CS682 – Software Development Lab

Code documentation
For
Surveyfy

By

Karthik Prasad

Guided By

Professor. Kenneth Fletcher

Department of computer Science

University of Massachusetts Boston

CODE DOCUMENTATION

1.1 Project Repository

The project can be downloaded or cloned from the Git Hub Repository with the following link. https://github.com/k-prasad/Surveyfy.git

1.2 Version Control

Git Repo Version Control has been used

1.3 Coding Standard

Google and ESLint coding standard have been used. Semicolon (;) in the entire code of the project is not required as statement terminator. Hence semicolon is not used.

1.4 Coding Indentation

All code files have four soft tab spaces as indentation.

1.5 Code files and function.

surveys

- **edit.ejs** - displays edit survey page.
- **index.ejs** – displays survey information
- **more_details** – generates line graph for individual surveys
- **new.ejs** – displays survey new page
- **report.ejs** – shows survey report page
- **show.ejs** – shows survey information page

respondents

- **end_page.ejs –** displays the end page after the survey is taken
- **respondent_answer.ejs -** This is the welcome page before respondents take

views

- **about.ejs** – displays about page. Displays information about the surveyfy web application and company's team.
- **forgot_password** - functionality to reset forgotten password form. After clicking forgot password. An email with the reset password link is sent to the admin's email address.
- **home.ejs** – home page is displayed. After clicking log in new password will be saved.
- **login.ejs** – login form. Admin log in form with email and password as credentials.
- With additional option of forgot password link.
- **register.ejs** – admin registration. Admin sign up page with all the user details.
- **reset.ejs** - password reset functionality (with a new password). After clicking log in new password will be saved.

questions

- **edit.ejs** - Edit/modify a question functionality. Left side container consisting of question types can be chosen to modify a question. Right side container consisting of question information can be edited to modify the question.
- **new**.**ejs** - Add a question page. Left container allows Admin to choose different question types. Right container is used for entering question details and uploading an image.

ROUTES:

index.js

- router.get("/", function(req, res)
  Displays home page.
- Survey.find({featuredSurveys: true, publicSurvey: true}, function(error, allSurveys)
  Displays all the surveys which are selected as Featured Surveys as well as Public Survey
- router.get("/about", function(req, res)
  Displays about page
- router.get("/register", function(req, res)
  Displays Registration Page
- router.post("/register", function(req, res)
  Posts data from registration form to our user.js model
- User.register(newUser, req.body.password, function(err, user)
  Registers a new user
- router.get("/login", function(req, res)
  Displays Login Page
- router.post("/login", passport.authenticate("local", {
      successRedirect: "/surveys",

```
    failureRedirect: "/login",

    failureFlash: true,

    successFlash: "Welcome to Surveyfy!"

}),function(req, res)
```

User or Admin Logged In to the web app

- router.get("/logout", function(req, res) {
  ```
      req.logout()

      req.flash("success", "Logged Out")

      res.redirect("/")

  })
  ```
  Logouts from the web app
- router.get("/forgot_password", function(req, res)
  Displays Forgot Password page
- router.get("/reset/:token", function(req, res)
  Password Reset confirmation Displayed
- router.post("/reset/:token", function(req, res)
  Password Reset is stored in user model

questions.js

- router.get("/new", middleware.isLoggedIn,function(req, res) {
  ```
      Survey.findById(req.params.id, function(err, survey) {

          if (err) {

              console.log(err)

          } else {

              res.render("questions/new", {survey, survey})

          }

      })

  })
  ```

Show Form to create questions

- router.post("/", middleware.isLoggedIn, upload.single('image'), function(req, res) {

```
  Survey.findById(req.params.id, function(err, survey) { // }) -2
    if (err) {
      console.log(err)
      res.redirect("/surveys")
    } else { // } -3
      cloudinary.uploader.upload(req.file.path, function(result) { // }) -4
        // add cloudinary url for the image to the question object under image property
        req.body.question.image = result.secure_url;
        // add author to question
      //   req.body.question.author = {
      //     id: req.user._id,
      //     username: req.user.username
      //   }
      // create new question
      Question.create(req.body.question, function(err, question) { // -5
        if (err) {
          req.flash("error", "The Create Operation not Successful.")
          console.log(err)
          return res.redirect("back")
        } else {
          // add new question to survey
          // ADD - USERNAME & ID TO QUESTION
          question.author.id = req.user._id
          question.author.username = req.user.username
          question.author.firstName = req.user.firstName
```

```
                    // SAVE - question
                    question.save()

                    survey.questions.push(question)
                    survey.save()
                    // redirect to surveys page
                    req.flash("success", "Successfully Created a Question.")
                    res.redirect("/surveys/" + survey._id)
                  }
              }) // -5

              }) // }) -4

          } // } -3

              }) // }) -2
          }) // }) -1
```

Create an add new question to survey

- router.get("/:question_id/edit", middleware.checkQuestionOwnership, function(req, res){
```
  Question.findById(req.params.question_id, function(err, foundQuestion) {
    if (err) {
      res.redirect("back")
    } else {
      res.render("questions/edit", {survey_id: req.params.id, question: foundQuestion})
    }
  })
```

```
})
```

Question Route Edit

-
  ```
  router.put("/:question_id", middleware.checkQuestionOwnership, function(req, res) {
     Question.findByIdAndUpdate(req.params.question_id, req.body.question, function(err,
  updatedQuestion){
        if(err){
           res.redirect("back");
        } else {
           res.redirect("/surveys/" + req.params.id );
        }
    });
  })
  ```

Question Route Update Functionality

-
  ```
  router.delete("/:question_id", middleware.checkQuestionOwnership, function(req, res) {
     // Find ID and Delete the question_id
     Question.findByIdAndRemove(req.params.question_id, function(err){
        if (err) {
           // Redirects back to the previous page
           res.redirect("back")
        } else {
           req.flash("success", "Question Successfully Deleted")
           res.redirect("/surveys/" + req.params.id)
        }
     })
  })
  ```

Question Route Delete Functionality

surveys.js

-
  ```
  router.get("/", middleware.isLoggedIn,function(req, res) {
    var noMatch = null;
    // To display the Survey searched by the user
    if (req.query.search) {
      // Get all survey from search string
      const regex = new RegExp(escapeRegex(req.query.search), 'gi'); // g - global, i - 
  ignore case
      Survey.find({name: regex}, function(error, allSurveys) {
        if (error) {
          console.log(error)
        } else {


           if(allSurveys.length < 1) {
            noMatch = "No campgrounds match that query, please try again.";


         }
          res.render("surveys/index", {surveys: allSurveys, noMatch: noMatch})
        }
      })
    } else {
      // Get all surveys from database
      Survey.find({}, function(error, allSurveys) {
        if (error) {
          console.log(error)
        } else {
  ```

```
                res.render("surveys/index", {surveys: allSurveys})
            }
        })
    }
})
```

Displays Index page. Which also includes Search Function.

-
```
router.post("/", middleware.isLoggedIn, function(req, res) {
    var name = req.body.name
    var description = req.body.description
    var startdate = req.body.startdate
    var enddate = req.body.enddate
    var author = {
        id: req.user._id,
        username: req.user.username,
        firstName: req.user.firstName
    }

    var topic = req.body.topic

    // Check to ensure if the survey added to featured surveys
    let featuredSurveys;
    if (req.body.featuredSurveys) {
        featuredSurveys = true
    } else {
        featuredSurveys = false
    }
    // Check to ensure if the survey added as a public survey
    let publicSurvey;
    if (req.body.publicSurvey) {
        publicSurvey = true
    } else {
        publicSurvey = false
    }

    // Check to ensure if the respondents should see UI elements or not
    let hideRespondentsNav;
    if (req.body.hideRespondentsNav) {
        hideRespondentsNav = true
    } else {
        hideRespondentsNav = false
    }
    var newSurvey = {name: name, description: description, startdate, enddate, author:
author, topic: topic, featuredSurveys: featuredSurveys, publicSurvey: publicSurvey,
hideRespondentsNav: hideRespondentsNav}
    // newSurvey.author.id = req.user._id
    // newSurvey.author.username = req.user.username
    // newSurvey.author.firstName = req.user.firstName

    // Create a new survey and save
    Survey.create(newSurvey, function(error, newSurveyCreated) {
        if (error) {
            console.log(error)
        } else {
            // redirecting to surveys
            console.log(newSurveyCreated)
            res.redirect("/surveys")
        }
    })


})
```

Posts a new survey into database.

-
  ```
  router.delete("/:id", middleware.checkSurveyOwnership, function(req, res) {
      Survey.findByIdAndRemove(req.params.id, function(err) {
          if (err) {
              console.log(err)
              res.redirect("/surveys")
          } else {
              res.redirect("/surveys")
          }
      })
  })
  ```

  Survye Delete Route function

MODELS:

answer.js

- Schema that create and stores the answer responses from respondents.
  option.js
- Schema to store the options for each question that is provided in the survey.
  question.js
- Schema that create and store the questions into the particular survey that is referred.

survey.js

- Schema that create and store the surveys.

user.js

- Schema that create and store admin user data.