Text Instructions: Module 9

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

**9-2 How to do custom validation**

How to capitalize a string in js

**9-3 How to validate using validator and Joi package**

**Doc:** [**https://joi.dev/api/?v=17.13.3**](https://joi.dev/api/?v=17.13.3)

Google > validator npm

* <https://www.npmjs.com/package/validator>

npm i validator

* <https://www.npmjs.com/package/@types/validator>

npm i –D @types/validator

google > joi

* <https://www.npmjs.com/package/joi>

npm i joi

Chatgpt >

import { Schema, model } from 'mongoose';

import { Guardian, LocalGuardian, Student, UserName } from './student/student.interface';

import validator from 'validator';

const userNameSchema = new Schema<UserName>({

firstName: {

type: String,

required: [true, 'First Name is required'],

trim: true,

maxlength: [20, 'First Name can not be more than 20 characters'],

validate: {

validator: function (value: string) {

const firstNameStr = value.charAt(0).toUpperCase() + value.slice(1);

console.log(value);

console.log(firstNameStr);

return firstNameStr === value;

},

message: '{VALUE} is not in capitalize format',

},

},

middleName: {

type: String,

trim: true,

},

lastName: {

type: String,

required: [true, 'Last Name is required'],

trim: true,

validate: {

validator: (value: string) => validator.isAlpha(value),

message: '{VALUE} is not valid',

}

},

});

const guardianSchema = new Schema<Guardian>({

fatherName: {

type: String,

required: [true, 'Father\'s Name is required'],

},

fatherOccupation: {

type: String,

required: [true, 'Father\'s Occupation is required'],

},

fatherContactNo: {

type: String,

required: [true, 'Father\'s Contact Number is required'],

},

motherName: {

type: String,

required: [true, 'Mother\'s Name is required'],

},

motherOccupation: {

type: String,

required: [true, 'Mother\'s Occupation is required'],

},

motherContactNo: {

type: String,

required: [true, 'Mother\'s Contact Number is required'],

},

});

const localGuardianSchema = new Schema<LocalGuardian>({

name: {

type: String,

required: [true, 'Local Guardian\'s Name is required'],

},

occupation: {

type: String,

required: [true, 'Local Guardian\'s Occupation is required'],

},

contactNo: {

type: String,

required: [true, 'Local Guardian\'s Contact Number is required'],

},

address: {

type: String,

required: [true, 'Local Guardian\'s Address is required'],

},

});

const studentSchema = new Schema<Student>({

id: {

type: String,

required: [true, 'Student ID is required'],

unique: true,

},

name: {

type: userNameSchema,

required: [true, 'Student Name is required'],

},

gender: {

type: String,

enum: {

values: ['male', 'female', 'other'],

message: '{VALUE} is not a valid gender. Allowed values: male, female, other.',

},

required: [true, 'Gender is required'],

},

dateOfBirth: {

type: String,

required: [true, 'Date of Birth is required'],

},

email: {

type: String,

required: [true, 'Email is required'],

// match: [/.+@.+\..+/, 'Please enter a valid email address'],

unique: true,

validate: {

validator: (value: string) => validator.isEmail(value),

message: '{VALUE} is not valid email type '

}

},

contactNo: {

type: String,

required: [true, 'Contact Number is required'],

},

emergencyContactNo: {

type: String,

required: [true, 'Emergency Contact Number is required'],

},

bloodGroup: {

type: String,

enum: {

values: ["A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-"],

message: '{VALUE} is not a valid blood group.',

},

},

presentAddress: {

type: String,

required: [true, 'Present Address is required'],

},

permanentAddress: {

type: String,

required: [true, 'Permanent Address is required'],

},

guardian: {

type: guardianSchema,

required: [true, 'Guardian information is required'],

},

localGuardian: {

type: localGuardianSchema,

required: [true, 'Local Guardian information is required'],

},

profileImg: {

type: String,

},

isActive: {

type: String,

enum: {

values: ['active', 'blocked'],

message: '{VALUE} is not a valid status. Allowed values: active, blocked.',

},

default: 'active',

},

});

export const StudentModel = model<Student>('Student', studentSchema);

==> make a joi shema by following this model

[ Chatgpt ]

**9-4 How to validate using Joi**

make a joi shema by student model

<https://joi.dev/api/?v=17.13.3>

**9-5 How to validate using zod**

Google > zod > doc

<https://zod.dev/>

Doc > installation > npm install zod

Doc > basic Usage

Doc > object

Chatgpt : this is my model (student.model.ts ). please create a zod validation schema

**9-6 Implement a custom instance method**

Google > mongoose Doc >

<https://mongoosejs.com/docs/typescript/statics-and-methods.html>