**Example:**

const mg = require("mongoose");

**mg.connect**("mongodb://127.0.0.1:27017/lju1")

  .then(() => { console.log("success"); })

  .catch((err) => { console.error(err); });

const personSchema = new **mg.Schema**({

  name: String,

  age: Number,

  active: Boolean

});

const Person = **mg.model**("Person", personSchema);

const **updatePerson** = **async (name, update)** => {

  try {

    const result = await Person.**updateOne**({ name }, update,{upsert:true});

    console.log('Update Result:', result);

  } catch (err) { console.error('Error updating person:', err); }

};

const **updatePersonById** = **async (id, update)** => {

  try {

    const updatedPerson = await Person.**findByIdAndUpdate**(id, update, { new: true, upsert: true });

    console.log('Updated Person:', updatedPerson);

  } catch (err) { console.error('Error updating person:', err); }

};

**const deletePersonById = async (id) => {**

  try {

    const deletedPerson = await Person.**findByIdAndDelete**(id);

    if (deletedPerson) {console.log('Deleted Person:', deletedPerson); }

    else {console.log('Person not found');}

  } catch (err) {console.error('Error deleting person by ID:', err);}

};

**updatePersonById**("66c0fd3453eced83156cb23d", { name:"LJU" ,age: 28, active: false });

**updatePerson**("test", { age: 34, branch:"CE",active: false });

**deletePersonById**("66c0fd3453eced83156cb23c");

### **Example**

**You are developing a MongoDB-based application using Mongoose. You need to define a userSchema that includes various validation rules to ensure data integrity and consistency.**

1. **Define a Mongoose schema called userSchema with the following fields and validation requirements:**
   * **username:**
     + **Required and must be between 4 and 20 characters long.**
     + **Must start with letters and end with digits.**
     + **Should be trimmed of any leading or trailing spaces.**
     + **Should be converted to uppercase before saving.**
   * **email:**
     + **Required, must be unique across the collection.**
     + **Must follow the standard email format.**
   * **age:**
     + **Must be a number between 18 and 65.**
   * **role:**
     + **Must be either 'user' or 'admin'.**
     + **Should default to 'user' if not provided.**

const mg = require('mongoose');

**mg.connect**("mongodb://127.0.0.1:27017/validations").then(()=>{console.log("success")}).catch((err)=>{console.error(err)});

const userSchema = new **mg.Schema**({

  username: {

    type: String,

    required: [true, 'Username is required'], **// Custom error message**

    minlength: [4, 'Username must be at least 4 characters long'],

    maxlength: [20, 'Username cannot be more than 20 characters long'],

    match:[/^[A-Za-z]+[0-9]+$/,"Must start with letters and end with digits"],

    trim:true,

    uppercase:true

  },

  email: {

    type: String,

    unique: [true, 'email already exists'],

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

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

  },

  age: {

    type: Number,

    min: [18, 'Age must be at least 18'],

    max: [65, 'Age must be less than 65']

  },

  role: {

    type: String,

    enum: ['user', 'admin'], **// Value must be either 'user' or 'admin'**

    default: 'user'

  }

});

const User = **mg.model**('User', userSchema);

const **createDoc**=async()=>

    {

        try{

            const **newUser** = new User({

                username: '  sfga34 ',

                email: ' a1e@xample.com',

                age: 25,

                role: 'user'

              });

            const result= await **newUser**.save(); **//for single data record**

            console.log(result);

        }

        catch(err)

        {

            console.log("Error Occured" + err);

        }

    }

**createDoc();**

**Example:**

**You are developing a MongoDB application using Mongoose and need to enforce specific validation rules on the email and product fields in your userSchema.**

1. **Define a Mongoose schema userSchema with the following fields:**
   * **email:**
     + **The field is required and must be unique in the database.**
     + **It should be validated to ensure it contains a valid email address format.**
     + **If the provided email is invalid, the error message should indicate that the email address is not valid.**
   * **product:**
     + **The field is required.**
     + **It should only allow alphanumeric characters (i.e., letters and numbers).**
     + **If the product field contains invalid characters, the error message should indicate that it is not a valid alphanumeric code.**
2. **Write an asynchronous function createDoc that creates and saves a document with random data.**

**const mg = require('mongoose');**

**const validator = require('validator');**

**mg.connect**("mongodb://127.0.0.1:27017/validations").then(()=>{console.log("success")}).catch((err)=>{console.error(err)});

const userSchema = new **mg.Schema**({

  email: {

    type: String,

    required: true,

    unique: true,

**validate**: { **validator**: (v) => {return **validator**.**isEmail**(v);},

**message**: `This is not a valid email address!`

    }

  },

  product: {

    type: String,

    required: true,

**validate**: {

**validator**: (v) => {return **validator**.**isAlphanumeric**(v);},

**message**: `This is not a valid alphanumeric code!`

    }

  }

});

const **User** = **mg.model**('User', userSchema);

const createDoc=async()=>

    {

        try{

            const **newUser** = new **User**({

                email: 'age@gmail.com',

                product:"fhxvf111"

              });

            const result= await **newUser**.**save()**; //for single data record

            console.log(result);

        }

        catch(err)

        {

            console.log("Error Occured" + err);

        }

    }

    createDoc();

**Example**

Write a node.js script to define a schema having fields like name, age, gender, email.   
**Apply following validations:**  
**(1)** Name field must remove leading/trailing spaces, minimum and maximum length should be 3 & 10 respectively, and name should be stored in lowercase  
**(2)** Age must accept value greater than 0.  
**(3)** Perform Email ID validation on Email field.  
**(4)** Gender must accept values in uppercase only and allowed values are “MALE” & “FEMALE” only.

**const mongoose = require('mongoose');**

**const validator = require('validator');**

**// Connect to MongoDB**

**mongoose.connect**("mongodb://127.0.0.1:27017/validations")

  .then(() => console.log("Connected to MongoDB"))

  .catch((err) => console.error("Connection error:", err));

**// Define the schema**

const **userSchema** = new **mongoose.Schema**({

  name: {

    type: String,

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

    minlength: [3, 'Name must be at least 3 characters long'],

    maxlength: [10, 'Name cannot be more than 10 characters long'],

    trim: true,

lowercase:true

  },

  age: {

    type: Number,

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

    min: [1, 'Age must be greater than 0']

  },

  email: {

    type: String,

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

    unique: true,

    validate: {

      validator: function(v) { return **validator**.isEmail(v); },

      message: props => `${props.value} is not a valid email address!`

    }

  },

  gender: {

    type: String,

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

    uppercase: true, **// Converts the gender to uppercase before saving**

    enum: {

      values: ['MALE', 'FEMALE'],

      message: '{VALUE} is not a valid gender! Allowed values are MALE or FEMALE'

    }

  }

});

**// Create the model**

const User = **mongoose.model**('User', **userSchema**);

**// Function to create a document**

const createDoc = async () => {

  try {

    const **newUser** = new User({

      name: '  LjUuser  ', **// Leading and trailing spaces will be trimmed, and name will be stored in lowercase**

      age: 25,

      email: 'lju@example.com',

      gender: 'male' **// This will be converted to 'MALE'**

    });

    const result = await **newUser**.**save();** **// Save the document**

    console.log(result);

  } catch (err) {

    console.error("Error occurred:", err.message);

  }

};

// Call the function to create a document

createDoc();