const express = require("express");

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

const ejs = require("ejs");

const validator = require('validator');

const mongoose = require("mongoose");

mongoose.connect("mongodb://127.0.0.1:27017/quizDB");

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static("public"));

app.set("view engine", "ejs"); //tell the app which is generated using express to use EJS as its view engine.

const quizSchema = new mongoose.Schema({

  title: String,

  options: [String],

  answer: String,

});

const Quiz = mongoose.model("Quiz", quizSchema);

const answerSchema = new mongoose.Schema({

  userName: {

    type: String,

    required: [true, "Name is required"],

    maxLength: 30

  },

  userEmail: {

    type: String,

    lowercase: true,

    required: [true, "Email address is requied"],

    validate: [validator.isEmail, 'invalid email']

  },

  userAnswer: [

    {

      questionId: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "Quiz",

      },

      selectedOption: { type: String, required: true },

    },

  ],

  userScore: String,

});

const answerModel = mongoose.model("Answer", answerSchema);

app.get("/", function (req, res) {

    //this is used when user do back it reload the page or removes the data saved that user enters

  res.header('Cache-Control', 'no-cache, no-store, must-revalidate');

  res.header('Pragma', 'no-cache');

  res.header('Expires', 0);

  Quiz.find({})

    .then(function (foundQuiz) {

      if (foundQuiz) {

        // console.log(foundQuiz);

        res.render("quiz", { quizz: foundQuiz });

      } else {

        res.send("Nothing to send please add the Questions");

      }

    })

    .catch(function (err) {

      console.log(err);

    });

});

app.post("/", function (req, res) {

  const ans = req.body.Quiz;

  console.log(ans);

  //here we use req.body so enter title or content in body option(postman) in x-www-forn-urlencoded

  const newTitle = req.body.title;

  const newOption = req.body.option;

  const newAnswer = req.body.ans;

  const newQuiz = new Quiz({

    title: newTitle,

    options: newOption,

    answer: newAnswer,

  });

  // saving the item in database also check for errors

  newQuiz.save()

    .then(function (result) {

      // console.log(result);

      res.send(result);

    })

    .catch(function (err) {

      console.log(err);

    });

});

app.get("/submitQuiz/:data", function (req, res) {

  const [userName, score] = req.params.data.split("-");

  res.render("submitQuiz", { name: userName, marks: score });

});

app.post("/submitQuiz", async function (req, res) {

  const newUser = req.body.userName;

  const newEmail = req.body.userEmail;

  const quizAnswers = req.body;

  const answers = [];

  console.log(quizAnswers);

  for (let questionKeyId in quizAnswers) {

    if (mongoose.Types.ObjectId.isValid(questionKeyId)) {

      answers.push({

        question: questionKeyId,

        userChoiceAnswer: quizAnswers[questionKeyId],

      });

    }

  }

  let count = 0;

  for (let answer of answers) {

    try {

      const result = await Quiz.findOne({ \_id: answer.question });

      if (result.answer === answer.userChoiceAnswer) {

        count++;

      }

    } catch (err) {

      console.log(err);

    }

  };

  const newAnswer = new answerModel({

    userName: newUser,

    userEmail: newEmail,

    userAnswer: answers.map((answer) => ({

      questionId: answer.question,

      selectedOption: answer.userChoiceAnswer,

    })),

    userScore: count

  });

  console.log(newAnswer.userScore);

  console.log(newAnswer);

  newAnswer.save()

    .then(function (result) {

      console.log(result);

    })

    .catch(function (err) {

      console.log(err);

    });

  // res.render("submitQuiz", { name: newUser, marks: count });

  res.redirect(`/submitQuiz/${newUser}-${count}`);

});

app.listen(3000, function () {

  console.log("Server is started on port 3000");

});

  // ////////////// LOGIN ////////////

  // app.get("/login", function (req, res) {

  //   res.render("login");

  // });

// ////////////// SIGN UP ////////////

// app.get("/signup", function (req, res) {

//   //this is used when user do back it reload the page or removes the data saved that user enters

//   res.header('Cache-Control', 'no-cache, no-store, must-revalidate');

//   res.header('Pragma', 'no-cache');

//   res.header('Expires', 0);

//   res.render("signup");

// });

// app.post("/signup", function (req, res) {

//   const newUserName = req.body.userName;

//   const newUserEmail = req.body.userEmail;

//   const newUserPassword = req.body.userPassword;

//   const newUser = new userModel({

//     name: newUserName,

//     email: newUserEmail,

//     password: newUserPassword

//   });

//   console.log(newUser);

//   newUser.save()

//     .then(function (result) {

//       console.log(result);

//     })

//     .catch(function (err) {

//       console.log(err);

//     });

//   console.log(res.status, res.statusCode, res.statusMessage);

//   res.redirect("/login");

// });

// const userSchema = new mongoose.Schema({

//   name: {

//     type: String,

//     required: [true, "Name is required"],

//     maxLength: 30

//   },

//   email: {

//     type: String,

//     unique: true,

//     lowercase: true,

//     required: [true, "Email address is requied"],

//     validate: [validator.isEmail, 'invalid email']

//   },

//   password: {

//     type: String,

//     required: [true, "Password field is required"],

//     maxLength: 15

//   }

// });

// userSchema.plugin(uniqueValidator);

// const userModel = mongoose.model("User", userSchema);

///////////////////////////////////////////////////////////////////////////////////////////

// res.redirect(`/finalsubmit?name=${newUser}&score=${userScore}`)