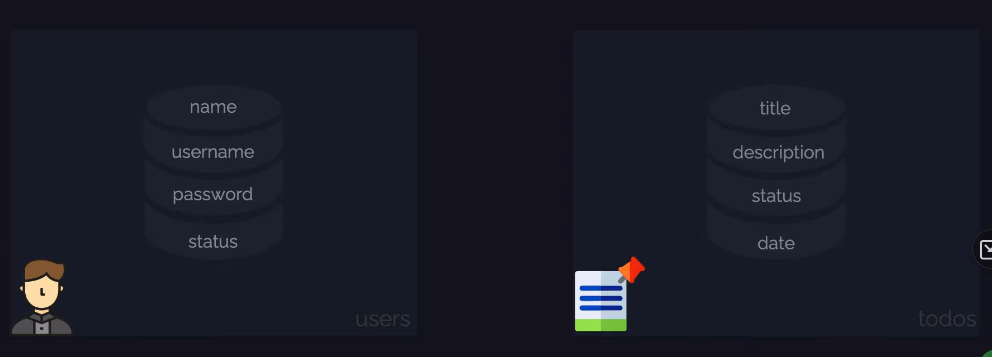
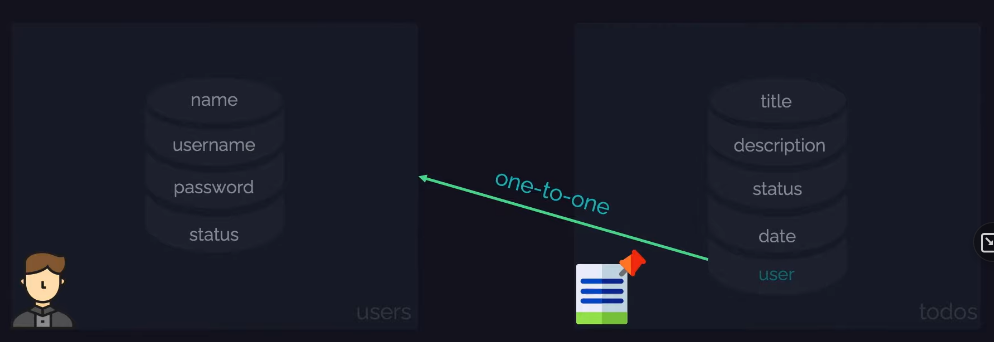
Mongodb ekta nosql ba non relational database holeo onek shomoy project er jonno amader ke database er bivinno collection er moddhe relation maintain korar proyojon pore. Ei tutorial e kivabe nosql mongodb ke kivabe relation maintain kore kaj korte hoy sheta niye discuss kora hobe.

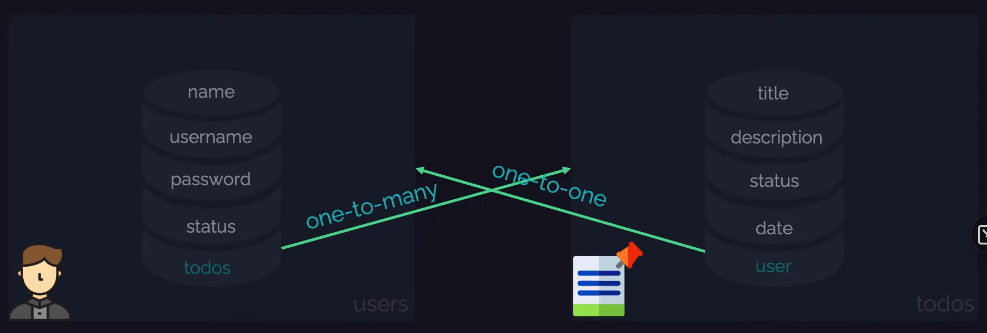
Ekhon porjonto amader example e amra duita collection dekhsi ----- 1. todos 2. users



Users collection e fieldgula chilo name, username, password, status. Todos collection e fieldgula chilo title, description, status and date. Ei duita collection er moddhe kono realtion chilo na. duita collection er moddhe realtion amra kivabe korte pari. Amra biyer analogy ta imagine korte pari. Biyer por meye cheler barite giye uthe. Evabei duita family er moddhe relation hoy. Ekhon todos collection er shathe users collection er relation korar jonno amra todos collection er document e ‘users’ name ekta field ante pari. Shekhane amra users collection theke kono ekta id diye rakhte pari. Jeno ekta todo document jokhon create hobe tokhon ami jodi bole dite pari ei todo ta kon user create koreche, shei user er id ta ami jodi shekhane rekhe dite pari taholei kintu ekta relation toiri hoye gelo. Ekhane ekta todo document ekjon useri create korbe. so dui joner moddhe one-to-one relation.



Ei relation ta one way hocche. Ek side theke. Ekhon amra jodi users theke todos er shatheo ekta relation create korte chai tahole amra user document e ‘todos’ name ekta field rakhte pari jekhane ekadhik todo document er info thakte parbe. mane ekjon user multiple todo create korte pare. Eta one-to-many realtion.



So user er shathe todo er relation ta hocche one-to-many. Onnodike todo er shathe user er relation hocche one-to-one. Ekhon jinishta amra example er maddhome dekhbo.

Vid\_28 tutorial sheshe code je obosthay chilo shekhan theke start kortesi. TododSchema.js e gelam. Todos collection e je document gula hocche shegula ei todoSchema ke follow korei hocche. Amra first e todo er shathe user er relation korte chacchi. One-to-one relation. Er jonno todoSchema te ‘user’ name ekta field create korchi. Ei ‘user’ field er moddhe amra rakhbo user er id ta. User shob info amar rakhar dorkar nai. Just user er id ta rakhte parley kaj hoye jacche. Id er type ta ki hobe? Mongodb te prottekta document er jonno je unique id autogenerate sheta objectId name ekta separate type. eta normal kono string na. amake type ta shetai bolte hobe. type : mongoose.Types.ObjectId diye dilam. ‘ref’ name arekta property dite hobe. Mane ami kar reference ei ‘user’ field ta nicchi? Kon model er reference e nicchi? Amra ‘User’ model er reference e nicchi. Amra UserHandler.js e const User = new mongoose.model(‘User’, userSchema) korechilam.

Ei ‘User’ tai ref : ‘User’ hishebe diye dilam.

Ekhon TodoHandler.js e gelam. Ekhane notun ekta todo document create korar jonno router.post(‘/createone’,….) route e hit kori req.body er data shoho. Todo document er title, description, status, date ---- ei fieldgula charao amake bole dite hobe kon user todo ta create korche. Amader ei route ta protected chilo na. keu authenticate kore todo post korto na. je keu publicly todo post korto. Ei route checkLogin middleware ta boshaya dicchi route ta protect korar jonno. checkLogin function ta user er deya token ta verify kore req object er moddhe userId (auto generated by mongodb) vore disilo. Ekhon TodoHandler.js e nicher kaj ta korlam.

*// post a todo*

router.*post*("/createone", checkLogin, async (*req*, *res*) *=>* {

*//res.send("create one todo");*

  try {

*const* todo = new *Todo*({

      ...req.body,

      user: req.userId,

    });

*const* result = *await* todo.*save*();

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

Ekhon shobar first e mongocompass e giye shob todo document delete kore dilam. Then postman e post method e ‘localhost:3000/user/login’ url e likhlam. Request er body te raw json select kore {‘username’ : ‘dw31’, ‘password’ : ‘samyfoot’} likhlam. Karon User collection e amar already ei user create kora chilo. Ekhon send chaplei response e ekta token pelam.

Ekhon ekta todo create korbo. Postmane e post method e ‘localhost:3000/todo/createone’ url e likhlam. Jehetu ei route ta protected tai request er header e ‘Authorization’ er value te ‘Bearer’ likhe space diye token ta paste kore dilam. Request er body te raw json select kore ei json ta likhlam.

{

   "title": "HTML",

   "description" : "Learn HTML from LWS",

   "status" : "active"

}

Ekhon send chaplei nicher response pelam.

{

    "title": "HTML",

    "description": "Learn HTML from LWS",

    "status": "active",

    "date": "2022-11-08T11:37:37.192Z",

    "user": "63691c200bf43edbfbd5c2d1",

    "\_id": "636a40217834b247f98c80b8",

    "\_\_v": 0

}

So amader ekta todo create hoye gelo.

Check kore dekhlam ei id ebong users collection ‘dw31’ username ala document er id same.

Ekhon postman e get method e url e ‘localhost:3000/todo/getmultiple’ likhlam. Request er header e token dilam. Ekhon send chaplei todos collection er array of todos ta response e pacchi.

*// get multiple todos*

router.*get*("/getmultiple", checkLogin, async (*req*, *res*) *=>* {

*//res.send("Get multiple todos");*

  try {

*const* result1 = *await* Todo.*find*({});

*//const result2 = await Todo.find({ status: "inactive" });*

    res.*status*(200).*json*(result1);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

[

    {

        "\_id": "636a40217834b247f98c80b8",

        "title": "HTML",

        "description": "Learn HTML from LWS",

        "status": "active",

        "date": "2022-11-08T11:37:37.192Z",

        "user": "63691c200bf43edbfbd5c2d1",

        "\_\_v": 0

    }

]

Jemon response pawar kotha temoni pacchi. But ekta jinish ullekkho todo document e amra just creator user er id ta niye ashchi. Ekhon todoSchema te je ‘ref’ set korsilam shetar khelata bujhte parbo. Amra ei id diyei ekhon creator user er info niye ashte parbo populate() method ta use kore. Amra .populate(‘user’) use korbo. Argument er ei ‘user’ ta holo ei user. Orthat document er property te je name dibo otai use korte hobe.

*// get multiple todos*

router.*get*("/getmultiple", checkLogin, async (*req*, *res*) *=>* {

*//res.send("Get multiple todos");*

  try {

*const* result1 = *await* Todo.*find*({}).*populate*("user");

*//const result2 = await Todo.find({ status: "inactive" });*

    res.*status*(200).*json*(result1);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

Ekhon request header e token diye get method e ‘localhost:3000/todo/getmultiple’ route hit korle nicher reponse pacchi.

[

    {

        "\_id": "636a40217834b247f98c80b8",

        "title": "HTML",

        "description": "Learn HTML from LWS",

        "status": "active",

        "date": "2022-11-08T11:37:37.192Z",

        "user": {

            "\_id": "63691c200bf43edbfbd5c2d1",

            "name": "Warner",

            "username": "dw31",

            "password": "$2b$08$L9CCHYKwZwFIjyF7h8goouRcGPNKZy8Cqk2LIY9uEhsRfv4TYtJCe",

            "\_\_v": 0

        },

        "\_\_v": 0

    }

]

So ekhon ‘user’ property er value hishebe creator user jabotiyo info chole ashche. Ekhon ami chacchi user just name ar username ashbe. Ar kichu na. er jonno populate method e 2nd parameter dite hobe.

*// get multiple todos*

router.*get*("/getmultiple", checkLogin, async (*req*, *res*) *=>* {

*//res.send("Get multiple todos");*

  try {

*const* result1 = *await* Todo.*find*({}).*populate*("user", "name username -\_id");

*//const result2 = await Todo.find({ status: "inactive" });*

    res.*status*(200).*json*(result1);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

Ekhon response ashlo …

[

    {

        "\_id": "636a40217834b247f98c80b8",

        "title": "HTML",

        "description": "Learn HTML from LWS",

        "status": "active",

        "date": "2022-11-08T11:37:37.192Z",

        "user": {

            "name": "Warner",

            "username": "dw31"

        },

        "\_\_v": 0

    }

]

-\_id na dile id chole ashto.

So todo theke user er dike (one to one) relation ta bujhe gelam.

Ekhon amra ulta ta dekhbo. Orthat UserHandler.js e kono ekta route er moddhe amra user jokhon niye ashbo tokhon oi user er corresponding shob todo niye ashbo. Amra user document ‘todos’ name ekta property add korbo. Shetar value hobe ekta array of objectids. Shei array theke amra further info niye ashbo.

UserSchema.js e gelam. userSchema te ‘todos’ name ekta field nilam.

UserSchema.js …

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

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

  name: {

    type: String,

    required: true,

  },

  username: {

    type: String,

    required: true,

  },

  password: {

    type: String,

    required: true,

  },

  status: {

    type: String,

    enum: ["active", "inactive"],

  },

  todos: [

    {

      type: mongoose.Types.ObjectId,

      ref: "Todo"

    },

  ],

});

module.exports = userSchema;

ekhon TodoHandler.js e jabo. Ekhon amar kaj holo jokhon kono user kono todo create korbe tokhon shathe shathe oi user document er todos field ta update kora. Orthat notun todo er id ta ei array te push korte hobe.

TodoHandler.js e userSchema require kore anlam. Const User = new mongoose.model(‘User’, userSchema) kore model ta niya ashlam. Router.post(‘/createone’,….) e chole gelam.

*// post a todo*

router.*post*("/createone", checkLogin, async (*req*, *res*) *=>* {

*//res.send("create one todo");*

  try {

*const* todo = new *Todo*({

      ...req.body,

      user: req.userId,

    });

*const* result = *await* todo.*save*();

*await* User.*updateOne*(

      {

        \_id: req.userId,

      },

      {

        $push: {

          todos: result.\_id,

        },

      }

    );

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }});

Shobar first e dw31 ebong samyfoot diye login korlam. Token copy korlam.

Ekhon post method e url e “localhost:3000/todo/createone” likhlam. Request er header e token dilam. Request er body te raw json select kore eta dilam.

{

    "title" : "CSS",

    "description" : "Learn CSS from LWS",

    "status" : "inactive"

}

Ekhon send chaplei ekta notun todo create holo. Ja response e pelam.

{

    "title": "CSS",

    "description": "Learn CSS from LWS",

    "status": "inactive",

    "date": "2022-11-08T15:57:31.541Z",

    "user": "63691c200bf43edbfbd5c2d1",

    "\_id": "636a7cf865d9e4cf0bed7c40",

    "\_\_v": 0

}

Eki sthate amar creator user i.e. dw31 document ta o update hoye gese. Mongocompass e giye check kore dekhlam dw31 document ta te ‘todos’ name ekta property chole ashche. Jetar value ekta array. Ei array 1st element e ekta objectId ache. Ei objectId ar ei id same.

Ekhon ami arekta todo create korbo dw31 user diye. Erpor dw31 user check korle dekha jabe ‘todos’ array te notun create howa todo er id add hoise.

Ekhon UserHandler.js e jabo ebong shob user document dekhar system korbo.

*// Get all users*

router.*get*("/getmultipleusers", async (*req*, *res*) *=>* {

  try {

*const* users = *await* User.*find*({}).*populate*(

      "todos",

      "title description -\_id"

    );

    res.*status*(200).*json*(users);

  } catch (error) {

    console.*log*(error);

    res.*status*(500).*json*({

      error: "There was an error",

    });

  }

});

Postman e get method e ‘localhost:3000/user/getmultipleusers’ likhe hit korlam. Response pailam eta.

[

    {

        "\_id": "63691c200bf43edbfbd5c2d1",

        "name": "Warner",

        "username": "dw31",

        "password": "$2b$08$L9CCHYKwZwFIjyF7h8goouRcGPNKZy8Cqk2LIY9uEhsRfv4TYtJCe",

        "\_\_v": 0,

        "todos": [

            {

                "title": "CSS",

                "description": "Learn CSS from LWS"

            },

            {

                "title": "Bootstrap",

                "description": "Learn Bootstrap from LWS"

            }

        ]

    }

]

Amar jehetu ekta useri ache tai array te oi ekta user dw31 pailam.

Important note : amra todoSchema kingba userSchema te je user ebong todos name extra field add korsi shekhane amra just type : objectId ebong ref disi. Then pore oi objectId ebong ref diye populate kore corresponding detail info niye ashchi. But amar user kingba todo olpo hole recommended way hocche populate avoid kora. Ekhetre todoSchema te user field e ja ja (name, username, status) proyojon shob object akare rekhe dibo. userSchema teo todos field er jonno same kaj ta korbo.

Final codes…

.env

DB\_URL = "mongodb://localhost:27017/YTStudentRestDb"

TODO\_URL = "mongodb://localhost:27017/todos"

SecretKey = "kujhikkujhik"

Connection.js …

*require*("dotenv").*config*();

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

*const* db\_url = process.env.TODO\_URL;

*const* *connector* = () *=>* {

  mongoose

    .*connect*(db\_url)

    .*then*(() *=>* console.*log*("Connection with databse established"))

    .*catch*((*err*) *=>* {

      console.*log*(err);

    });

};

module.exports = connector;

index.js …

*const* express = *require*("express");

*const* connector = *require*("./data/connection");

*const* todoHandler = *require*("./data/TodoHandler");

*const* userHandler = *require*("./data/UserHandler");

*const* app = *express*();

*require*("dotenv").*config*();

app.*use*(express.*json*());

*connector*();

app.*use*("/todo", todoHandler);

app.*use*("/user", userHandler);

*const* *errorHandler* = (*err*, *req*, *res*, *next*) *=>* {

  if (res.headersSent) {

*return* *next*(err);

  } else {

    res.*status*(500).*json*({ error: err });

  }

};

app.*use*(errorHandler);

app.*listen*(3000, () *=>* {

  console.*log*("Listening on port 3000");

});

UserSchema.js …

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

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

  name: {

    type: String,

    required: true,

  },

  username: {

    type: String,

    required: true,

  },

  password: {

    type: String,

    required: true,

  },

  status: {

    type: String,

    enum: ["active", "inactive"],

  },

  todos: [

    {

      type: mongoose.Types.ObjectId,

      ref: "Todo",

    },

  ],

});

module.exports = userSchema;

UserHandler.js …

*const* express = *require*("express");

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

*const* jwt = *require*("jsonwebtoken");

*const* userSchema = *require*("./UserSchema");

*const* bcrypt = *require*("bcrypt");

*const* router = express.*Router*();

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

*// Signup*

router.*post*("/signup", async (*req*, *res*) *=>* {

  try {

*const* hashedPassword = *await* bcrypt.*hash*(req.body.password, 8);

*const* newUser = new *User*({

      name: req.body.name,

      username: req.body.username,

      password: hashedPassword,

    });

*const* result = *await* newUser.*save*();

    res.*status*(200).*json*(result);

  } catch {

    res.*status*(500).*json*({

      error: "Signup Failed!",

    });

  }

});

*// Login*

router.*post*("/login", async (*req*, *res*) *=>* {

  try {

*const* user = *await* User.*find*({ username: req.body.username });

    if (user && user.length > 0) {

*const* isValidPassword = *await* bcrypt.*compare*(

        req.body.password,

        user[0].password

      );

      if (isValidPassword) {

*const* token = jwt.*sign*(

          {

            username: user[0].username,

            userId: user[0].\_id,

          },

          process.env.SecretKey,

          { expiresIn: "1h" }

        );

        res.*status*(200).*json*({

          token: token,

          message: "Login Successfull!",

        });

      } else {

        res.*status*(401).*json*({ error: "Authentication Failed!" });

      }

    } else {

      res.*status*(401).*json*({ error: "Authentication Failed!" });

    }

  } catch {

    res.*status*(401).*json*({ error: "Authentication Failed!" });

  }

});

*// Get all users*

router.*get*("/getmultipleusers", async (*req*, *res*) *=>* {

  try {

*const* users = *await* User.*find*({}).*populate*(

      "todos",

      "title description -\_id"

    );

    res.*status*(200).*json*(users);

  } catch (error) {

    console.*log*(error);

    res.*status*(500).*json*({

      error: "There was an error",

    });

  }

});

module.exports = router;

checkLogin.js …

*const* jwt = *require*("jsonwebtoken");

*const* *checkLogin* = (*req*, *res*, *next*) *=>* {

*const* { authorization } = req.headers;

  try {

*const* token = authorization.*split*(" ")[1];

*const* decoded = jwt.*verify*(token, process.env.SecretKey);

*const* { username, userId } = decoded;

    req.username = username;

    req.userId = userId;

*next*();

  } catch {

*next*("Authentication Failed!");

  }

};

module.exports = checkLogin;

TodoSchema.js …

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

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

  title: {

    type: String,

    required: true,

  },

  description: String,

  status: {

    type: String,

    enum: ["active", "inactive"],

  },

  date: {

    type: Date,

    default: Date.*now*(),

  },

  user: {

    type: mongoose.Types.ObjectId,

    ref: "User",

  },

});

*// instance methods*

todoSchema.methods = {

*findActive*: *function* () {

*return* mongoose.*model*("Todo").*find*({ status: "active" });

  },

*findActiveCB*: *function* (*cb*) {

*return* mongoose.*model*("Todo").*find*({ status: "inactive" }, cb);

  },

};

*// static methods*

todoSchema.statics = {

*findByJs*: *function* () {

*return* this.*find*({ title: /js/i });

  },

};

*// query helpers*

todoSchema.query = {

*byLanguage*: *function* (*language*) {

*return* this.*find*({ title: new *RegExp*(language, "i") });

  },

};

module.exports = todoSchema;

TodoHandler.js …

*const* express = *require*("express");

*const* router = express.*Router*();

*const* todoSchema = *require*("./TodoSchema");

*const* userSchema = *require*("./UserSchema");

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

*const* checkLogin = *require*("./checkLogin");

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

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

*// get multiple todos*

router.*get*("/getmultiple", checkLogin, async (*req*, *res*) *=>* {

*//res.send("Get multiple todos");*

  try {

*const* result1 = *await* Todo.*find*({}).*populate*("user", "name username -\_id");

*//const result2 = await Todo.find({ status: "inactive" });*

    res.*status*(200).*json*(result1);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*//get a todo by id*

router.*get*("/get/:id", async (*req*, *res*) *=>* {

*//res.send("Get a todo");*

  try {

*const* id = req.params.id;

*const* result = *await* Todo.*findById*(id);

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// get only active todos*

router.*get*("/getactivetodos", async (*req*, *res*) *=>* {

  try {

*const* todo = new *Todo*();

*const* result = *await* todo.*findActive*();

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// get only active todos with cb*

router.*get*("/getactivetodoscb", (*req*, *res*) *=>* {

*const* todo = new *Todo*();

  todo.*findActiveCB*((*err*, *data*) *=>* {

    if (err) {

      res.*status*(500).*json*({ error: err });

    } else {

      res.*status*(200).*json*(data);

    }

  });

});

*// get todos with only js*

router.*get*("/js", async (*req*, *res*) *=>* {

  try {

*const* result = *await* Todo.*findByJs*();

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// get todos with by query helper*

router.*get*("/language", async (*req*, *res*) *=>* {

  try {

*const* result = *await* Todo.*find*().*byLanguage*("css");

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// post a todo*

router.*post*("/createone", checkLogin, async (*req*, *res*) *=>* {

*//res.send("create one todo");*

  try {

*const* todo = new *Todo*({

      ...req.body,

      user: req.userId,

    });

*const* result = *await* todo.*save*();

*await* User.*updateOne*(

      {

        \_id: req.userId,

      },

      {

        $push: {

          todos: result.\_id,

        },

      }

    );

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// post multiple todos*

router.*post*("/createmultiple", async (*req*, *res*) *=>* {

*//res.send("crete multiple todos");*

  try {

*const* result = *await* Todo.*insertMany*(req.body);

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// update a todo by id*

router.*put*("/updatebyid/:id", async (*req*, *res*) *=>* {

  try {

*const* id = req.params.id;

*const* updateData = { $set: { status: "active" } };

*const* options = { new: true };

*const* result = *await* Todo.*findByIdAndUpdate*(id, updateData, options);

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// update multiple todos*

router.*put*("/updatemultiple", async (*req*, *res*) *=>* {

*//res.send("update multiple todos");*

  try {

*const* filterData = { title: "JAVASCRIPT" };

*const* updateData = { status: "active" };

*const* result = *await* Todo.*updateMany*(filterData, updateData);

    res.*status*(200).*json*(result);

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// delete a todo by id*

router.*delete*("/delete/:id", async (*req*, *res*) *=>* {

  try {

*const* id = req.params.id;

*const* result = *await* Todo.*findByIdAndDelete*(id);

    res.*status*(200).*json*({ message: "deleted successfully" });

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

*// delete multiple todos*

router.*delete*("/deletemultiple", async (*req*, *res*) *=>* {

*//res.send("delete multiple todos");*

  try {

*const* filterdata = { status: "active" };

*const* result = *await* Todo.*deleteMany*(filterdata);

    res.*status*(200).*json*({ message: "multiple todos deleted" });

  } catch (error) {

    res.*status*(500).*json*({ error: error });

  }

});

module.exports = router;