Node.js Token based auth

```
// 1. load required packages

var express = require("express");

var path = require("path");

var bodyParser = require("body-parser");

var mongoose = require("mongoose");

var jwt = require("jsonwebtoken");

mongoose.Promise = global.Promise;

var cors = require("cors");

var instance = express();

var router = express.Router();

// 2. configure middlewares

instance.use(router);

instance.use(bodyParser.urlencoded({ extended: false }));

instance.use(bodyParser.json());

instance.use(cors());

// 3. connect to mongodb

mongoose.connect(

  "mongodb://localhost/ProductsAppDb",

  { useNewUrlParser: true }
```

```
);

var dbConnect = mongoose.connection;

if (!dbConnect) {

  console.log("Sorry Connection is not established");

  return;

}



/* #region Code for the User Management */

// 4. Users Schema

var userSchema = mongoose.Schema({

  UserName: String,

  Password: String

});

var userModel = mongoose.model("Users", userSchema, "Users");



/* #region Create/Register User */

// 5. create a new user

instance.post("/api/users/create", function(request, response) {

  var user = {

    UserName: request.body.UserName,

    Password: request.body.Password
  };

  userModel.create(user, function(err, res) {

    if (err) {

      response.statusCode = 500;

      response.send({ statusCode: response.statusCode, message: err });

    }

    response.send({ statusCode: 200, message: res });

  });

});

/* #endregion */

/* #endregion */



/* #region Login User and Generate Token */

// 6. the secret for the JWT

var jwtSettings = {

  jwtSecret: "dbcsbiobc0708hdfcyesbombob"

};

// set the secret with express object

instance.set("jwtSecret", jwtSettings.jwtSecret);

var tokenStore = "";

// 7. authenticate user
```

```
instance.post("/api/users/auth", function(request, response) {

  var user = {

    UserName: request.body.UserName,

    Password: request.body.Password

  };



  console.log("In Auth User " + JSON.stringify(user));

  userModel.findOne({ UserName: request.body.UserName }, function(err, usr) {

    if (err) {

      console.log("Some error occured ");

      throw error;

    }

    // 7a. if user not found the respond error

    if (!usr) {

      response.send({

        statusCode: 404,

        message: "Sorry!User is not available"

      });

    } else if (usr) {

      // 7b. user is available but password not match the

      // respond error
}

    console.log("In else if " + JSON.stringify(usr));

    if (usr.Password != user.Password) {

      response.send({

        statusCode: 404,

        message: "Sorry!User Name and Password does not match"

      });

    } else {

      // 7c. sing-In the user and generate token

      var token = jwt.sign({ usr }, instance.get("jwtSecret"), {

        expiresIn: 3600

      });

      // save token globally

      tokenStore = token;

      response.send({

        authenticated: true,

        message: "Login Success",

        token: token

      });

    }

  }

});
```

```
});

/* #endregion */

/* #region The code for Product API */
var productsSchema = mongoose.Schema({
  ProductId: Number,
  ProductName: String,
  CategoryName: String,
  Manufacturer: String,
  Price: Number
});


var productModel = mongoose.model("Products", productsSchema, "Products");
// 8. verify the token and provide access
instance.get("/api/products", function(request, response) {
  // 8a. read request headers header contains bearer<space><token>
  var tokenRecived = request.headers.authorization.split(" ")[1];
  // 8b. verify the token
  jwt.verify(tokenRecived, instance.get("jwtSecret"), function(err, decoded)
{
    console.log("in verify");

    if (err) {

      console.log("in auth error");

      response.send({ success: false, message: "Token verification failed"

});

    } else {

      console.log("in auth success");

      // 8c. decode the request

      request.decoded = decoded;

      productModel.find().exec(function(err, res) {

        if (err) {

          response.statusCode = 500;

          response.send({ status: response.statusCode, error: err });

        }

        response.send({ status: 200, data: res });

      });

    }

  });

});



instance.post("/api/products", function(request, response) {
  // parsing posted data into JSON

  var prd = {

    ProductId: request.body.ProductId,

    ProductName: request.body.ProductName,

    CategoryName: request.body.CategoryName,

    Manufacturer: request.body.Manufacturer,

    Price: request.body.Price

  };

  productModel.create(prd, function(err, res) {

    if (err) {

      response.statusCode = 500;

      response.send(err);

    }

    response.send({ status: 200, data: res });

  });

});



instance.get("/api/products/:id", function(request, response) {});



instance.put("/api/products/:id", function(request, response) {});
instance.delete("/api/products/:id", function(request, response) {});



/* #endregion */

// 6. start listening

instance.listen(4070, function() {

  console.log("started listening on port 4070");

});
```