**SIMPLE CRUD**

1. **Model**

const mongoose =require('mongoose');

const reviewSchema = new mongoose.Schema({

title:{

       type:String,

       trim:true,

       required:[true,'Please enter the title'],

       maxlength:100

},

text:{

       type:String,

       trim:true,

       required:[true,'Please enter the text'],

},

rating:{

       type:Number,

       required:[true,'Please enter the Rating'],

       min:1,

       max:10

},

createdAt:{

    type:Date,

    default:Date.now()

}

});

module.exports=mongoose.model('Review',reviewSchema);

1. **Controller**

const ErrorResponse= require('../util/errorResponse')

const asyncHandler =require('../middleware/async')

const Review=require('../model/Review');

//create

exports.Create=asyncHandler(async(req,res,next)=>{

        const CreateReview=await Review.create(req.body);

        res.status(200).json({success:true,data:CreateReview})

});

//Read All

exports.Read=asyncHandler(async(req,res,next)=>{

        const ReadReview=await Review.find();

        res.status(200).json({success:true,count:ReadReview.length,data:ReadReview})

});

//Read One

exports.ReadOne=asyncHandler(async(req,res,next)=>{

        const ReadSingleReview=await Review.findById(req.params.id);

        res.status(200).json({success:true,data:ReadSingleReview})

});

//Update One

exports.Update=asyncHandler(async(req,res,next)=>{

        const UpdateSingleReview=await Review.findByIdAndUpdate(req.params.id,req.body,{

        new:true,

        runValidators:true

    });

        return res.status(200).json({success:true,data:UpdateSingleReview})

});

//Delete One

exports.Delete=asyncHandler(async(req,res,next)=>{

        const DeleteSingleReview=await Review.findByIdAndDelete(req.params.id);

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

});

1. **Router**

const express = require('express');

const Review=require('../model/Review');

const {

    Create,

    Read,

    ReadOne,

    Update,

    Delete}= require('../controller/Review');

const router= express.Router();

router

.route('/')

.post(Create)

.get(Read);

router

.route('/:id')

.get(ReadOne)

.put(Update)

.delete(Delete);

module.exports=router;

1. **Server**

const express =require('express');

const dotenv =require('dotenv');

dotenv.config({path:'./config/config.env'});

const morgan= require('morgan');

const connectDB = require('./config/db')

const errorHandler= require('./middleware/error');

const Review=require('./router/Review');

connectDB();

const app=express();

if(process.env.NODE\_ENV==='DEVELOPMENT')

{

    app.use(morgan('dev'))

}

app.use(express.json())

app.use('/api/Review',Review);

app.use(errorHandler);

const port=process.env.PORT || 5000;

const server=app.listen(port,console.log(`server running in

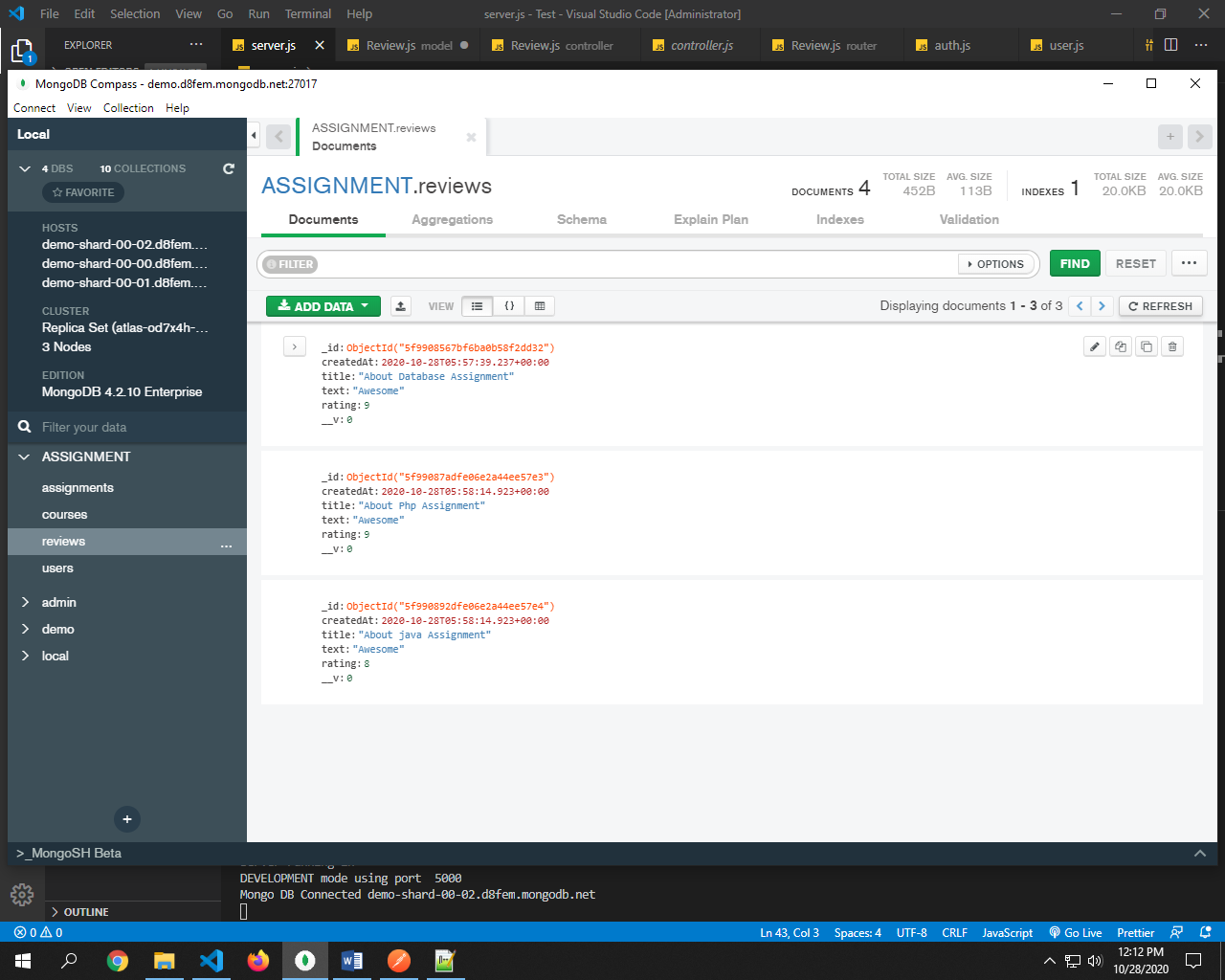
${process.env.NODE\_ENV} mode using port  ${port}`));

process.on('unhandledRejection',(err,promise)=>{

    console.log(`Error : ${err.message}`);

    server.close(()=>process.exit(1));

})



**CRUD WITH FOREIGN KEY AND ACCESS USER ROLE**

1. **Model**

**Add below code in the above model**

assignment:{

    type:mongoose.Schema.ObjectId,

    ref:'assignment',

    required:true

},

user:{

    type:mongoose.Schema.ObjectId,

    ref:'user',

    required:true

},

// only one student can give onE review

reviewSchema.index({assignment:1,user:1}, {unique:true});

1. **Controller**

**Controller for create a post with login as a student only with particular assignment**

const Assignment=require('../model/assignment');

//create

exports.Create=asyncHandler(async(req,res,next)=>{

    req.body.assignment=req.params.assignmentId;

    req.body.user=req.user.id;

    const assignment=await Assignment.findById(req.params.assignmentId);

    if(!assignment)

    {

        return next(new ErrorResponse(

            `No assignment with id of ${req.params.assignmentId}`,404));

    }

    const CreateReview=await Review.create(req.body);

    res.status(201).json({success:true,data:CreateReview})

});

**Controller for read all post or particular assignment**

//Read All

exports.Read=asyncHandler(async(req,res,next)=>{

});

**Controller for read single review or review according to particular assignment**

//Read One

exports.ReadOne=asyncHandler(async(req,res,next)=>{

        const ReadSingleReview=await (await Review.findById(req.params.id)).populate({

            path:'assignment',

            select:'assignmentHeading'

        });

        if(!ReadSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        res.status(200).json({success:true,data:ReadSingleReview})

});

**Controller for update single review**

//Update One

exports.Update=asyncHandler(async(req,res,next)=>{

        let UpdateSingleReview=await Review.findById(req.params.id);

        if(!UpdateSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        //make sure its student and admin

        if(UpdateSingleReview.user.toString()!== req.user.id && req.user.role!=='admin')

        {

        return next(new ErrorResponse(

            `Not authorize to updtae Review `,401));

        }

        UpdateSingleReview=await Review.findByIdAndUpdate(req.params.id,req.body,{

        new:true,

        runValidators:true

    });

        return res.status(200).json({success:true,data:UpdateSingleReview})

});

**Controller for delete single review**

//Delete One

exports.Delete=asyncHandler(async(req,res,next)=>{

    let DeleteSingleReview=await Review.findById(req.params.id);

        if(!DeleteSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        //make sure its student and admin

        if(DeleteSingleReview.user.toString()!== req.user.id && req.user.role!=='admin')

        {

        return next(new ErrorResponse(

            `Not authorize to delete Review `,401));

        }

        await DeleteSingleReview.remove();

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

});

**Complete code**

const ErrorResponse= require('../util/errorResponse')

const asyncHandler =require('../middleware/async')

const Review=require('../model/Review');

const Assignment=require('../model/assignment');

//create

exports.Create=asyncHandler(async(req,res,next)=>{

    req.body.assignment=req.params.assignmentId;

    req.body.user=req.user.id;

    const assignment=await Assignment.findById(req.params.assignmentId);

    if(!assignment)

    {

        return next(new ErrorResponse(

            `No assignment with id of ${req.params.assignmentId}`,404));

    }

    const CreateReview=await Review.create(req.body);

    res.status(201).json({success:true,data:CreateReview})

});

//Read All

exports.Read=asyncHandler(async(req,res,next)=>{

    if(req.params.assignmentId)

    {

        const ReadReview=await Review.find({assignment:req.params.assignmentId});

        res.status(200).json({success:true,count:ReadReview.length,data:ReadReview})

    }

    else{

        res.status(200).json(res.advancedResult)

    }

});

//Read One

exports.ReadOne=asyncHandler(async(req,res,next)=>{

        const ReadSingleReview=await (await Review.findById(req.params.id)).populate({

            path:'assignment',

            select:'assignmentHeading'

        });

        if(!ReadSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        res.status(200).json({success:true,data:ReadSingleReview})

});

//Update One

exports.Update=asyncHandler(async(req,res,next)=>{

        let UpdateSingleReview=await Review.findById(req.params.id);

        if(!UpdateSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        //make sure its student and admin

        if(UpdateSingleReview.user.toString()!== req.user.id && req.user.role!=='admin')

        {

        return next(new ErrorResponse(

            `Not authorize to updtae Review `,401));

        }

        UpdateSingleReview=await Review.findByIdAndUpdate(req.params.id,req.body,{

        new:true,

        runValidators:true

    });

        return res.status(200).json({success:true,data:UpdateSingleReview})

});

//Delete One

exports.Delete=asyncHandler(async(req,res,next)=>{

    let DeleteSingleReview=await Review.findById(req.params.id);

        if(!DeleteSingleReview)

        {

        return next(new ErrorResponse(

            `No Review with id of ${req.params.id}`,404));

        }

        //make sure its student and admin

        if(DeleteSingleReview.user.toString()!== req.user.id && req.user.role!=='admin')

        {

        return next(new ErrorResponse(

            `Not authorize to delete Review `,401));

        }

        await DeleteSingleReview.remove();

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

});

1. **Router**

**Replace rout with protect and authorize function**

const router= express.Router({mergeParams: true});

const advanceResult= require('../middleware/advancedResult');

const{protect,authorize}= require('../middleware/auth')

router

.route('/')

.post(protect,authorize('student','admin'),Create)

.get(Read);

**Rout get with advance result**

router

.route('/')

.post(protect,authorize('student','admin'),Create)

.get(advancedResult(Review,{

    path:'assignment',

    select:['assignmentHeading']

}),Read);

**Give link to assignment also for path setting**

const reviewsRouter= require('./review');

router.use('/:assignmentId/review',reviewsRouter)

**complete code**

const express = require('express');

const Review=require('../model/Review');

const {

    Create,

    Read,

    ReadOne,

    Update,

    Delete}= require('../controller/Review');

const router= express.Router({mergeParams: true});

const advancedResult= require('../middleware/advancedResult');

const{protect,authorize}= require('../middleware/auth')

router

.route('/')

.post(protect,authorize('student','admin'),Create)

.get(advancedResult(Review,{

    path:'assignment',

    select:['assignmentHeading']

}),Read);

router

.route('/:id')

.get(ReadOne)

.put(protect,authorize('student','admin'),Update)

.delete(protect,authorize('student','admin'),Delete);

module.exports=router;

1. **Server**

**Complete**

const path= require('path');

const express =require('express');

const dotenv =require('dotenv');

const morgan= require('morgan');

const fileupload = require('express-fileupload');

const cookieParser= require('cookie-parser');

const route=require('./router/router');

const courses=require('./router/courses');

const Review=require('./router/Review');

const auth=require('./router/auth');

const connectDB = require('./config/db')

const errorHandler= require('./middleware/error');

dotenv.config({path:'./config/config.env'});

connectDB();

const app=express();

if(process.env.NODE\_ENV==='DEVELOPMENT')

{

    app.use(morgan('dev'))

}

//File upload

app.use(fileupload());

//static folder

app.use(express.static(path.join(\_\_dirname,'public')));

app.use(express.json())

app.use(cookieParser())

app.use('/api/assignment',route);

app.use('/api/courses',courses);

app.use('/api/Review',Review);

app.use('/api/auth',auth);

app.use(errorHandler);

const port=process.env.PORT || 5000;

const server=app.listen(port,console.log(`server running in

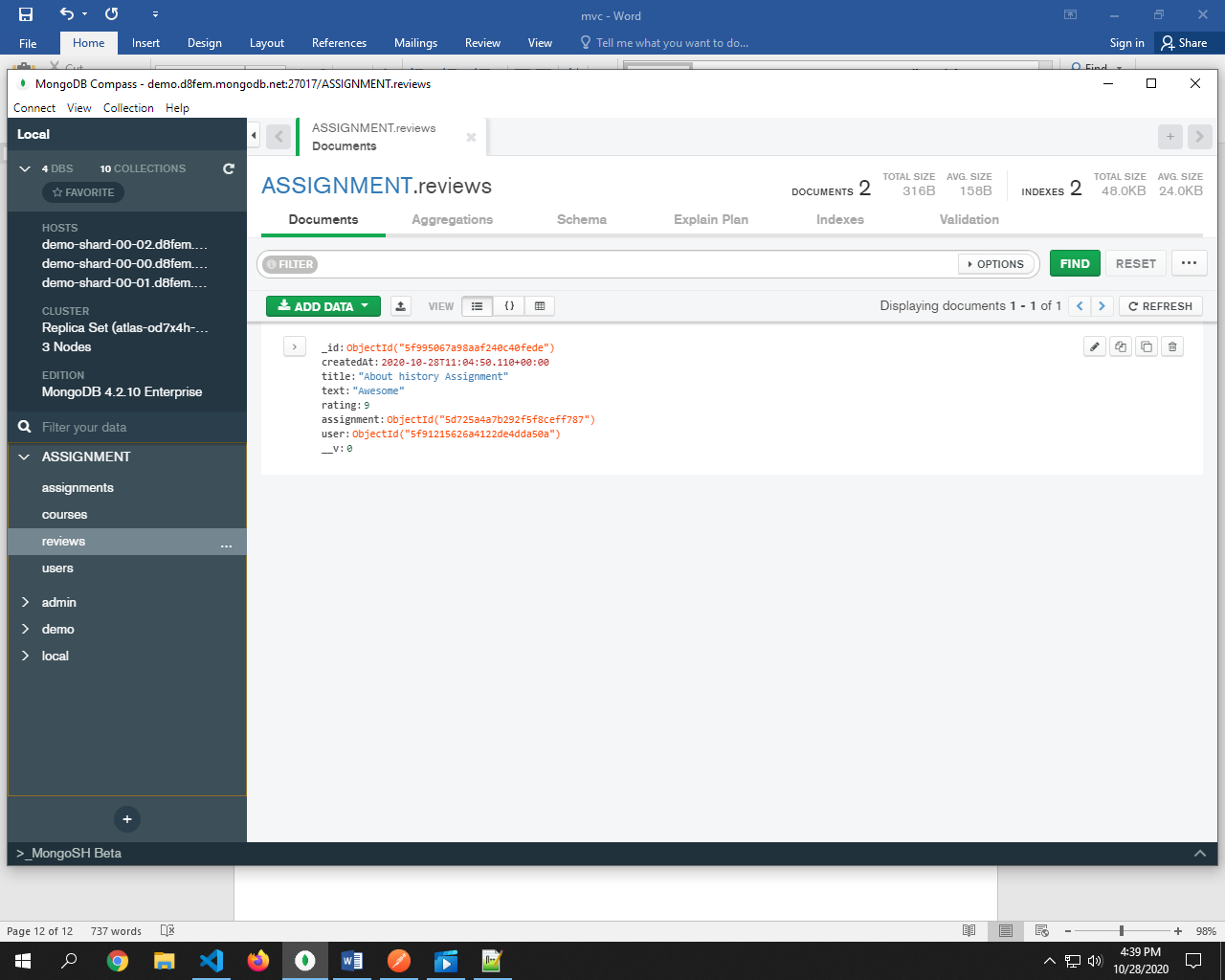
${process.env.NODE\_ENV} mode using port  ${port}`));

process.on('unhandledRejection',(err,promise)=>{

    console.log(`Error : ${err.message}`);

    server.close(()=>process.exit(1));

})



// @desc        Read Student Profile

// @route       GET api/student/studentProfile

exports.readStudentprofile = asyncHandler(async(req,res,next)=>{

    const student=await studentProfile.find().populate({path:'studentAuthentication'});

    res.status(201).json({success:true,count:student.length,data:student})

});

// @desc        Read Single Student Profile

// @route       GET api/student/studentProfile

exports.readSingleStudentprofile = asyncHandler(async(req,res,next)=>{

    const studentId=await studentAuthentication.findById(req.studentAuthentication.id);

    if(!studentId)

    {

        return next(new ErrorResponse(

            `No Student Registered with id of ${req.studentAuthentication.id}`,404));

    }

    const student=await studentProfile.findOne(req.studentAuthentication.id).populate({path:'studentAuthentication'});

    res.status(201).json({success:true,count:student.length,data:student})

});