

|  |  |
| --- | --- |
|  |  |
|  | CONFERENCE  MANAGEMENT  SYSTEM  Internet Praktikum TK |
|  |  |
|  |  |
|  | Amber Moien  Prabhjot Singh  Pradeep Baradur  Sandeep Subedi  Sanjay Subedi |
|  |  |

# Table of Contents

Table of Contents i

1. Introduction 2

1.1. Overview 2

1.2. Goals 2

2. Architecture 3

2.1. Overall Architecture 3

2.1.1. Mean Stack 3

2.1.2. MongoDB 3

2.1.3. Express 3

2.1.4. AngularJS 4

2.1.5. Node.js 4

2.1.6. Additional Tools 4

2.2. Component Interaction 5

3. Project Components 6

3.1. Front-end 6

3.2. Back-end 6

3.3. Database Schema 7

4. Use Case Scenarios 1

5. Developer Guide 2

5.1. Getting started 2

5.2. Project Structure 3

6. Summary 5

7. Future Work 5

8. References 5

List of Figures 5

List of Abbreviation 5

# 

# Introduction

## Overview

Objective of this document is to explain the Conference Management System (CMS)—a web based application. The document explains all the components of CMS in detail. It also discusses the issues faced during the design and development phase of CMS along with the steps taken to resolve them.

This document can serve as a user guide and a technical guide for developers.

## Goals

Any **User** of CMS can:

* Register to the system by giving relevant information and selecting a role (Author or/and Reviewer).
* Login to system.
* View/Edit personal details.

The **Author** objectives for the CMS are to:

* Can make a research paper submission against a topic
* Can save the submission as a draft before paper submission
* Can edit the draft before making final submission
* Can view the submission once the paper is submitted
* Can make only one submission against a topic
* Can withdraw the submission within a specified time period (i.e. before review period starts)
* Can only submit within valid time duration (i.e. after the conference has started and review is not opened)

The **Reviewer** objectives for the CMS are to:

* Can submit a review against a paper
* Can edit the review draft before making final review submission
* Can view the submitted review once the review is finally submitted
* Can submit only one review against a paper
* Can edit the review until review deadline is closed

The **Chair** objectives for the CMS are to:

* Can create a conference
* Can open a conference for making submissions
* Can close a conference
* Can extend the deadline for a conference
* Can assign a reviewer/s against a paper
* Can view the list of submitted papers
* Can view the list of submitted reviews

# Architecture

## Overall Architecture

### Mean Stack

### 

[](http://mean.io/)

EAN is a framework for an easy starting point with [MongoDB](https://www.mongodb.org/), [Node.js](http://www.nodejs.org/), [Express](http://expressjs.com/), and [AngularJS](https://angularjs.org/) based applications. It is designed to give you a quick and organized way to start developing MEAN based web apps with useful modules like Mongoose and Passport pre-bundled and configured. We mainly try to take care of the connection points between existing popular frameworks and solve common integration problems.

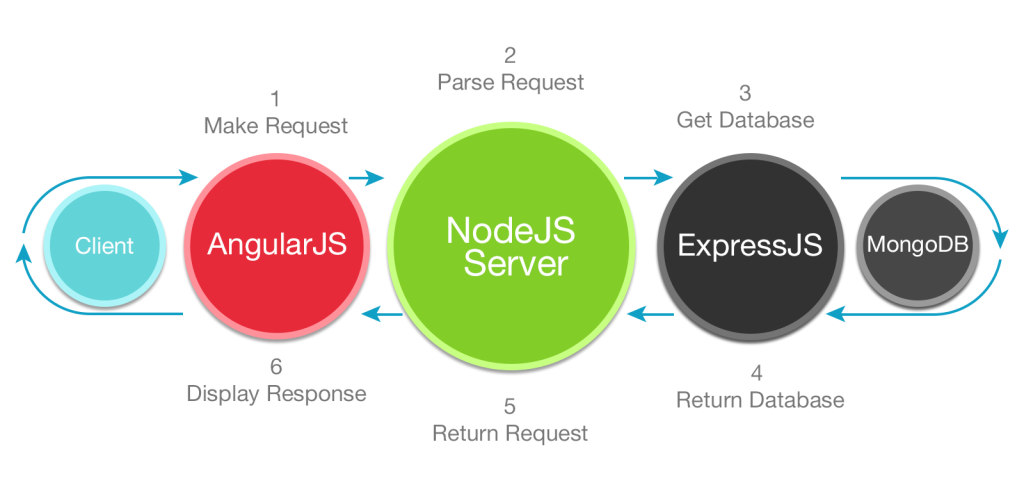


Figure 1: MEAN Stack Framework

### MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

* Documents (i.e. objects) correspond to native data types in many programming languages.
* Embedded documents and arrays reduce need for expensive joins.
* Dynamic schema supports fluent polymorphism.

### Express

Express is a great web development framework for node.js. It provides easy access to stuff like routing, requests and sessions. socket.io is an abstraction layer for Websockets, with Flash and XHR fallbacks, that runs in both node.js and the client.

### AngularJS

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

### Node.js

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

### Additional Tools

* [Mongoose](http://mongoosejs.com/) - The mongodb node.js driver in charge of providing elegant mongodb object modeling for node.js
* [Angular-Moment](https://github.com/urish/angular-moment) - Full featured date library for parsing, validating, manipulating, and formatting dates.
* [Angular-ACL](https://github.com/mikemclin/angular-acl) – A service that allows protect/show content based on the current user's assigned role(s), and those role(s) permissions (abilities).
* [Angular-nvd3](http://krispo.github.io/angular-nvd3/#/) - An AngularJS directive for NVD3 re-usable charting library (based on D3) which helps customize your charts with ease.
* [Twitter Bootstrap](http://getbootstrap.com/) - The most popular HTML, CSS, and JS framework for developing responsive, mobile first projects.
* [UI Bootstrap](http://angular-ui.github.io/bootstrap/) - Bootstrap components written in pure AngularJS

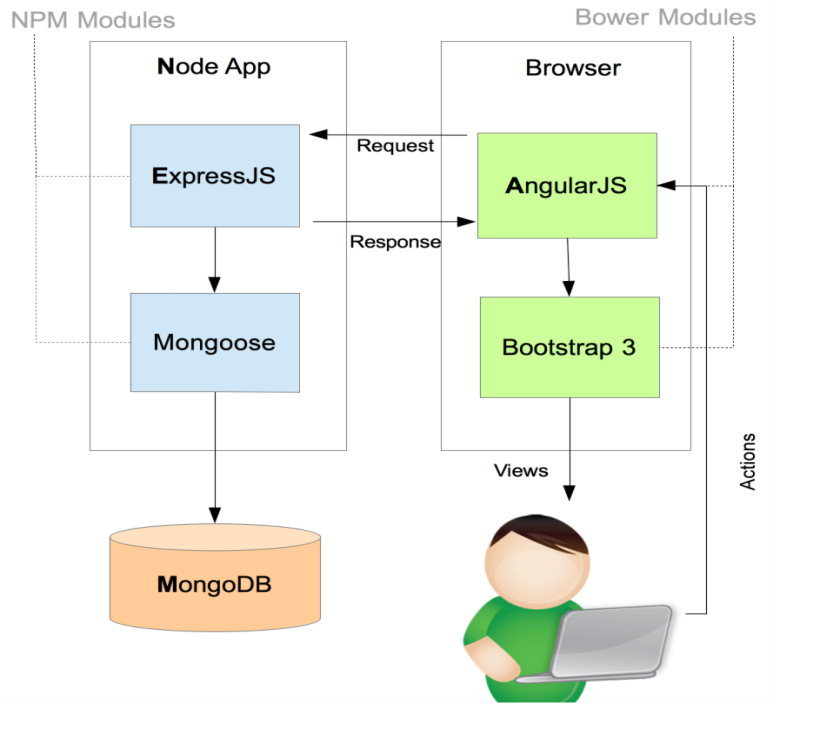


Figure 2: MEAN Stack Application Architecture

## Component Interaction

**Client Side:**

Controllers are main components of Mean Stack (especially Angular). Each view is connected to a controller. Routes are essential to be defined in order to bind the view with its relevant controller. The figure 3 below represents the full work flow of the Stack.

**Server Side:**

Routes are defined under route folder. Route specifies the path a controller/model is supposed to follow. Routes are essential to be defined in order to bind the view with its relevant controller.

The figure 3 below represents the full work flow of the Stack.

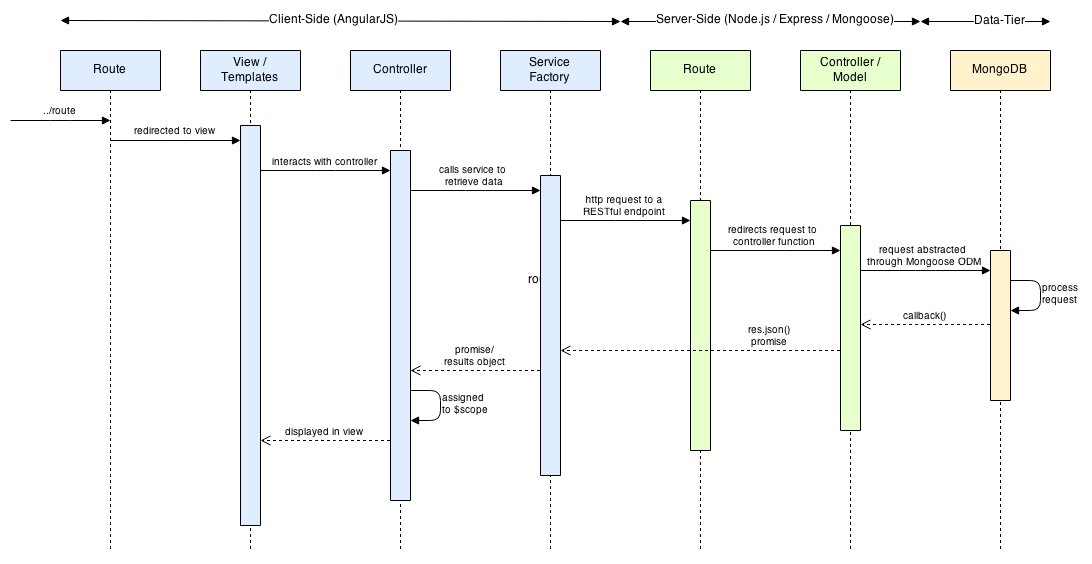


Figure 3: MEAN Stack Component Interaction

# Project Components

## Front-end

The frond-end of the application mainly consists of different client side technologies such as AngularJS, JQuery, Bootstrap and CSS (Cascading Style Sheets). We chose these technologies to provide better User-experience (UX). AngularJS provides the opportunity to extend HTML data attributes by using **ng attributes**. AngularJS incorporates the famous MVC (Model-View-Controller) pattern and also helps developer to build a Single-Page application without hassle. Furthermore, we used Bootstrap library make the application responsive and we have followed ‘Mobile-First’ approach.

For authentication Angular-JWT library is used which authenticates the used based on tokens and Angular-ACL library for authorization purpose. For ex. an ‘Author’ should not be able to access the ‘Chair’ users’ features.

## Back-end

In order to provide stable and effective server for the application, we have used services of NodeJS.

Node.js is a JavaScript runtime built on [Chrome's V8 JavaScript engine](https://developers.google.com/v8/). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](https://www.npmjs.com/), is the largest ecosystem of open source libraries in the world. Furthermore, the server code is managed by Express JS, the backend part of MEAN Stack, which allows the designing of single page web application with minimal effort and efficient features for plug-in development.

Node.js basically provides three things

1. Bindings to the system it runs on
2. An event loop
3. A thread pool

With these components combined we get a platform that supports non-blocking I/O through asynchronous programming and significant better performance compared to traditional architectures.

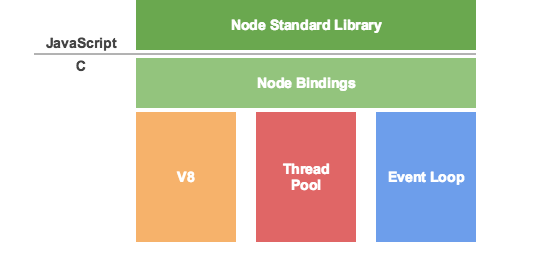


Figure 4: Node JS Architecture

## Database Schema

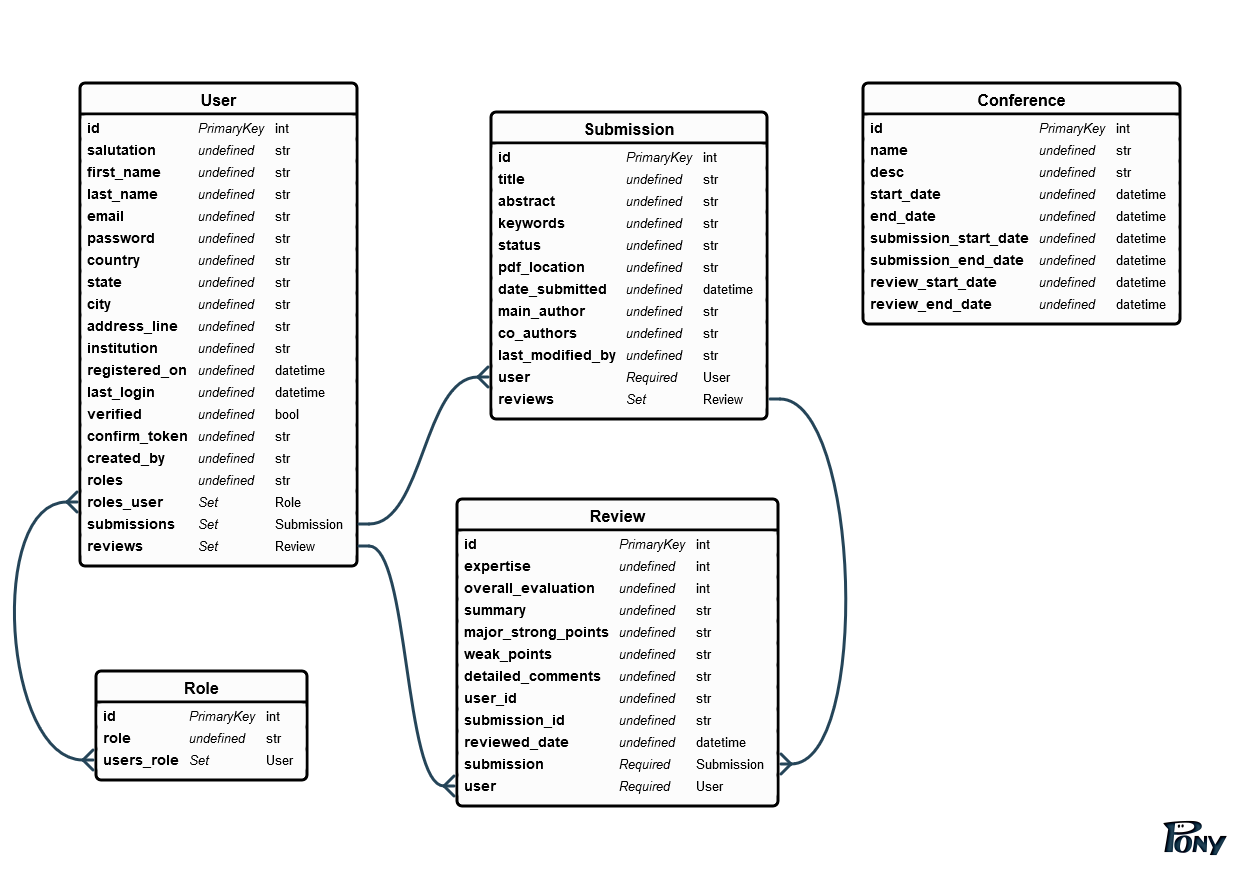


Figure 5: Database Schema

# Use Case Scenarios

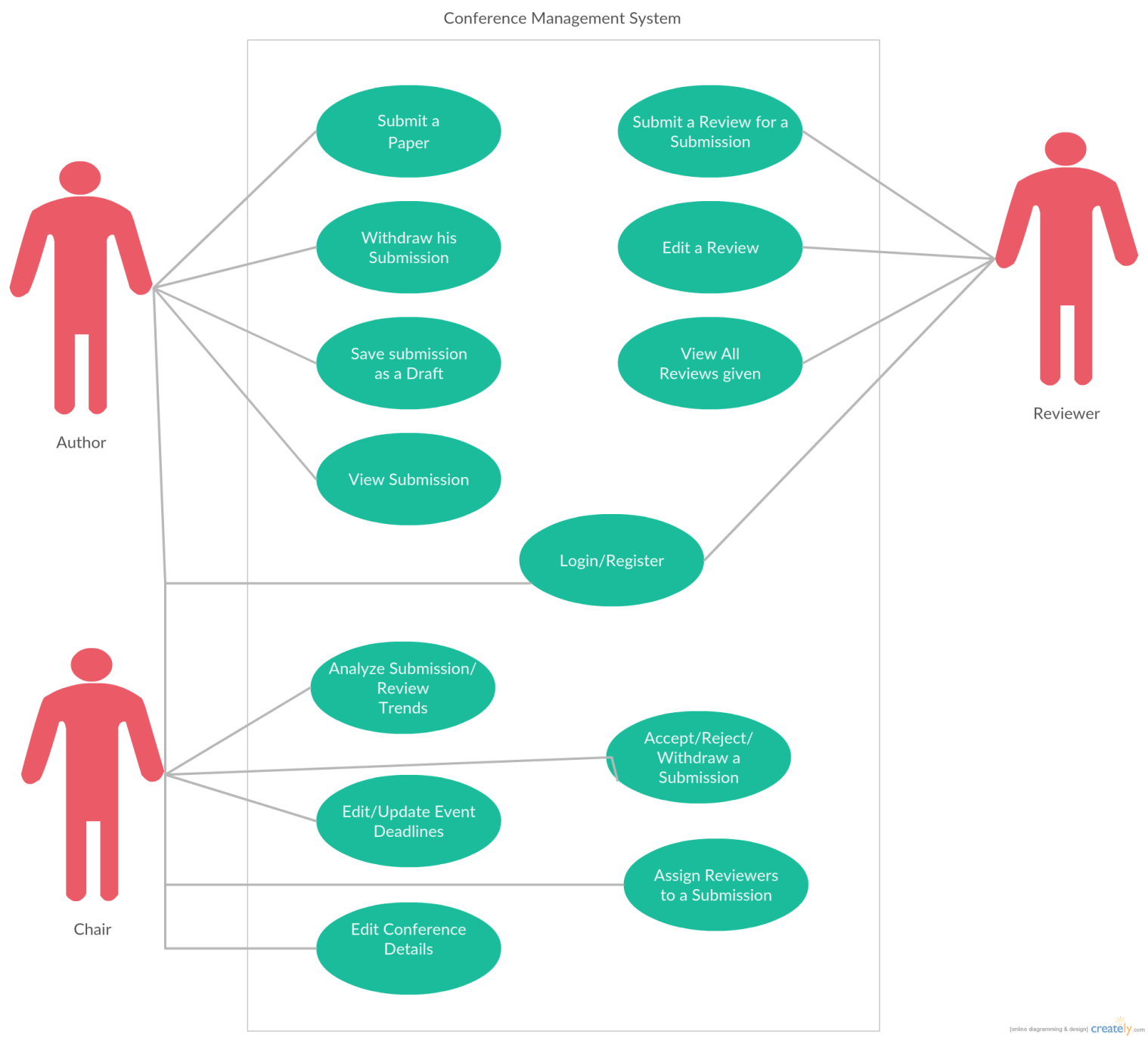


Figure 6: Use Case Diagram - CMS

# Developer Guide

This guide will help developer to run the project in your local system, and explains project structure to enable developer to add maintain and contribute to the project.

## Getting started

This process will install MongoDB, NodeJS and ExpressJS

* + - * Clone this repo:  
        git clone ssh://git@scm.informatik.tu-darmstadt.de/iptk-ss2016/iptk-ss2016-team-mike.git
      * change directory using below command   
        cd iptk-ss2016-team-mike
        + Installs the required npm packages  
          For windows users:   
          npm install --no-bin-links
        + For Linux:  
          npm install
* Installs the required bower components  
  bower install
* rename config\_template.js to config.js
* Add mongodb URL in config.js
* run intidb.js to initial database with dummy values. First, create only the db tables using createRoles() method. Then uncomment the createUser() method and run the script again.
* Start Node server:  
  npm start
* Goto to your browser and access the app using localhost:3000   
  You should be able to see **'Conference Management Systems'** on your browser

## Project Structure

All bower dependencies are mentioned in bower.json and all npm modules are mentioned in package.json.

We are using ui-router module to declare all routes easily. All such routes are defined in app.js file.

All links to be displayed in the UI are then added to views/nav.html file. Each route generally has its separate template.html and separate controller.

Any Angular directive to be used is defined in app/js/directives folder.

We make all API calls from respective app/js/factories/\*\*factory.js file. In the $http service we generate a token to be used for authentication and authorization at the back-end. This token is generated in app.js file.

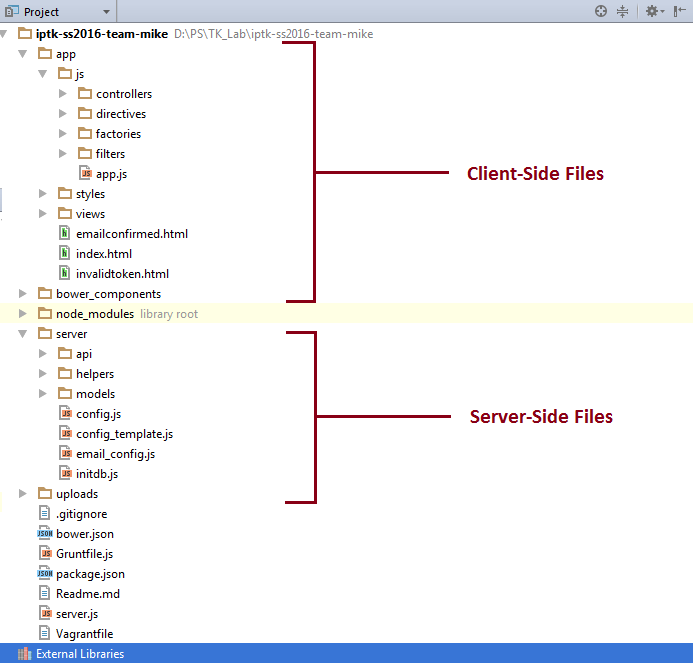


Figure 7: Project Structure

On the Server side, server.js initializes the authorization and authentication services, and the route structure is setup as the folder structure for each type of REST Api. We have used express.js for defining such routes.

All web APIs are written in server/api folder. Each api call used http token to perform authorization check and the queries the mongoDB and returns the required json to the client browser.

Example of Web APIs:



Figure 8: Web Apis

# Summary

CMS is a web based conference management system. It is developed using MEAN Stack. It successfully stores all the information related to a conference. Chair can open, close or extend a conference. An Author can submit his research work and reviewers are assigned to the paper by the chair. Anyone can sign up using just an email Id. This system is fully functional for one conference at the moment. Whenever a user signs up a verification email is sent to the respective email address. Also, for each paper submission, respective author receives notification via email about its success status of submission. Similarly, for each review submission, reviewer is notified via email about its submission as a record.

# Future Work

Currently, the application accommodates one conference at a time. This functionality can be extended, in future, to facilitate multiple conferences at the same time. Application also uses static scheduling for assigning papers to reviewers – means that papers are manually assigned to the reviewers by the chair. Dynamic assignment of papers can be a potential feature to be added in upcoming version of CMS.

# References

1. <https://scm.informatik.tu-darmstadt.de/projects/iptk-ss2016-team-mike>
2. <http://bluefletch.com/wp-content/uploads/2014/05/mean-arch.png>
3. <https://programmaticponderings.files.wordpress.com/2015/01/custom-search-architecture-angular2.png>
4. <http://trinathswebapps.blogspot.de/2016/01/history-of-angularjs.html>
5. <https://programmaticponderings.files.wordpress.com/2015/01/folder-structure-of-search-package-with-callouts.png>
6. <http://www.danielbaulig.de/socket-ioexpress/>
7. <https://docs.mongodb.com/manual/introduction/>
8. <https://docs.angularjs.org/guide/introduction>

# List of Figures

Figure 1: MEAN Stack Framework 3

Figure 2: MEAN Stack Application Architecture 4

Figure 3: MEAN Stack Component Interaction 5

Figure 4: Node JS Architecture 6

Figure 5: Database Schema 7

Figure 6: Use Case Diagram - CMS 1

Figure 7: Project Structure 3

Figure 8: Web Apis 4

# List of Abbreviation

CMS Conference Management System

MEAN Mongodb, Express, Angular Js, Node Js