Document Assessor (User View)

Sandya Sebastian
Msc Software Engineering
Maynooth University
Maynooth, Ireland
sandhya sebastian.2020@mumail.ie

Jyothi Sara Thomas
Msc Data Science and Analytics
Maynooth University
Maynooth, Ireland
jyothi.thomas.2020@mumail.ie

***Abstract*—This report illustrates the technical overview of the Document Assessor web application. The web application is developed focusing on functions from a user perspective. The main functionalities of the Document Assessor web app are that the users can sign into their online account and sign up with assessment tasks. Users who have signed up with an assessment task can access the resources within the task, assess the document resources and submit their assessment review.**

Keywords—Assessment task, Resources, Assessment form

I. Introduction

The aim of this document is to provide a technical overview of the Document Assessor web application at the user view perspective. It is a MongoDB data-model which is realisable in JSON, together with accompanying RESTful API in Node Express (NodeJS and Express) and prototype interface with HTML, CSS and JS which demonstrates consumption of the API, for accessing document assessment tasks online. This document will cover the architecture, frontend and backend design, database structure and functionalities of the web app.

II. Application Overview

The Document Assessor website is developed for users to carry different document assessments online which are provided by researchers. Researcher assigns different assessment tasks (that contain various resources) to different users who had once registered. Users can signin to the online account and choose to perform resources(documents) assessments by signing up for the assessment tasks assigned to them. The users should submit an assessment form where they provide feedback about the resources they assessed. This will help the researcher to know the feedback about the resources they assigned. The developed application does not cover the researcher view, i.e. creation and assigning of resources and tasks are not covered. These data are inserted manually to database.

Following are the main components of the developed application:

- Users: Users can register online to get assessment tasks assigned by the researcher. After registration, users can signin to their account using registered credentials.
- Dashboard: Upon login, users can view the doc assessor dashboard which gives links to *all tasks* (tasks assigned by the researcher which are not signed up by the user) and *my tasks* which lists the assessment tasks that are signed up by the user.
- Assessment Task: An assessment task comprises one or more resources which are assigned to a user. Once the task is assigned to user, user can view the assessment title, description, number of resources to be assessed and a link to instructions. Once signed up the users can see the resources of the assessment task from *my tasks* page. Users cannot view the resources that need assessment until they sign up with that assessment task.
- Resources: Users who have signed up with an assessment task can iterate, browse or otherwise explore the assigned resources to perform their assessment. Each resource is given as a link to the document to be assessed.
- Assessment Form: For each resource, users need to complete an assessment form that is submitted to the system. Assessment form includes a text box where users can write a feedback or assessed suggestions about the resource and a rating from 1-5.

III. Application Flow

The flowchart given below explains the application flowchart:

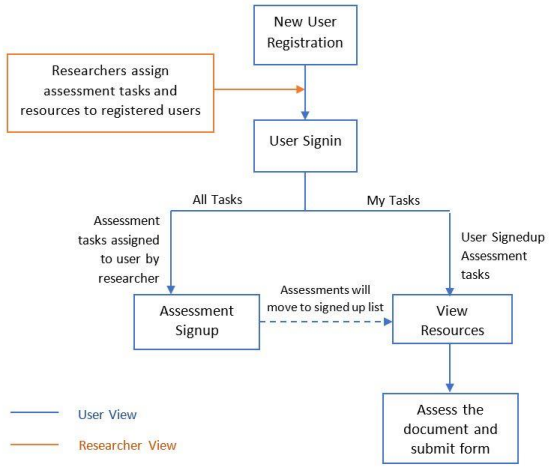

Fig. 1. Application Flow Chart

IV. Application Architecture

The Document Assessor web application is developed using HTML as front end, Node.js as the application runtime and Mongo DB to store the database.

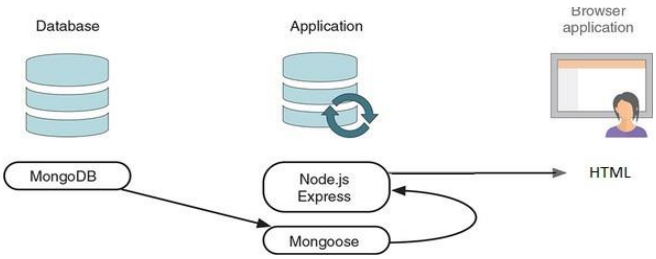

Fig. 2. Application Architecure

V. Front End Design

Html was chosen for frontend as it is easily understandable, and implementation is easy for beginners in web technology. The Html front-end is implemented with Bootstrap 4 framework, CSS stylesheet, javascript and jQuery's. The communication between front-end and back-end is implemented using XMLHttpRequest object.

UI Navigation:

- A user can register in the website through the register interface in the landing page.
- User can sign in from the landing page through the sign in interface. Once signed in user can see the dashboard. From the dashboard the user has two options:

1. Navigate to 'All Tasks' .This page will list all the tasks which is assigned to the user by the researcher. The user can 'sign up' for the resource by clicking the signup button against each task. In this list the user will not be able to access the resources of the assessment task.

2. Navigate to 'My tasks'. It is the list of tasks the user has already signed up. This list displays the title, description, the number of resources and link to the instructions of each task. From the 'My tasks' page the user can navigate to the list of resources, select and view the resource and submit an assessment for the resource.

VI. Back End Design

Node.js is used as runtime environment to build the DocAssessor web application.

A. *Node Express*

ExpressJS is used as framework in NodeJS and Mongoose as mongoDB ODM for NodeJS. Other packages used are body-parser, bcryptjs, ejs, express-validator and jsonwebtoken.

MVC (Models, Views, Controllers) design pattern along with Routes are used for organizing the application.

- Models: Models contains all the javascript codes for the database models, i.e. users, assessments, resources and forms.
- Views: View holds the html and javascript scripts for the front-end layout.
- Controllers: These are the logic of how the app handles the incoming requests and outgoing responses. The controller methods are organized into three files for user, assessments and resources.
  - The user controller holds methods for user registration (receive json and store data to users collection in database), user login and user sessions.
  - The assessment controller holds methods for retrieving assessment lists from assessments collection in database and updates the collection when user signup for an assessment task.
  - The resource controller holds methods for retrieving resource list from resources collection from database and stores the assessment forms associated with each resource submitted by users.
- Routes: Routes are guide which tell the client (browser) to go to which Controller once a specific url/path is requested.

B. *Database*

The database DocAssesssor for the web application is cloud hosted in MongDB Atlas. For user view of the Doc Assessor, 4 collections are used.

Collection	Key	Description
users	_id	Object Id(auto generated primary key)
	firstName	First Name of the user
	lastName	Last Name of the user
	userName	Preferred username
	email	email id (Should be unique)
	password	password
	createdAt	registered time(auto-updated during registration)
assessments	_id	Object Id(auto generated primary key)
	title	title of task
	description	description of task
	numAssessmentsPerUser	number of resources to be assessed
	instructions	link to instructional material
	resources	array of resources object ID from resources collection
	userSignup	Boolean for whether user signed up for assessment task
resources	_id	Object Id(auto generated primary key)
	task	title of resource
	link	url/link to resource to be assessed
	user : id	object Id from users collection to whom the resource is assigned
	user : userName	userName from users collection to whom the resource is assigned
	submissionStatus	Boolean for whether assessment form submitted
	form	object Id from forms collection if submission status true
forms	_id	Object Id(auto generated primary key)
	text	Feedback provided by user for the resource
	rating	Rating given by user for the resource (1-5)

Fig. 3. Database Structure

VII. Hosting The Application

A. System Pre-requisites

The system pre-requisites for hosting the Doc Assessor web application are:

- Npm v6.13.6 or higher
- NodeJS v13.8.0 or higher
- Internet connection (to connect to cloud db)

B. Data Pre-requisites

As the researcher view is not included within scope of the developed app, the data for assessments and resources collections needs to be injected manually or using any data generation packages to the DocAssessor database in MongoDB.

The IP address of the host system needs to be whitelisted to the DocAccessor IP list in Mongodb.

C. Application Hosting

- To host the application, navigate to the app folder DocAssesor.
- Do **npm install** for installing all the project dependencies
- Then **node app** to get the app running on local host on port 4000.
- Go to http://localhost:4000/ from web browser.

References

[1] https://www.w3schools.com/js/ ,w3schools javascript tutorial.

[2] https://www.w3schools.com/html/, w3schools html tutorial.

[3] https://www.w3schools.com/css/, w3 schools css tutorial.

[4] https://www.w3schools.com/bootstrap4 , w3schools bootstrap tutorial.

[5] https://codeburst.io/writing-a-crud-app-with-node-js-and-mongodb-e0827cbbdafb/ ,Eslam Maged, crudapp with node and monogd tutorail.

[6] https://medium.com/gce-neiist/the-mean-stack-a-practical-example-gce-thesis-ca50af9f1ea8/ ,Rafel Velchior, mean-stack example

[7] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/, send and receive data to server using XMLHttpRequest.