TK Internet Praktikum

TECHNISCHE UNIVERSITÄT DARMSTADT

Team Nine - Conference Management System
M.E.A.N. Stack based web application

Name: Dhanasekar Boopalan
Matriculation Number: 2863180

Name: Sheelendra Singh
Matriculation Number: 2412861

Name: Dhanashree Joshi
Matriculation Number: 2594064

Name: Mohd. Danish
Matriculation Number: 2756420

Name: Shashank Singh
Matriculation Number: 2678605

1. Introduction

CMS (**C**onference **M**anagement **S**ystem) is a web application based on M.E.A.N. (**M**ongo-**E**xpress-**A**ngular-**N**ode) stack technologies for an end-to-end solution.

2. Technical Documentation

2.1. Technical Architecture

2.1.1. What is M.E.A.N.?

M.E.A.N stands for

1. MongoDB
   a. Which is a NoSQL database and more scalable than relational databases. It replaces the traditional relational-table based approach to JSON objects with dynamic schemas. Different Mongoose APIs can be used to perform the CRUD(Create-Read-Update-Delete) operations for MongoDB.
2. Express framework
   a. Express is a web application server framework for node.js which acts as a middleware between HTTP requests and server side responses. Also, provides routing mechanisms to bind client side controllers with their respective server side services.
3. Angular JS
   a. Dynamic web pages can be made with the use of Angular JS technology, which provides us with features like two-way data binding and dependency injection for preparing a cutting edge client side.
4. Node JS
   a. It is a Javscript based runtime environment providing with a node-package-manager (npm) giving access to the open-source libraries help us in preparing reliable web applications.

In our project we have used the following technologies/libraries on server and client side.

2.1.2. Server Side

#	**Dependency**	**Description**
1	Mongoose	An object modelling package for node.js
2	Promise	To handle asynchronous calls
3	body-parser	To parse the html files
4	io.sockets	To implement user-chat feature in the web application

5	Fs-extra	To access file-system in Node JS
6	Express	Web application framework for node.js
7	mongoose-q	To handle asynchronous calls (promises)
8	method-override	To override HTTP verbs
9	multer	A Node JS middleware for handling multipart/form data
10	Node-mailer	Send an e-mail with Node JS
11	Smtp transport	Component of Node mailer

2.1.3. Client Side

#	**Dependency**	**Description**
1	Angular	To apply MVC pattern on client side
2	angular-route	For client side page routing
3	angular-socket-io	To implement chat module
4	angular-ui-router	For client side routing
5	angular-xeditable	To perform all CRUD operations inline in a web page
6	bootstrap	To apply responsive CSS properties
7	font-awesome	Use icons in various pages
8	highcharts	To implement chart feature in the application
9	jquery	Client side scripting
10	socket.io	To implement chat application
11	socket.io-client	For chat application

2.1.4. Some important files used in the project

app.js
This is a server side file and contains the mongoDb connect URL, application general error handling, setting application route end-points to be used for routing client requests to specific URL get or posts.

www
This file contains code to create the server, define port.

passport JS
A Node JS based Authentication strategy, and used for session management, can be used with any Express-based web application.

Significance of package.json
All server related dependencies like body-parser, mongoose, socket.io etc. which we are using in the application can be saved in this file.
We need to use the command "npm install" to add these dependencies in our project and "npm install -g --save" to install (globally) and save dependencies to package.json

Significance of bower.json
All client side dependencies like angular-ui-router, angular-xeditable etc are contained in this file. We need to use the command "bower install" to add these dependencies in our project.

Significance of .gitignore file
.gitignore file contains an updated list of project files which we would like to ignore while doing a **git push**.

What's an **appRoutes.js?**
This is a client side (angular) configuration file where we define an angular module as collection of routes i.e. link the html with their respective controller and the URL which invokes them.

app.js (client side)
An angular module where we mention all the services, controller names and certain other functions (needed to be called runtime) in the HTML files.

2.1.5. Database Models

Following database models have been used in our application

#	**Schema Name**	**Description**
1	User	Stores user details along with myConference (referring Conference schema) and mySubmission (referring Submission schema)
2	Conference	Stores conference details along with conferenceMembers and createdBy(referring User schema), conferenceSubmission(referring Submission schema)
3	Submission	Stores fields specific to each submission object along with submittedBy (referring User schema) and reviewId(referring Review schema)
4	Review	Stores fields specific to each review object along with submissionId(referring Submission schema), reviewerId(referring Review schema) and conferenceId(referring Conference schema)

2.1.6. "routes" Folder

"routes" folder contains javascript (.js) files containing different HTTP GETs and POSTs requests with respect to various front-end service calls to different database schemas using Mongoose APIs.

The files here contain various database models which we would like to use to get/update/delete data from. For example, *submission.js* file uses following mongoose models for object modelling the MongoDB documents: User, Conference, Review models and itself.

We then implement each of these HTTP GETs and POSTs with the corresponding Mongoose API enclosed and called with respect to each of the data models (schemas).

Similarly, other javascript (.js) files are responsible for their set of database operations as listed below:

#	**File Name**	**Description**
1	authenticate.js	Server side handling the Authentication functionality of users.
2	conference.js	Various operations related to Conference model.
3	editChairConference.js	Various operations related to Chair-conference model.
4	index.js	Home page of the application.
5	privilege.js	Handling privilege related operations for normal users.
6	profile.js	User profile related operations are handled here.
7	review.js	Review related operations are defined here.
8	submission.js	Submission operations with respect to Conference by the users are defined here.
9	users.js	Route URL defined here.

2.1.7. "uploads" folder

All the uploaded submissions (PDF files) are stored locally here. The corresponding upload functionality is achieved through the installed Multer library using npm.

2.1.8. "public" folder

All the client side javascript and HTML files are stores here. This folder further comprises of a **JS** folder which consists of two more folders- (1) controllers and (2) services. The respective Controller and Service (.js) files are contained in them.

2.1.9. Important Angular Concepts

Angular controllers

Angular controllers are used to handle views to which they are attached. We use the $scope inbuilt variable to do a two-way binding, to fetch/populate the HTML fields in the front-end with the data from the server/database.
Apart from $scope, angular controllers can make use of other important variables like **$stateParams** (to get request parameters), **$stateProvider** to provide state change to a different HTML, **ui-sref** directive to bind a link with the corresponding HTML page. Usage of **$q** and **promise** to handle asynchronous calls properly. For example, the ConfControllerModule makes use of **$scope**, **$rootScope** (to access currently logged-in user, that is stored through checkLoggedin function in appRoutes.js) and the services createConference(), ListConferenceNormal(), ListConferenceChair(), Join() to be performed using the corresponding ConfService variable declared in the angular.module arguments.

Angular Services

Angular services are used to send HTTP GET and POST requests to the server side which are handled by corresponding javascript (.js) files contained in the **routes** folder, explained earlier in section 2.1.6. For example, ConfServiceModule in which CreateConference function with post to "/conf/createConf" is exposed, and with the help of router.route functionality is directed to corresponding URL and function signature in conference.js file in "routes" folder to access operations related to conference schema.

Angular views

These are simple HTML files which makes use of the corresponding controller defined in the appRoutes.js file. For example, to bind the "Create Conference" link in home.html to the corresponding HTML view makes use of ui-sref directive (direct to the respective ".state") and corresponding Controller (.js) file to be loaded when called upon, is defined in appRoutes.js:

```
.state('home.create_conf', {
    url: "/createconf",
    templateUrl: 'views/createConferenceChair.html',
    controller: 'ConfController',
    resolve:{
        logincheck: checkLoggedin
    }
})
```

2.1.10. App.js and AppRoutes.js files

app.js…
All the angular module dependencies such as the controller, service modules and other dependencies need to be mentioned in the **app.js** file (client-side).
Here is a list of some of the angular dependencies that we have defined in this file, along with the theme declared for the usage of editable-options in front-end (to the user):

```
angular.module('cms', ['ui.router','xeditable','angularMoment','ui.bootstrap','ngRoute',
'appRoutes','LoginControllerModule','LoginServiceModule',

'UserControllerModule','UserServiceModule','HomeControllerModule','HomeServiceModule','PreReqS
erviceModule','PreReqControllerModule',
'editUserControllerModule','RemoveReqControllerModule','ConfControllerModule','ConfServiceModu
le','ConfControllerNormalUserModule',

'ConfServiceNormalModule','MyConfServiceModule','MyConfControllerModule','SubmissionController
Module','SubmissionServiceModule'
]).run(function(editableOptions,$rootScope) {
    editableOptions.theme = 'bs3';
    $rootScope.value = {
        getId: function(row) {
            return row._id
        }
    }
});
```

appRoutes.js…
All the front-end resolvable states, with URL-frontendView-controller, injected into **ui-view** div tag of index.html are defined here.
checkLoggedin function to check whether the user is authenticated or not and setting the **authenticated** and **user** rootScope variable based on their authentication status is contained here.

2.1.11. Controllers

#	Controller file name	Functionality
1.	ConfControllerModule	Controller to handle conference related operations such as ListConference, createConference, Join.
2.	ConfControllerNormalUserModule	Conference related operations for a normal user are defined here.
3.	editUserControllerModule	Profile related modifications handled here.
4.	HomeControllerModule	Logout() and isRole()-to check the privilege status of user, functions defined here.
5.	LoginControllerModule	User authentication functions, login defined here.
6.	MyConfControllerModule	Handling data to be displayed in My Conferences page.
7.	PreReqControllerModule	Admin related operations defined here.
8.	RemoveReqControllerModule	Admin related My Conferences page operations defined here.
9.	ReviewControllerModule	Reviewer related functionality defined here.
10.	SubmissionControllerModule	Submission related operations such as saveSubmission, upload pdf, withdraw are contained here.
11.	UserControllerModule	Register user functionality defined here.
12.	ChartControllerModule	Code to display the overall status of different conferences using Highcharts.

13.	ChatControllerModule	Code to create conference specific chats contained here.

2.1.12. Service classes- corresponding to the controllers

Middleware to handle HTTP based GET/POST operations.

#	**Service file name**	**Functionality**
1.	ConfServiceModule	Middleware service class corresponding to ConfControllerModule file.
2.	ConfServiceNormalModule	Middleware service class corresponding to ConfControllerNormalModule exposing corresponding operations to the POST URL to be matched.
3.	HomeServiceModule	Middleware service class corresponding to HomeControllerModule class.
4.	LoginServiceModule	Middleware service class corresponding to LoginControllerModule.
5.	MyConfServiceModule	Middleware service class corresponding to MyConfControllerModule file.
6.	PreReqServiceModule	Middleware service class corresponding to PreReqControllerModule file.
7.	ReviewServiceModule	Middleware service class corresponding to ReviewControllerModule file.
8.	SubmissionServiceModule	Middleware service class corresponding to SubmissionControllerModule file.

9.	UserServiceModule	Middleware service class corresponding to UserControllerModule file.
10.	ChatServiceModule	Middleware service class corresponding to ChatControllerModule file.

3. User Manual

3.1. Setting up the initial Application.

3.1.1. Creation of Admin user

Following are the steps to create the Admin user:

1. We need to hardcode the Admin, change the following code in passport-init.js:
```
newUser.privilege = "admin";
newUser.status = "admin";
```
2. Run the application from Terminal by executing “npm start”
3. Run the MongoDb services:
    1. Either by starting the Mongo Db service from Services Windows application
    2. Or by executing “mongo.exe” and subsequently “mongod.exe” from MongoDb/bin folder.
4. Hit the browser with URL: http://localhost:3000
5. Click on “Register” and enter the Admin details.
6. Click on “Register” button here.
7. The user will get an e-mail.
8. By clicking on the e-mail the user will finally be created.
9. Alternatively, you can check in the user collection created in the database using Mongo Chef.

3.1.2. Registration of normal users

1. Change the following code in passport-init.js to:
```
newUser.privilege = "normal";
newUser.status = "pending";
```
2. Run the application “npm start” in the terminal.
3. Hit the URL http://localhost:3000

4. Click on Register button.

5. Enter user details and click on Register button at the bottom.

6. User would receive an e-mail for verification of his/her details.
7. Click on the verification email link to successfully verify the user details.

8. After email is verified log-in window will be re-opened.
9. User can now enter his/her credentials to log-into the application.

3.1.3. Normal users asking for privileged access from Admin

1. Start the application.
2. Log-in to the application with the normal user credentials.
3. Click on “Request for Privilege” side-bar menu option/link.
4. Provide your comments in the comments section and click on Submit.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.4. As an Admin→Grant privileges

1. Run the application, login page will appear.
2. Log-in as Admin.
3. Click on “New Requests” from side-bar menu.
4. Accept/Reject the requests based on the requirements.
5. If you Accept, then the user will become the Chair.
6. For Rejection, the Admin needs to provide his/her comments.
7. For removing duplicate privilege requests, you can check the “Manage Privileges” side-bar menu.
8. Click on Submit button to remove the request here.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.5. As a Chair→ Create Conference

1. Log-in as Chair, the one whose request for privilege was accepted by Admin.
2. Click on “Create Conference” side-bar menu.
3. Enter the details such as conference name, description, submission, review deadline.
4. Click on “Create Conference” button.
5. A success message will be printed.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.6. As a Chair→ View Conference details

1. Login to the application with Chair credentials.
2. Click on "My Conferences" side-bar menu.
3. You would see a list of conferences.
4. Click on "View" link under the action column.
5. A new view showing the details for that conference would be displayed.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.7. As a Chair→ Conference page operations

1. Login to the application with Chair credentials.
2. Click on "My Conference" side-bar menu.
3. You would see a list of conferences→click on View link
4. To close the submission→Click on the Close Submission button (just below the conference name title). "Review Stage*" message is shown here.
5. On the same page→ change/modify Submission End date /(or) Review End date→ click on the dates to change them inline.
6. On the same page→Auto Assign Reviewers: Click on Auto Assign button.
7. On the same page→Manual Assignment of Reviewers: Click on the Assign button.
8. On the same page→Show Conference, Author details: Click on the conference and Author name links. A modal window will appear on the screen.
9. On the same page→To close the Review stage: Click on Close Review button (just below the conference name title). "Ended*" message will be shown here.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.8. As a Chair→Review submission

1. Log-into the application.
2. Click on “My Conferences” side-bar menu.
3. It will display the conferences created by the logged-in chair person.
4. Click on the “View” hyper-link under the “Action” column.
5. If submission end date > today’s date, then a new view with “Submission Stage” message will be displayed.
6. User can either Auto assign or manually Assign the Reviewers for the conference on this page.
7. Now, the Action would be changed to “Assigned” status under the Action column.
8. On the same page, user can edit the submission end date and/or review end date.
9. Now, click on “Close Review” button to end the review.
10. The status will be changed to Ended* under the conference name at the top-left position of screen.
11. User can now either Accept/Reject the submissions displayed on the screen.
12. An E-mail would be sent to Author (of the submission).
13. Also, chair can “Withdraw” by clicking on the Withdraw button corresponding to each submission.
14. An E-mail would be sent to the Author (of the submission).
15. Also, chair can view review details by clicking on the reviewer email link under the “Reviewer” column.

3.1.9. As a Chair→Chart and display conference reports

1. Login to the application
2. Click on “Charts & Reports” side-bar menu.
3. A pie-chart showing chair-specific created conferences will be displayed.
4. Click on any of the conferences on pie-chart to display a bar-chart.
5. The bar-chart would be separate for submission and review statuses.
6. Also, user can download either or both of the pie and (or) bar-charts in 4 different file formats.

3.1.10. As a Normal user→performing Conference operations

1. Login to the application.
2. Request for privilege→already explained in section 3.1.3.
3. See all conferences→to see all the conferences click on the “All Conferences” side-bar menu.

4. To join a conference→click on the Join button under the Action column.
5. Joined conferences will be displayed as Joined under the Action column.
6. To see the conference details→Click on the More Info under the View Details
7. On the same page, to list the conferences joined by the user→click on "My Conferences" side-bar menu.
8. On the same page, to see the document submissions for one conference→click on the "My Submission" link in the list under the Submission column.
9. A new view showing the conference submission would be displayed.
10. The user can Download any previously uploaded submission i.e. the PDF file.
11. Upload a new PDF file→click on Choose File button to browse for the file.
12. Save the submission→click on Save button.
13. Submit the submission→click on Submit button.
14. Withdraw the submission→click on the Withdraw button.
15. The user can also see the Review Details on the same page.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.

3.1.11. As a Normal user (assigned as a reviewer)→Provide reviews and comments

1. Log-into the application.
2. Click on "My Conferences" side-bar menu link to display the conferences.
3. User can click on "Review Assigned" link to provide his/her reviews for the submission.
4. This will open a new view, where user can Download the submission (i.e. PDF) by clicking on the "Download Submission" button.
5. User can provide his/her reviews for this submission.
6. Click on "Submit" button.

3.1.12. As a Normal user→Conference specific chat

1. Log into the application.
2. Click on “Chat” side-bar menu.
3. A list of “joined” conferences will be displayed as hyper-links.
4. Click on the desired conference name to open a chat window specific to that conference.
5. The chat window will have a Send ! and a Close Chat button to send text and close the window respectively.

Logout from the application

1. Click on icon on top-right.
2. The user will be redirected to the Login Page.