Text Instruction : Module 12

GitHub Link:   
  
<https://github.com/Apollo-Level2-Web-Dev/Level2-Batch4-PH-University-Server/tree/part-2>

Requirement Analysis:   
  
<https://docs.google.com/document/d/10mkjS8boCQzW4xpsESyzwCCLJcM3hvLghyD_TeXPBx0/edit?usp=sharing>

12-1 Avoid Repetition of Try-Catch , use catchAsync

12-2 Implement Your Army Middleware

utils > catchAsync.ts :

import { NextFunction, Request, RequestHandler, Response } from "express";

const catchAsync = (fn: RequestHandler) => {

    return (req: Request, res: Response, next: NextFunction) => {

        Promise.resolve(fn(req, res, next)).catch((err) => next(err));

    };

};

export default catchAsync;

user.route.ts:

import express, { NextFunction, Request, Response } from 'express';

import { UserControllers } from './user.controller';

const router = express.Router();

const shenaBahini = (req: Request, res: Response, next: NextFunction) => {

    // console.log('I am a Shenabahini');

    console.log(req.body);

    next();

}

router.post('/create-student', shenaBahini, UserControllers.createStudent);

export const UserRoutes = router;

12-3 Implement validateRequest Middleware

student.validation.ts:

export const studentValidationSchemas = {

    studentValidationSchema,

};

middlewares > validateRequest.ts:

import { NextFunction, Request, Response } from "express";

import { AnyZodObject } from "zod";

const validateRequest = (schema: AnyZodObject) => {

    return async (req: Request, res: Response, next: NextFunction) => {

        try {

            await schema.parseAsync({

                body: req.body,

            });

            return next();

        } catch (error) {

            next(error);

        }

    };

};

export default validateRequest ;

12-4 Create Academic Semester Interface

**1. Using String Literal Union**

This is the most common and precise way to define a type for months:

type Month =

| "January"

| "February"

| "March"

| "April"

| "May"

| "June"

| "July"

| "August"

| "September"

| "October"

| "November"

| "December";

const currentMonth: Month = "January"; // ✅ Valid

const invalidMonth: Month = "Jan"; // ❌ Error: Type '"Jan"' is not assignable to type 'Month'

academicSemester.interface.ts:

type Month =

  | "January"

  | "February"

  | "March"

  | "April"

  | "May"

  | "June"

  | "July"

  | "August"

  | "September"

  | "October"

  | "November"

  | "December";

export type AcademicSemester = {

    name: 'Autumn' | 'Summar' | 'Fall',

    code: '01'| '02' | '03',

    year: Date,

    startMonth: Month ;

    endMonth: Month ;

}

12-5 Create Academic Semester Model

import { model, Schema } from "mongoose";

import { TAcademicSemester, TAcademicSemesterCode, TAcademicSemesterName, TMonths } from "./academicSemester.interface";

const Months: TMonths[] = [

    "January",

    "February",

    "March",

    "April",

    "May",

    "June",

    "July",

    "August",

    "September",

    "October",

    "November",

    "December",

];

const AcademicSemesterName: TAcademicSemesterName[] = ['Autumn', 'Summar', 'Fall'];

const AcademicSemesterCode: TAcademicSemesterCode[] = ['01', '02', '03'];

const academicSemesterSchema = new Schema<TAcademicSemester>({

    name: {

        type: String,

        required: true,

        enum: AcademicSemesterName,

    },

    year: {

        type: Date,

        required: true,

    },

    code: {

        type: String,

        required: true,

        enum: AcademicSemesterCode,

    },

    startMonth: {

        type: String,

        required: true,

        enum: Months,

    },

    endMonth: {

        type: String,

        required: true,

        enum: Months,

    },

},

    {

        timestamps: true,

    },

);

export const AcademicSemester = model<TAcademicSemester>(

    'AcademicSemester',

    academicSemesterSchema

)

12-6 Create Academic Semester validation, route and controller

12-7 Create Academic Semester Service

ok

12-8 Handle Logical Validation of Academic Semester