Ajke amra dekhbo kivabe json web token (jwt) er maddhome express js application e amra authentication korte pari.

Authenticate korar jonno first e amader application e kichu user proyojon. User ashbe kivabe? User ke definitely amader application e sign up korte hobe. Tarpor shei user ke amra database e add korte parbo. Erpore user je credentials diye signup korlo shei credentials diye she login korbe. login korar por amra orthat server ekta json web token dibo. Shei token ta diye user amader application er jekono protected route e jokhon hit korbe tokhon amra check korbo user shothik token diye request koreche kina. Shothik hole requested datar access take dibo. Otherwise dibo na. etai holo authentication pattern er ekta brief description. Ekhon coding er maddhome bistarito dekhbo.

Vid\_27 tutorial sheshe amar code je obosthay chilo shekhan thekei ajker kaj shuru korbo. At first, amader user collection create korar jonno ekta user schema create korbo. Amader ‘data’ folder e UserSchema.js name ekta file nilam. Sheknane userSchema create korlam. Dhore nicchi, user collection e ekta ‘name’ thakbe user er. Shetar type dilam string, ebong eta ‘required’ hobe. Arekta property thakbe ‘username’. Etaro type hobe string ebong etao required hobe. Arekta jinish dorkar hob. Sheta holo ‘password’. Etao diye dilam. ‘status’ field tao todoSchema er moto rekhe dilam.

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"],

  },

});

module.exports = userSchema;

ekhon ei userSchema er upor base kore user signup er bebostha korbo. Amader ‘data’ folder e TodoHandler.js ache. Amader ekta UserHandler.js file o create korte hobe ekhon.

.env

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

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

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*();

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*("Listenin 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"],

  },

});

module.exports = userSchema;

UserHandler.js …

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

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

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

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

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

*// signup*

router.*get*("/signup", (*req*, *res*) *=>* {});

module.exports = router;

inside ‘data’ folder => connection.js, TodoHandler.js, TodoSchema.js, UserHandler.js, UserSchema.js

so amar boilerplate code hoye gelo. Signup e jehetu user amake kichu info dibe shehetu amra eta ke post method e receive korbo. Ekhane ekta trycatch block niye ashbo. newUser = new User(…) kore database er ‘users’ collection e ekta document anar bebostha shuru korbo. Ekhane user er infogula (name. username, password) dite hobe. Password deyar shomoy protected way te hash kore nibo jate programmer o jate na jane user er password ki. Otherwise hacker ra kam shaira felbe. So password ta ke encrypt korte hobe. Etar jonno bcrypt name ekta library use korbo. Npm install bcrypt kore sheta niye ashlam. bcrypt.hash(…) name ekta method ache. Bcrypt.hash() amader plain password ke shudhu encrypt kore na, password er prottekta text age pore kichu extra text boshaya ney. Kotogula text boshabe sheta amra saltRounds name ekta parameter er maddhome likhe dite pari.

UserHandler.js …

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

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

*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, 10);

*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 (error) {

    res.*status*(500).*json*({ errorMessage: "Signup failed!" });

  }

});

module.exports = router;

ekhon postman e giye post method e localhost/user/signup --- ei url likhlam. Request er body section e raw json select kore

{

"name": "smith",

"username": "smith49",

"password": "indiamybunny"

}

likhe send kore dilam. Response e pelam nicher jinishta…

{

"name": "smith",

"username": "smith49",

"password": "$2b$10$1wC/vEnyNLp3UAbzMopIs.7nwreqs578X/60bcIqA0LM1NlCEBsfq",

"\_id": "6367bcf7e98b5f52cfc4bf37",

"\_\_v": 0

}

Ekhon mongoCompass e ‘todos’ database e giye ‘users’ collection check kore dekhlam amar document create hoye ache.

Ekhon amra dekhbo login process. Login er jonno amader alada route banate hobe. Etao post method e nite hobe. Karon shekaneo amake username, password input hishebe nite hobe.

.env…

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

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

SecretKey = "kujhikkujhik"

UserHandler.js ….

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

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

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

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

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

*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, 10);

*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 (error) {

    res.*status*(500).*json*({ errorMessage: "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*({

          access\_token: token,

          message: "Login Successful!",

        });

      } else {

        res.*status*(500).*json*({

          error: "Authentication Failed!",

        });

      }

    } else {

      res.*status*(500).*json*({ error: "Authentication Failed!" });

    }

  } catch {

    res.*status*(500).*json*({

      error: "Authentication Failed!",

    });

  }

});

module.exports = router;

ei example e login er shomoy user amake username ar password request er body te json akare pathabe. Ami ekhane validation korbo na. amake at least dekhte hobe je username ta she dicche shei username ta amar database e ache kina. So user er pathano username diye amake find kore dekhte hobe database e ache kina. Const user = await User.find(…) kore shetai korlam. Ekane user hishebe ekta array pabo. Erpor if-else diye check korlam array element at least 1 ta kina. Jodi hoy tahole client er pathano password (normal plain text password) er shathe array er first element er password (which is hashed) compare korlam bcrypt.compare() diye. Jodi eta true return kore orthat jodi mile tahole json web token diye token create korbo. Er jonno npm install jsonwebtoken kore package ta niye ashlam. Token create korar jonno jwt.sign() use korlam. Etate first parameter ney user er kichu public info. Then programmer er deya ekta secret key ar last e kichu options.

Now postman e giye post method e localhost:3000/user/login url likhlam. Request er bodyte raw json select kore nicher json likhlam.

{“username”: “smith49”, “password”: “indiamybunny”}

Then send korle response e token ar ‘login successful!’ lekha pacchi. json object tar kono kichu vul pathale "Authentication Failed!" response ashe.

Ekhon token ta ke verify korar jonno jwt.io website e giye encoded part e token ta ke paste kore dilam. Decoded part e header ebong payload section e duita json object dekhte pacchi. Header part e amar token er algorithm ebong type ta mention kora ache. Ar payload part e token e deya public info ta dekhacche. Ekdom laste verify signature part e amader signature er verification er jonno secret key ta chacche. Ota diye dile signature verified dekhacche.

Authenticated user ba client ke amra token diye disi. Ekhon oi user ke ami bolbo jokhoni tumi amar kono protected route e hit korba tokhoni tumi token ta diba. otherwise ami tomake data dibo na. for e.g. ami ‘localhost:3000/todo/getmultiple’ --- ami ei route ta protect korte chai. Je je route ami protect korte chai shei shei route e user ke hit korte hole obosshoi access token dite hobe. Sheta kivabe ensure korbo shetai ekhon dekhchi. Route gulake protect korar jonno amader middleware use korte hobe. Ami je route ke protect korte chai shei route er callback function er age middleware ta boshaya dibo. ‘data’ folder e checkLogin.js name ekta file nilam. Shekhane checkLogin name ekta middleware banalam. checkLogin e req, res, next er access amar ache. Amake age decide korte hobe user ba client amake kivabe token ta pathabe. She req.body er moddhe kingba request er header e pathate pare. Best practice hocche request er header e neya. Amar ei application ta ekta backend api. Frontend jekono kichu diye banano hote pare jemon react, vue or angular. Ami ekhane postman theke client er request nicchi. But request ta jekono dhoroner web application ba web frontend theke ashte pare. Sheta kono ekta framework o hote pare. Maximum khetre oi framework gula header e authentication token pathate prefer kore. Ba built in vabe shevabei thake. So ekta pattern hocce header e pathano. ‘Authorization’ namok ekta header e pathano. ‘Authorization’ er value te token lekhar age ‘Bearer’ likhe dite hobe.

Ekhon checkLogin middleware e first e const {authorization} = req.headers kore ‘Authorization’ header er value ta ber kore anlam. Then ekta trycatch block nilam. Try er moddhe first e ekta ‘token’ variable er moddhe only jwt token ta nilam. Ei jwt token er moddhe je data ta ache sheta decode kore ber kore ante hobe. Jwt.verify(token, secretOrPublicKey,[options, callback]) name jsonwebtoken er ekta function ache jeta diye ami token ta verify korte pari. Eta ekta asynchronous process. Jodi token verified hoy tahole ei function ta token er payload ta return kore. Jodi invalid hoy tahole instantly okhan theke error throw kore. Ekhon ‘decoded’ name ekta variable er moddhe ei function ta call kore dilam. ‘decoded’ er moddhe payload er object ta chole ashlo. Then sheta amra destructure kore username, userId ber kore ene req object er moddhe dhukaya dilam jate porer route e dorkar hole bebohar korte pari. Erpor next() call kore dilam. Erpor catch block e next(‘Authentication Failed!’) diye dilam. Ei error ta index.js er error handler er kache chole jabe.

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;

ekhon TodoHandler.js e checkLogin ta require kore anbo. Then nicher position e boshaya dibo eta ke.

TodoHandler.js …

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

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

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

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

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

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

*// 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 });

  }

});

Now at first postman e giye post method e ‘localhost:3000/user/login’ url e likhlam. Request er body te raw json select kore shekhane {‘username’: ‘smith49’, ‘passoword’ : ‘indiamybunny’} likhe send chaplei repsonse token pelam.

Ekhon get method e url e ‘localhost:3000/todo/getmultiple’ likhlam. Request er header ‘Authorization’ name ekta key likhlam. Etar value dilam ‘Bearer hijibiji’. Ekhon request send korle response pacchi ‘Authentication Failed!’. But ‘Authorization’ er value te ‘Bearer’ likhe shothik token diye send korlei amar todos collection er shob document response e chole ashche. Orthat ‘localhost:3000/todo/getmultiple’ route ta checkLogin diye protect kore fellam.

Now 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*("Listenin 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"],

  },

});

module.exports = userSchema;

UserHandler.js …

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

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

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

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

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

*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, 10);

*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 (error) {

    res.*status*(500).*json*({ errorMessage: "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*({

          access\_token: token,

          message: "Login Successful!",

        });

      } else {

        res.*status*(401).*json*({

          error: "Authentication Failed!",

        });

      }

    } else {

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

    }

  } catch {

    res.*status*(401).*json*({

      error: "Authentication Failed!",

    });

  }

});

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*(),

  },

});

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

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

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

*// 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 });

  }

});

*//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", async (*req*, *res*) *=>* {

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

  try {

*const* todo = new *Todo*({

      title: req.body.title,

      description: req.body.description,

      status: req.body.status,

      date: req.body.date,

    });

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

    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;

package.json …

{

  "scripts": {

    "start": "nodemon index.js",

    "production": "NODE\_ENV=production nodemon index.js"

  },

  "dependencies": {

    "bcrypt": "^5.1.0",

    "body-parser": "^1.20.0",

    "cookie-parser": "^1.4.6",

    "cors": "^2.8.5",

    "dotenv": "^16.0.3",

    "ejs": "^3.1.8",

    "express": "^4.18.1",

    "jsonwebtoken": "^8.5.1",

    "mongoose": "^6.7.0",

    "multer": "^1.4.5-lts.1"

  }

}