ACES- Data Analytics SOFTWARE _Node_Server

CONFIG/KEYS

```
module.exports = {
  mongoURI: "mongodb+srv://Sameer:1010@cluster0.kq5wym0.mongodb.net/test",
  secretOrKey: "secret",
};
```

CONFIG/PASSPORT

```
const JwtStrategy = require("passport-jwt").Strategy;
const ExtractJwt = require("passport-jwt").ExtractJwt;
const mongoose = require("mongoose");
const User = mongoose.model("users");
const keys = require("../config/keys");

const opts = {};
opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken();
opts.secretOrKey = keys.secretOrKey;

module.exports = (passport) => {
  passport.use(
    new JwtStrategy(opts, (jwt_payload, done) => {
      User.findById(jwt_paylo.id)
        .then((user) => {
          if (user) {
            return done(null, user);
          }
          return done(null, false);
        })
        .catch((err) => console.log(err));
    })
  );
};
```

models/User

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

const UserSchema = new Schema({
  name: {
    type: String,
    required: true,
```

```
  },
  email: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
  },
  Role: {
    type: String,
    required: true,
  },
  verified: {
    type: Boolean,
    default: false,
  },
  //   date: {
  //     type: Date,
  //     required: Date.now(),
  //   },
});
module.exports = User = mongoose.model("users", UserSchema);
```

routes\api\users.js

```
const express = require("express");
const router = express.Router();
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const keys = require("../../config/keys");
const validateRegisterInput = require("../../validation/register");
const validateLoginInput = require("../../validation/login");
const User = require("../../models/User");
var mysql = require("mysql");

router.post("/schema", function (req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  var connection = mysql.createConnection({
    host: req.body.name,
    port: req.body.port,
    user: req.body.user,
    password: req.body.password,
    database: req.body.database,
  });
  connection.query(
    "SELECT schema_name FROM information_schema.schemata",
    function (err, rows) {
      let result = Object.values(JSON.parse(JSON.stringify(rows)));
      res.status(200).json(rows);
    }
```

```
  );
});

router.post("/database", function (req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  var connection = mysql.createConnection({
    host: req.body.connectivity.name,
    port: req.body.connectivity.port,
    user: req.body.connectivity.user,
    password: req.body.connectivity.password,
    database: req.body.connectivity.database,
  });
  connection.query(
    `SELECT TABLE_NAME FROM information_schema.tables WHERE table_schema
='${req.body.databaseName}' ;`,
    function (err, rows) {
      res.status(200).json(rows);
    }
  );
});

router.post("/databaseTable", function (req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  var connection = mysql.createConnection({
    host: req.body.connectivity.name,
    port: req.body.connectivity.port,
    user: req.body.connectivity.user,
    password: req.body.connectivity.password,
    database: req.body.connectivity.database,
  });
  connection.query(
    `SELECT * FROM ${req.body.databaseName}.${req.body.databaseTableName}`,
    function (err, rows) {
      res.status(200).json(rows);
    }
  );
});
router.post("/queryGeneration", function (req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body);
  var connection = mysql.createConnection({
    host: req.body.connectivity.name,
    port: req.body.connectivity.port,
    user: req.body.connectivity.user,
    password: req.body.connectivity.password,
    database: req.body.connectivity.database,
  });
  connection.query(req.body.requestQuery, function (err, rows) {
    console.log(rows);
    res.status(200).json(rows);
  });
});
router.post("/forgotPassword", (req, res) => {
  // console.log(req.body);
```

```
  const email = req.body.email;
  User.findOne({ email }).then((user) => {
    if (!user) {
      console.log("email not found");
      return res.json({ email: "Email not found", otp: "" });
    } else {
      console.log(user);
      return res.json({
        email: "Otp Send Succesfully",
        otp: req.body.otp,
        user: user,
      });
    }
  });
});
router.post("/updatePassword", (req, res) => {
  const email = req.body.email;
  console.log(req.body);
  User.findOne({ email }).then((user) => {
    if (!user) {
      console.log("email not found");
      return res.json({ email: "Email not found", otp: "" });
    } else {
      console.log(user.password);
      user.password = req.body.password;
      bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(user.password, salt, (err, hash) => {
          user.password = hash;
          user.save().catch((err) => console.log(err));
        });
      });
      return res.json({ email: "user Exists", otp: req.body.otp, user: user });
    }
  });
});
router.post("/register", (req, res) => {
  const { errors, isValid } = validateRegisterInput(req.body);
  if (!isValid) {
    return res.status(400).json(errors);
  }
  User.findOne({ email: req.body.email }).then((user) => {
    console.log(user, req);
    if (user) {
      return res.status(400).json({ email: "Email already exists" });
    } else {
      const newUser = new User({
        name: req.body.name,
        email: req.body.email,
        password: req.body.password,
        Role: req.body.Role,
      });
      console.log(newUser);
      bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(newUser.password, salt, (err, hash) => {
```

```
            newUser.password = hash;
            newUser
              .save()
              .then((user) => res.json(user))
              .catch((err) => console.log(err));
          });
        });
      }
    });
  });

  router.post("/login", (req, res) => {
    // console.log(res, res, "here");
    const { errors, isValid } = validateLoginInput(req.body);
    if (!isValid) {
      return res.status(400).json(errors);
    }

    const email = req.body.email;
    const password = req.body.password;

    User.findOne({ email }).then((user) => {
      if (!user) {
        return res.status(404).json({ emailnotfound: "Email not found" });
      }
      bcrypt.compare(password, user.password).then((isMatch) => {
        console.log(password, user.password);
        if (isMatch) {
          const payload = {
            id: user.id,
            name: user.name,
            Role: user.Role,
          };

          jwt.sign(
            payload,
            keys.secretOrKey,
            {
              expiresIn: 31556929,
            },
            (err, token) => {
              res.json({
                success: true,
                token: "Bearer " + token,
              });
            }
          );
        } else {
          return res
            .status(400)
            .json({ passwordincorrect: "Password incorrect" });
        }
      });
    });
```

```
});
module.exports = router;
```

validation\login.js

```
const Validator = require("validator");
const isEmpty = require("is-empty");
module.exports = function validateLoginInput(data) {
  let errors = {};
  data.email = !isEmpty(data.email) ? data.email : "";
  data.password = !isEmpty(data.password) ? data.password : "";
  if (Validator.isEmpty(data.email)) {
    errors.email = "Email Field is required";
  } else if (!Validator.isEmail(data.email)) {
    errors.email = "Email is invalid";
  }
  if (Validator.isEmpty(data.password)) {
    errors.password = "Password field is required";
  }
  return {
    errors,
    isValid: isEmpty(errors),
  };
};
```

validation\register.js

```
const Validator = require("validator");
const isEmpty = require("is-empty");

module.exports = function validateRegisterInput(data) {
  let errors = {};
  data.name = !isEmpty(data.name) ? data.name : "";
  data.email = !isEmpty(data.email) ? data.email : "";
  data.password = !isEmpty(data.password) ? data.password : "";
  data.password2 = !isEmpty(data.password2) ? data.password : "";
  data.Role = !isEmpty(data.Role) ? data.Role : "";

  if (Validator.isEmpty(data.name)) {
    errors.name = "Name Field is required";
  }

  if (Validator.isEmpty(data.email)) {
    errors.email = "Email Field is required";
  } else if (!Validator.isEmail(data.email)) {
    errors.email = "Email is Invalid";
  }
```

```
  if (Validator.isEmpty(data.password)) {
    errors.password = "Password Field is required";
  }
  if (Validator.isEmpty(data.password2)) {
    errors.password2 = "Password Field is required";
  }
  if (!Validator.isLength(data.password, { min: 6, max: 30 })) {
    errors.password = "Password must be at least 6 characters";
  }
  if (!Validator.equals(data.password, data.password2)) {
    errors.password2 = "Passwords must match";
  }
  if (Validator.isEmpty(data.Role)) {
    errors.name = "Role Field is required";
  }
  return {
    errors,
    isValid: isEmpty(errors),
  };
};
```

views\error.ejs

```
<h1><%= message %></h1>
<h2><%= error.status %></h2>
<pre><%= error.stack %></pre>
```

views\indes.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

views\profile.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>
      Node Js MySQL Fetch and Show Records from MySQL Database Example
    </title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
  </head>
  <body>
    <div class="container mt-4">
      <% if (messages.success) { %>
      <p class="alert alert-success mt-4"><%- messages.success %></p>
      <% } %>
      <br />
      <table class="table">
        <thead>
          <tr>
            <th>#Id</th>
            <th>Name</th>
            <th>Email</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          <% if(data.length){ for(var i = 0; i< data.length; i++) {%>
          <tr>
            <th scope="row"><%= (i+1) %></th>
            <td><%= data[i].name%></td>
            <td><%= data[i].email%></td>
          </tr>
          <% } }else{ %>
          <tr>
            <td>No data ever existed.</td>
          </tr>
          <% } %>
        </tbody>
      </table>
    </div>
  </body>
</html>
```

package.json

```
{
  "name": "ocean_beta_backend",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.20.2",
    "concurrently": "^7.6.0",
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "debug": "^4.3.4",
    "dotenv": "^16.0.3",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-flash": "^0.0.2",
    "express-session": "^1.17.3",
    "express-validator": "^7.0.1",
    "http-errors": "^2.0.0",
    "is-empty": "^1.2.0",
    "jsonwebtoken": "^9.0.0",
    "mongodb": "^6.0.0",
    "mongoose": "^7.0.1",
    "morgan": "^1.10.0",
    "mysql": "^2.18.1",
    "nodemailer": "^6.9.4",
    "nodemon": "^2.0.22",
    "openai": "^3.2.1",
    "passport": "^0.6.0",
    "passport-jwt": "^4.0.1",
    "validator": "^13.9.0"
  },
  "engines": {
    "node": ">=14 <15"
  }
}
```

server.js

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

const mongoose = require("mongoose");
const bodyParser = require("body-parser");
```

```
const passport = require("passport");
const users = require("./routes/api/users");
const app = express();
const cors = require("cors");
require("dotenv").config();
var mysql = require("mysql");
var session = require("express-session");
var flash = require("express-flash");
var userRoute = require("./routes/api/users");
var path = require("path");
var logger = require("morgan");
var cookieParser = require("cookie-parser");
app.set("view engine", "ejs");
app.use(logger("dev"));
app.use(bodyParser.json());
app.use(cookieParser());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, "public")));
const BASE_URL = process.env.BASE_URL;
const { MongoClient, ServerApiVersion } = require("mongodb");
const { modelName } = require("./models/User");

const db = require("./config/keys").mongoURI;

app.options("*", cors());
//
app.use(
  bodyParser.urlencoded({
    extended: false,
  })
);

const url =
  "mongodb+srv://Sameer:1010@cluster0.kq5wym0.mongodb.net/?
retryWrites=true&w=majority";
const client = new MongoClient(url);

// Database Name
const dbName = "test";
async function main() {
  // Use connect method to connect to the server
  await client.connect();
  console.log("Connected successfully to server");
  const db = client.db("test");
  console.log(db, "Sssssssssssssss");
  const collection = db.collection("users");
  console.log(collection);

  // the following code examples can be pasted here...

  return "done.";
}
main()
  .then((res) => console.log(res, "Sssssssss"))
```

```
    .catch(console.error)
    .finally(() => client.close());
  const corsOptions = {
    origin: process.env.BASE_URL,
    credentials: true,
    allowedHeaders: ["sessionId", "Content-Type"],
    exposedHeaders: ["sessionId"],
    methods: "GET,HEAD,PUT,PATCH,POST,DELETE",
    preflightContinue: false,
  }; //
  app.use(
    session({
      secret: "123@abcd",
      resave: false,
      saveUninitialized: true,
      cookie: { maxAge: 60000 },
    })
  );
  app.use(flash());
  // app.use("/", nodeRoutes);
  // app.use("/database", userRoute);
  // app.use(function (req, res, next) {
  //   next(createError(404));
  // });

  app.use(function (err, req, res, next) {
    res.locals.message = err.message;
    res.locals.error = req.app.get("env") === "development" ? err : {};
    res.status(err.status || 500);
    res.render("error");
  });

  app.use(cors(corsOptions));
  app.use((req, res, next) => {
    res.setHeader("Access-Control-Allow-Origin", "*");
    res.header(
      "Access-Control-Allow-Headers",
      "Origin, X-Requested-With, Content-Type, Accept"
    );
    next();
  });
  app.use(bodyParser.json());

  // const db = process.env.DATABASE;
  app.get("/", (req, res) => {
    res.status(201).json("server started");
  });
  var collections = mongoose.connections[0].collections;
  var names = [];
  console.log(collections);
  Object.values(collections).forEach(function (k) {
    console.log(k);
    names.push(k);
  });
```

```
console.log(names);
mongoose
  .connect(db, { useNewUrlParser: true })
  .then((res) => console.log("Mongoose Connected"))
  .catch((err) => console.log(err));
app.use(passport.initialize());

require("./config/passport")(passport);
app.use("/api/users", users);
const port = process.env.PORT || 5001;
console.log(port);
app.listen(port, () => console.log(`server running on port ${port}`));
```