CS615 - Document Assessor Project

Kevin Kinsella
Department of Computer Science
National University of Ireland, Maynooth
Maynooth, Kildare
kevin.kinsella.2020@mumail.ie

Yeabsira Mintesnot Abegaz
Department of Computer Science
National University of Ireland, Maynooth
Maynooth, Kildare
yeabsira.mintesnot.2020@mumail.ie

I. Introduction

This paper with will describe a dynamic web application that allows researchers the ability to create assessment tasks that allows them to automatically assign users to an assessment task. Once an user has been assigned to an assessment task, the user can judge the resource with a free text answer and a rating. The project required developing a data-model alongside a RESTful API and an interface for distributing the document assessment tasks online. The project involved solving the following two user stories:

- Researchers must be able to sign into their account and create/edit/delete assessment tasks. An assessment must consist of a title, description, a link to instructions, the number of links that will be assessed per user. The researcher further creates a list of resources that users must assess (The list of links can be passed through a text file). All assessments are stored persistently in the database.
- Once an assessment task has been created a researcher can assign users to the assessment tasks. The researcher further, creates an assessment form for each of the links the user has been assigned to. The form contains fields to complete each resource with a free-text answer and a rating on a 5-point scale. All forms along with assessments are stored persistently in the database.

II. Methodology

The wide range of technologies chosen in this project where broken up into the following sections front-end components, back-end components, database management and code management. The technologies used are as follows:

- **Front-End Components**: Involved using technologies such as HTML [1], CSS [2], JavaScript [3] and Bootstrap [4].
- **Back-End Components**: Involved using technologies such as NodeJS [5], Express [6], NPM [7].
- **Database Management**: Involved using technologies such as MongoDB [8].
- **Code Management**: Involved using technologies such as GitHub [9].

III. Model View Controller Pattern

The architecture of this application implements the Model-View-Controller [10] (MVC) pattern. The purpose of this pattern is to split an application in specific sections that cover their own purposes. The important thing to note about this architecture is that the model and view never interact with each other and any interactions are done through the controller. Having the controller between the model and the view means having the logic of data and data view completely separate. They are explained as follows:

A. Model

The model is responsible for handling all of the data logic of requests and also interacts with the database (eg. getting, saving, updating and deleting of the data).

B. View

The view is only concerned with how to present the information the controller sends it. It is a template file that dynamically renders HTML [1] based on the information the controller sends it. The view will send back the final presentation back to the controller.

C. Controller

The controller handles the request flow coming from the client side and will tell the rest of the server what to do, it acts as the 'middle-man' between model and view. It first queries the database based on the request and should never directly interact with the data logic, it tells the model what to do. After the model has sent back the data to the controller, the controller interacts with the view to render the data to the user. The view sends back the presentation to the controller, the controller sends the response back to the client.

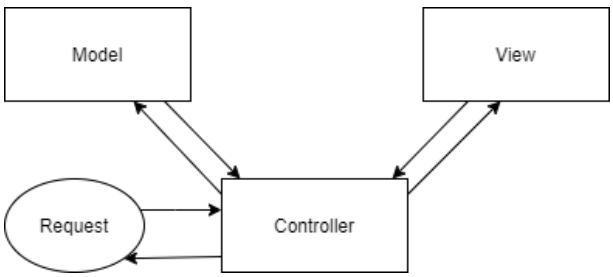

Fig. 1. Model-View-Controller

IV. Database

This section will discuss the model part of the MVC [10] architecture. The data-models defined for this project are: assessment, resource, form, researcher and user. They are described as follows:

A. Assessment Schema

The assessment schema consists of a title, description, instructions, number of links per user, whether the assessment has started, the resources associated with the assessment, the researcher who created the assessment, the total number of links, the total number of links completed and the percentage of links reviewed in the assessment. The resources field is an array of resource schemas which follows a one-to-many relationship between resources and resource. The researcher field is a researcher schema field that stores the id and username of the registered researcher.

TABLE I
Assessment Schema

Field Name	Field Type	Description
title	string	title of assessment
description	string	description of assessment
instructions	string	instructions for assessment
numLinksPerUser	number	assessments per user
started	boolean	has assessment started
resources	array	resource schemas
researcher	object	researcher schema
totalNumLinks	number	total links
totalLinksCompleted	number	reviews complete
percentageComplete	number	% of links complete

B. Resource Schema

The resource schema consists of the assessment task, links associated with the resource, the forms associated with resource, the user who is assigned to the resource and whether the resource has started. The number of links in the links array is defined by the number of links per user defined in the assessment schema above. Similarly the forms field is an array of form schema, which contains a form for each of the links the user must assess. The user field is a user schema field that stores the id and username of the mocked user.

TABLE II
Resource Schema

Field Name	Field Type	Description
task	string	assessment task
links	Array	links user assesses
forms	Array	array of form schema
user	object	user schema
started	boolean	has resource started

C. Form Schema

The form schema consists of the link the user must assess, a free-text answer field, the rating the user gave and whether the review has started.

TABLE III
Form Schema

Field Name	Field Type	Description
link	string	link user reviews
text	string	review user gives
rating	number	rating between 1-5
started	boolean	has review started

D. Researcher Schema

The researcher schema consists of the username and password of the researcher.

TABLE IV
Researcher Schema

Field Name	Field Type	Description
username	string	username of researcher
password	string	password user gives

E. User Schema

The user schema consists of a username of the user.

TABLE V
User Schema

Field Name	Field Type	Description
username	string	username of researcher

F. Database Overview

The diagram below gives an example of the database structure where an assessment has been created with six total links and two links per user. In this scenario, one assessment is created with three resources associated with the assessment in a one-to-many relationship. Each resource contains the two links required for review, the user that is assigned to resource and also an array of forms which contains a review for each of the links the user is required to assess. The user submits each review to a single form.

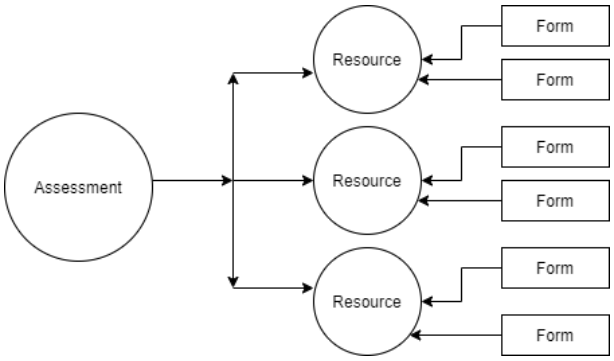

Fig. 2. Database Overview

V. Front End

This section will discuss the view part of the MVC [10] architecture. There is an assessment, resource, form, researchers and partials views.

A. Assessments View

- **Index**: The assessments/index page displays a Bootstrap [4] Jumbotron welcoming to the Document Assessor Application, there is also a button where a researcher can create an assessment this redirects to the assessment/new page. This page also has the title, description, instructions and links per user of every assessment in the database. There is also a view more button to redirect to the assessment/show page of a particular assessment.
- **Show**: The assessments/show page displays a Bootstrap [4] card at the top of the page, it shows the assessment title, description, instructions and links per user and a progress bar with a percentage complete depending on how many reviews have been submitted. If the researcher owns the assessment and the assessment has not started, buttons for add new resource, edit assessment and delete assessment are available, otherwise these are not visible. The page goes on to display another card for each resource in the assessment, here the resource number and links are displayed. If an user has been assigned to a resource a button for start resource is available, otherwise a drop-down menu is available where the researcher can assign an user. If the researcher owns the assessment and the resource has not started, buttons for edit and delete resource are available. Once the resource has started a form is created for each link and is displayed by clicking the expandable button using a Font-Awesome [12] plus icon. Inside the drop-down a form is visible for the user to submit the review and rating. If the review has been submitted, buttons for edit and restart are available to the user.
- **New**: The assessments/new page displays a form where a researcher can create an assessment. The fields displayed in the form are title, description, instruction, number of links per user, a text file upload and a drop down list where a researcher can select single/multiple users for the assessment.
- **Edit**: The assessment/edit page displays a form where a researcher can edit an assessment. The fields display are title, description, instruction, number links per user, a text file upload and a drop down list where a researcher can select single/multiple users for the assessment.

B. Resources View

- **New**: The resources/new page displays a form where a researcher can create a new resource. The fields displayed are instructions and number of links which are both disabled. There is also a text file upload and a drop down list where a researcher can assign a single user. The create resource button submits the form.
- **Edit**: The resources/edit page displays a form where a researcher can edit a resource. The researcher has the option of changing some of links by passing in new links in the text file upload and selecting the links to delete by clicking the check-boxes. The researcher can also assign/unassign users with the drop-down option. The update resource button submits the form.

C. Forms View

- **Edit**: The form/edit page displays a form where a user can edit a form review. The fields displayed are review and rating, along with a update form button which will submit the form.

D. Researchers View

- **login**: The researchers/login page displays a form where a researcher can login to the application. The fields displayed are username and password, along with a login button which will submit the form.
- **register**: The researchers/register page displays a form where a new researcher can register to the application. The fields displayed are username and password. There is also a register button which will submit the form.

E. Partials View

- **header**: The header is displayed on all pages of the application, it contains links to stylesheets, the Bootstrap [4] CDN and a Font-Awesome CDN in the head element. There is a navigation-bar, displaying the application title, register and login buttons. If the researcher is logged in, the researchers username is visible and a logout button is available also.
- **footer**: The footer is displayed on all pages with a copyright for the application.

VI. Backend

This section will discuss the controller part of the MVC [10] architecture. There is an assessments, resources, forms, researchers and assigns controllers.

A. Assessments Controller

- **Index**: The assessment index controller is responsible for finding all of the assessments in the database and displaying the assessments/index view.
- **Show**: The assessment show controller finds said assessment by ID and populates the assessment with the associated resources and forms. Further, it finds all of the users in the database not assigned to assessment (so an user can not be assigned twice) and calls a database method to calculate the number of links that have been reviewed. Finally it passes assessment, users and percentage complete as variables and displays the assessment/show page.
- **New**: The assessment new controller finds all users in the database, then renders the assessments/new page passing users as a variable.
- **Create**: The assessment create controller creates an assessment and the necessary resources. The researcher will pass title, description, instruction, number of links per user, a text file containing links and either none or single/multiple users through the assessments/new form. The number of resources created depends on the number

of links in the text file and the number of links per user. Using these variables an assessment is created and stored persistently in the database. Finally the assessment/show page is rendered showing the newly create assessment.
- **Edit**: The assessment edit controller finds the said assessment by ID and all the users in the database, then displays the assessment/edit page passing the assessment and users as variables.
- **Update**: The assessment update controller deletes the current assessment and associated resources and creates a new assessment exactly like the assessment create route. Finally it redirects to the assessment/show page of the newly created assessment.
- **Delete**: The assessment delete controller simply deletes the assessment and the associated resources.

B. Resources Controller

- **New**: The resource new controller is responsible for finding the assessment by ID and populating the the associated resources. It then finds all of the users in the database not assigned to assessment (so an user can not be assigned twice) and rendering the resources/new page passing the assessment and the filtered users.
- **Create**: The resource create controller creates a new resource and adds it too the assessment to store persistently in the database. The researcher will pass the text file with the links and an user (if researcher wants) through the /resources form. The resource is created and pushed to be associated with the assessment. Finally the controller redirects to the assessment/show page of said assessment.
- **Edit**: The resource edit controller finds the assessment by ID and populates the resources array. It then finds the resources by ID and finds all of the users in the database not assigned to assessment (so an user can not be assigned twice). The resources/edit page of said resource is rendered passing variables assessment, resource and filtered users.
- **Update**: The resource update controller is responsible for updating an existing resource. It first finds the assessment and resource by ID, then modifies the user (if specified) and/or changes the links associated with the resource so all changes are persistent in the database. Finally the controller redirects to the assessment/show page of said assessment.
- **Delete**: The resource destroy controller finds the assessment by ID, reduces the total number of links by the number of links per user and then deletes the resource. Finally redirecting to assessment/show page of said assessment.

C. Forms Controller

- **Create**: The form create controller is responsible for creating a form for each of the links in the resource. First it finds the assessment and resource by ID, sets the assessment to started, then for each of the resource links creates a form and pushes them to the resource form array and saves so the forms are stored in the database. Finally, redirecting to the assessment/show page of said assessment.
- **Edit**: The form edit controller finds the assessment, resource and form by ID, then renders the form/edit page passing assessment, resource and form as variables.
- **Update**: The form update controller updates the users review. It takes the text and rating the user has supplied and updates the database. If the form has not started, the number of completed links in the assessment will increase by one. It then redirects to the assessment/show page.
- **Delete**: The form delete controller allows the user to restart a form. First it finds the assessment and resource by ID, then pulls the form out of the resource form array. It then deletes the form, creates a new form and pushes it to the resource forms array. It will also decrease the total number of links field by one in the assessment. Finally the controller redirects to the assessment/show page of said assessment.

D. Researchers Controller

- **Register - New**: The register new controller renders the researchers/register page.
- **Register - Create**: The register post controller creates a new researcher by username, and registers the researcher in the database using Passport [11] which is an authentication middleware. This method salts the usernames password in the database. The controller then redirects to the researcher/register page if there is an error or redirects to the assessment/index page in the registration was successful and logs the user in.
- **Login - New**: The login new controller renders the researchers/login page.
- **Login - Post**: The login post controller authenticates the researchers using a middleware function from Passport [11] called authenticate. It then redirects to the /assessments/index page if successful or back to the researchers/login page if unsuccessful.
- **Logout - Index:** The researcher logout controller logouts a researcher from the application using a logout function from Passport [11], then redirects to the /assessments/index page.

E. Assigns Controller

- **Create**: The assign controller is responsible for assigning a single user to a resource. It finds the user passed through the form, then finds the user by username in the database and the resource by ID. The controller then sets the user field in the resource to the user found in the database and stores persistently in the database. It then redirects back to the assessment/show page of that resource.

VII. Middleware

This is section will describe the functions in the middleware/index file.

- **asyncErrorHandler:** Handles all promise rejections by resolving and catching the error.
- **isLoggedIn:** Checks if the researcher is logged in. Researchers must be logged in to view the assessments/new and assessments/edit forms and to create, update and delete assessments. They must also be logged in to view the resources/new and resources/edit forms and to create, update and delete resources.
- **checkIfResearcherExists:** Finds the researcher in the database if exists and redirects back to the researchers/register page. This check occurs when a new researcher tries to register.
- **checkAssessmentOwnership:** Checks whether a logged in researcher owns the assessment by comparing the ID of logged in researcher with the researcher ID in the database. The researcher must own the assessment to view the assessments/edit page and to update and delete an assessment, also to view the resource/new and resource/edit pages and to create, update or delete a resource. The controller then redirects back to said assessment if the researcher doesn't own the assessment.
- **checkAssessmentStarted:** Checks the assessment in the database whether it has started or not, redirects back to said assessment if it has started. Researchers are unable to view the assessments/edit page or to update or delete an assessment once it has started.
- **checkResourceStarted:** Checks the resource in the database whether it has started or not, redirects back to the parent assessment if it has started. Researchers are unable to view the resource/edit page or to update or delete a resource once it has started.
- **checkAssessmentCorrectLinks:** Checks whether the number of links evenly divides into the number of links required for the user to assess. This check is applied when the researcher is creating an assessment.
- **checkAssessmentCorrectUsers:** Checks if the researcher has assigned too many users to the assessment. It first finds the number of resources in the assessment and checks if the number of users assigned is greater than the number of resources, if so it redirects back to the assessment/new page.
- **checkAssessmentUpdateCorrectLinks:** Checks whether the number of links evenly divides into the number of links required for the user to assess. This check is applied when the researcher is updating an assessment.
- **checkResourceCreateCorrectLinks:** Checks whether the number of links evenly divides into the number of links required for the user to assess. This check is applied when the researcher is creating resource.
- **checkResourceUpdateCorrectLinks:** Checks whether the number of links evenly divides into the number of links required for the user to assess. This check is applied when the researcher is updating a resource.

VIII. Mock Data

The users in this application are mocked using an NPM [7] package called faker [13], when the application is spun up current users in the database are deleted and new ones are created. The source code is available in seeds/users.js.

References

[1] HTML - Hypertext Markup Language
https://html.com/
[2] CSS - Cascading Style Sheets
https://en.wikipedia.org/wiki/Cascading_Style_Sheets
[3] JavaScript - Programming language
https://www.javascript.com/
[4] Bootstrap - Open source toolkit for developing HTML, CSS and JS projects
https://getbootstrap.com/
[5] Node.js - JavaScript runtime environment
https://nodejs.org/en/
[6] Express - Node.js application framework
https://expressjs.com/
[7] NPM - Node Package Manger
https://www.npmjs.com/
[8] MongoDB - Document based database
https://www.mongodb.com/
[9] GitHub - Software development version control
https://github.com/
[10] MVC - Model-View-Controller
https://en.wikipedia.org/wiki/Model
[11] Passport - Middleware Node.js - for publishing open-source Node.js projects
http://www.passportjs.org/
[12] Faker - NPM package for generating fake data
https://www.npmjs.com/package/faker
[13] Font-Awesome - logos and icons
https://fontawesome.com/