select:”-videoUrl”

},

}).exec().

1. Calculate courseProgressCount from CourseProgress Model by using this code

Let courseProgressCount = await CourseProgress.findOne({

courseId: courseId,

userId: userId

})

Consle.log(“courseProgressCount”,courseProgressCountn)

1. If course details not found then send error response
2. Let create totatl duration of course by using this code

Let totatCourseDuration = 0;

courseDetails.courseContent.foreach((content)=>{

content.subSection.foreach((subSection)=>{

const timeDurationInSeconde = parseInt(subsSection.timeDuration)

totalCourseDuration += timeDurationInSecond

})

})

1. Make a variable (TotalDuration) store and call utility function {convertSecondToDuration) and send totalDuration
2. Send success reponse to user and send this detail by this code

return res.status(200).json({

success: true,

data: {

courseDetails,

totalDuration,

completedVideos: courseProgressCount?.completedVideos

? courseProgressCount?.completedVideos

: [],

},

});

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

5. getInstructorCourse API

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

1. Find instructorId from req.user.id
2. Find instructorCourse from Course Model
3. If instrunctorCourse not found then send erro response
4. If instructor course found then populate this data to user by this code

Let instructorCourse = await Course.find({

Instructor:instructorId,

}).sort({createdAt:-1}).populate({path:”courseContent”, populate:{path:”subSection”},

}).exec()

1. Then send success resposne to user with instructorCourse

------------------------------------------------------------------------------------------------------------------------------------------------6. deleteCourse API

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

1. Find courseId from req.body
2. Find course by courseId from Course Model
3. If course not found then send error response
4. Check studentEnrolled and find studentId from courseenrolled by loop method using this code

Const studenEnrolled = course.studentEnrolled

For(const studentId of StudentEnrolled){

await User.findByIdAndUpdate({studentId,{$pull:{courses:courseId},

})

}

1. Check courseSection and find sectionId from courseContent by loop method using this code

Const courseSections = course.courseContent

For(const sectionId of courseSections){

Const section = await Section.findById(sectionId)

If(section){

Const subSections = section.subSection

For(const subSectionId of subSections){

await SubSection.findByIdAndDelete(subSectionId)

}

}

Await Section.findByIdAndDelete(sectionId)

}

Await Course.findByIdAndDelete(courseId)

1. Then send success response to user

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

7. updateCourse API

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

1. Find courseId for req.body
2. Make variable (updates) of req.body
3. Find course detail from Course Model by findById
4. Update course image by using this code

if (req.files) {

console.log("thumbnail update");

const thumbnail = req.files.thumbnailImage;

const thumbnailImage = await uploadImageToCloudinary(

thumbnail,

process.env.FOLDER\_NAME

);

course.thumbnail = thumbnailImage.secure\_url;

}

for (const key in updates) {

if (updates.hasOwnProperty(key)) {

if (key === "tag" || key === "instructions") {

course[key] = JSON.parse(updates[key]);

} else {

course[key] = updates[key];

}

}

}

await course.save();

const updatedCourse = await Course.findOne({

\_id: courseId,

})

.populate({

path: "instructor",

populate: {

path: "additionalDetails",

},

})

.populate("category")

.populate("ratingAndReviews")

.populate({

path: "courseContent",

populate: {

path: "subSection",

},

})

.exec();

res.json({

success: true,

message: "Course updated successfully",

data: updatedCourse,

});

1. Then send success message to user

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

\*\*\*\*\*\* **Category Controller**

===========🡺>>>>>>

1. CreateCategory API

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

1. Find all {name, description} from req body
2. Check if one is missing then send error response
3. Then save category in Category Model by using create method
4. Send success message to user

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

1. showAllCategories API

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

1. find all categories in Category Model by using find() method
2. populate the course
3. check the status of course has been published or not by usinf this code

const categoriesWithPublishedCourses = allcategories.filter((category)=>

category.courses.some((course)=> course.status === “Published”)

)

1. then send success response and also send category with published data

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

1. categoryPageDetails API

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

Make getRandomInt function having find random number by this code

function getRandomInt(max) {

return Math.floor(Math.random() \* max);

}

1. find category id from req body
2. if category id not found then send error response
3. find selectcategory by categoryId from Category Model using findById
4. then populate this data by this code

.populate({

Path:”courses”,

Match:{status:”Published”},

Populate:”ratingAndReviews”

}).exec()

1. if selectcategory not found then send error response
2. if selectCategory length is 0 the send error response (no course found for the select category)

after this code

1. const categoriesExceptSelected = await Category.find({

\_id: { $ne: categoryId },

});

let differentCategory = await Category.findOne(

categoriesExceptSelected[getRandomInt(categoriesExceptSelected.length)]

.\_id

)

.populate({

path: "courses",

match: { status: "Published" },

})

.exec();

console.log();

const allCategories = await Category.find()

.populate({

path: "courses",

match: { status: "Published" },

})

.exec();

const allCourses = allCategories.flatMap((category) => category.courses);

const mostSellingCourses = allCourses

.sort((a, b) => b.sold - a.sold)

.slice(0, 10);

res.status(200).json({

success: true,

data: {

selectedCategory,

differentCategory,

mostSellingCourses,

},

});

1. send success message to user and this data { selectedCategory, differentCategory , mostSellingCourses }

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

\*\*\*\*\*\*\* **CourseProgress Controller**

===============🡺>>>>>>

1. UpdateCourseCategory API

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

1. Find {courseId, subSectionId} from req body

ii. find userId from req.user.id

iii. find subSection detail from sub Section Model by subSectionId

iv if not found subSection detail then send error response

v find the couseProgress Detail from courseProgress Model by CourseId, userId

vi. if courseProgress Detail not found then send error response

vii. else when in courseProgress having completedVideo that includes subSectionId

viii. then send error message subsection already completed

by helping this code

if (!courseProgress) {

return res.status(404).json({

success: false,

message: "Course progress Does Not Exist",

});

} else {

if (courseProgress.completedVideos.includes(subsectionId)) {

return res.status(400).json({

error: "Subsection already completed",

});

}

courseProgress.completedVideos.push(subsectionId);

}

1. Then save the courseProgress data
2. The send success message to user (course progress updated

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\* SubSection Controller

============🡺>

1. createSubSection API

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

1. Find {sectionId , title , description } from req body
2. Find video from req.files.video
3. If one is missing then send error response
4. The upload the video cloudinary import from utils folder by helping this code

const uploadDetails = await uploadImageToCloudinary(

video,

process.env.FOLDER\_NAME

);

console.log(uploadDetails);

1. Save the data in subsection model by using create method help by this code

const SubSectionDetails = await SubSection.create({

title: title,

timeDuration: `${uploadDetails.duration}`,

description: description,

videoUrl: uploadDetails.secure\_url,

});

1. Update subsection id in section Model and populate subSection by using this code

const updatedSection = await Section.findByIdAndUpdate(

{ \_id: sectionId },

{ $push: { subSection: SubSectionDetails.\_id } },

{ new: true }

).populate("subSection");

1. Send success message to user with update section data

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

1. updateSubSection API

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

1. find all {subsectionId, sectionId, title, description} from req body
2. find subsectionDetail form SubSection Model by subsectionId
3. if subsection detail not found then send error response
4. if title is undefined then set the title of subsection with this title

if(title !==undefined){

subSection.title = title

}

1. if description is undefined then set the description of subsection with this description above code
2. if video and req.files is undefined then upload the video in cloudinary by using this code

if (req.files && req.files.video !== undefined) {

const video = req.files.video;

const uploadDetails = await uploadImageToCloudinary(

video,

process.env.FOLDER\_NAME

);

subSection.videoUrl = uploadDetails.secure\_url;

subSection.timeDuration = `${uploadDetails.duration}`;

}

1. save the subsection data
2. update section detail in section Model and poulate the subsection detail by using this code

const updateSection = await Section.findByID(sectionId).populate(‘subSection’);

1. send success response to user with update Section data

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

1. DeleteSubSection API

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

1. Find {sectionId, subSectionId } from req body
2. Remove the subsection id from section Model by using this code

await Section.findByIdAndUpdate(

{ \_id: sectionId },

{

$pull: { // pull means remove

subSection: subSectionId,

},

}

);

1. Find the subSection detail from Subsection Model by subSectionId
2. If subSection not found then send error response
3. Update the section detail in Section Model and populate subsection by using this code

const updatedSection = await Section.findById(sectionId).populate(

"subSection"

);

1. Send success response to user with updated Section

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

\*\*\*\*\*\* Section Controller

=======🡺>

1. createSection API

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

1. find {sectionName, courseId} from req body
2. if one is missing then send error response
3. save the section detail in Section Model by using create Method
4. update section detail in course model and populate courseContent with subsection by

using this code

const updatedCourse = await Course.findByIdAndUpdate(

courseId,

{

$push: {

courseContent: newSection.\_id,

},

},

{ new: true }

)

.populate({

path: "courseContent",

populate: {

path: "subSection",

},

})

.exec();

1. send success response to user with update course

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

1. updateSection API

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