Mongoose is object data modeling ( ODM ) for nodejs. In mongodb data or object is stored in Bison ( binary json ).

We can connect mongodb by mongodb package.

We will connect mongoose with nodejs. In the backend mongoose will connect by driver with the mongodb database.

Driver: when we want to install a hardware into our computer. Then we need an interface which we connect our hardware to the computer system.

Nodejs => mongoose => mongo driver => mongodb. After this connect. Now node js is connect to the mongodb.

**Benefits of using mongoose**

* Abstraction from raw low level MongoDB
* Relationship between NoSQL data. Although mongo is non relational database but we can make relationship between NoSQL.
* Provides Schema Validation. We can validate schema field. We can define constraints in schema
* Object – Data mapping – translation of data into object that our code understands and vice versa
* 40 – 60% less code compared to raw mongodb package

Main things is that, schema constraints, validation, object mapping, less code write, relationship between NoSQL data.

**Mongoose schema, model:** first we need to design a schema. After designing schema we need to make a model using this schema. By using this model we can use mongoose method, property and insert data to the database. **Note:** when we will create a model. It mainly a class.

We can make model after creating a schema or we can make a model in different file. First import schema then make model using this schema.

**In same file schema and model and export:**

const mongoose = require("mongoose");

const todoSchema = mongoose.Schema({

  title: {

    type: String,

    required: true,

  },

  description: String,

  status: {

    type: String,

    enum: ["active", "inactive"],

  },

  date: {

    type: Date,

    default: Date.now(),

  },

});

// modeol return a class

const Todo = new mongoose.model("Todo", todoSchema);

module.exports = Todo;

**Post a single data using mongoose:**

// post single todo

const createSingleTodo = async (req, res, next) => {

  const newTodo = new Todo(req.body);

  try {

    await newTodo.save();

    res.status(200).send({ msg: "successfully created todo!" });

  } catch (error) {

    res.status(500).send({ msg: " failed to create todo!" });

  }

};

**Post multiple data:**

// post multiple todo

const createMultipleTodo = async (req, res, next) => {

  await Todo.insertMany(req.body)

    .then(() => res.status(200).send({ msg: "successfully inserted todo" }))

    .catch((error) => console.log(error));

};

**Update single data:**

// update single todo

const updateSingleTodo = async (req, res) => {

  await Todo.updateOne(

    { \_id: req.params.id },

    {

      $set: {

        status: "inactive",

      },

    }

  )

    .then(() => res.status(200).send({ msg: "successfully updated todo" }))

    .catch(() => console.log("Faild to updat"));

};

**Get all data:**

// get all todo

const getAllTodo = async (req, res) => {

  const todoList = await Todo.find({});

  res.status(200).json(todoList);

};

**Get single data and skip property:** we can use changing like limit(), select(), skip()

// get single todo

const getSingleTodo = async (req, res) => {

  const todo = await Todo.findOne({ \_id: req.params.id }).select({ \_id: 0 });

  res.json(todo);

};

**Limit the search result data:** and limit the property

// get all todo

const getAllTodo = async (req, res) => {

  const todoList = await Todo.find({ status: "active" })

    .limit(8)

    .select({ status: 0 });

  res.status(200).json(todoList);

};

**Delete single data:**

// delete single todo

const deleteSingleTodo = async (req, res) => {

  try {

    await Todo.deleteOne({ \_id: req.params.id });

    res.status(200).send("Deleted successfully");

  } catch (error) {

    res.status(404).send({ msg: " failed to delete", err: error });

  }

};

Note: if we use async/await. So no need to use callback pattern. Callback means .then().catch().

If we do not use await. So in this case we can use callback. Like Todo.find().then().catch();

**With async await:**

// get single todo

const getSingleTodo = async (req, res) => {

  try {

    const todo = await Todo.findOne({ \_id: req.params.id }).select({ \_id: 0 });

    res.json(todo);

  } catch (error) {

    console.error(error);

  }

};