Conference Management System

Team Victor:
Abhijit Singh (2765013)
Partik Kewalramani (2983918)
Kumar Saurabh (2257628)
Pallav Gupta(2567101)
Sufyaan Chand Jakate (2730857)
Manish Kumar (2690007)

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Table of Contents

1. Introduction

Conference Management System is a web based solution to support the organization of academic conferences, workshops and seminars. This provides a common portal which can be used by many users with a flexibility to manage events, submissions, reviews and profile information. It provides features in order to allow authors for registering into the system and submit their papers with the corresponding event. Similarly reviewers are allowed to review papers which are assigned to them by chair.
Access to all features of the system is controlled by a set of permissions which are allocated on the basis of a user's roles at the conference. For example, only chair is responsible for the administrative tasks such as assigning papers to reviewers, schedule management and inspect the abstract data.
Through this design document we will provide the narrative and graphical documentation of Conference Management System. It includes technical description and demonstrates the user documentation that briefly describes how to use the application.

1.1. Purpose

Every academic or professional organization requires some extent of communication in order to exchange, review and manage data. In this scenario conference management system act as a useful tool which will allow users to exchange, review and manage data in a well organized and controlled way.
Purpose of this project is to create similar web based solution in order to provide various functionalities. Such that many users (Authors, Reviewers and Chairs) make use of a common portal in order to perform tasks that are relevant to their roles.

1.2. Scope

This documentation gives detailed information about the architecture of the application and used technologies. It provides detailed flow of project from the technical details of each and every part of interaction from login to scheduling of events.
This document will provide insightful details that will be helpful for any technical person to overtake the project and further add on new functionality. The knowledge transfer details are being mentioned to make the life of developer easy of understanding flow at each step and the technology used.

1.3. Definitions, Acronyms, and Abbreviations

- COMS: Conference Management System
- Author: The user who is responsible for the submission of the papers.
- Reviewer: The user who is responsible for reviewing the assigned papers.
- Chair: The user who is responsible for the administrative tasks and data.
- 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.
- JavaScript: It is a high-level, dynamic, untyped, and interpreted programming language.
- Hypertext Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages.
- AngularJS (commonly referred to as "Angular" or "Angular.js") is an open-source web application framework mainly maintained by Google and by a community of individuals most notably, Rangle.io as well as group developers and corporations to address many of the challenges encountered in developing single-page applications.
- Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.
- Express Js: Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications. It is the de facto standard server framework for node.js.
- REST: It is the software architectural style of the World Wide Web. REST's coordinated set of constraints, applied to the design of components in a distributed hypermedia system, can lead to a higher-performing and more maintainable software architecture.
- MongoDB: MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

2. Design Overview

2.1. Introduction

COMS provide functionality to manage submissions, reviews and administrative tasks. The system revolves around three types of users: Author, Reviewer and Chair where each user is capable of performing limited tasks which are relevant to their roles.

Below are the tasks lists for each role.

- Role: User
  - Register to the system (with email, password)
  - Login(with registered email, password) and logout
  - User's profile management(edit, remove)
  - Multiple roles(author, review, chair)
- Role: Author
  - Create new submission (if submission is open)
  - Access current submissions (look, edit, withdraw submissions)
- Role: Reviewer
  - Access assigned submissions (look at an assigned submission)
  - Make a review (if review is open)
  - Edit a review (if review is open)
- Role: Chair
  - List all submissions (look, withdraw submissions)
  - List all authors
  - List all reviewers
  - List all reviews
  - Assign paper to reviewer
  - Schedule management(set close deadlines for submissions and reviews)
  - View summary data and reports

2.2. Components

Components involved are:-

- Angularjs
- Node JS
- MongoDB
- Mongoose

2.3. System Architecture

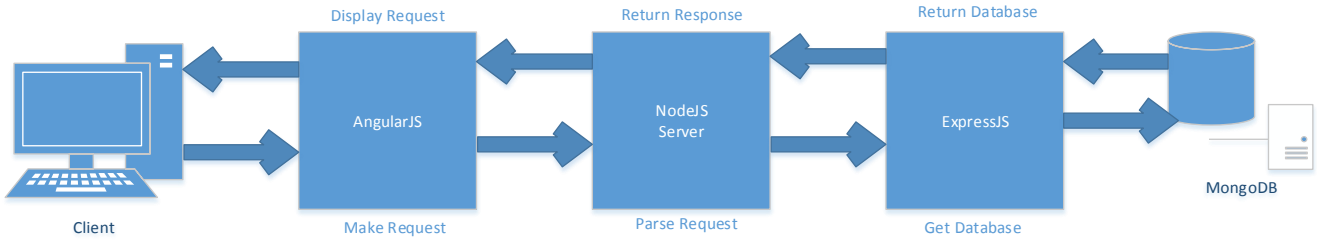

Figure 1: System Architecture

AngularJS is the initial technology which uses in front end when client first interacts with website as login. After which angular JS in turn makes interaction with NodeJs server to run Mongoose queries and routes so that different CSS pages are loaded.
NodeJS interacts with ExpressJS for interaction with MongoDB. ExpressJS uses Mongoose running on NodeJs and establishes connection with MongoDB. The stable database MongoDB is used for saving all the relevant data for the project.

2.4. System Interface Design

Figure (2) is the context diagram of the system that defines the boundary between the system and its environment, showing the entities that interact with it. This represents the high level view of the system. We have two types of users (Chair and normal user) which interact with the system where each of them can perform various actions that are shown below:

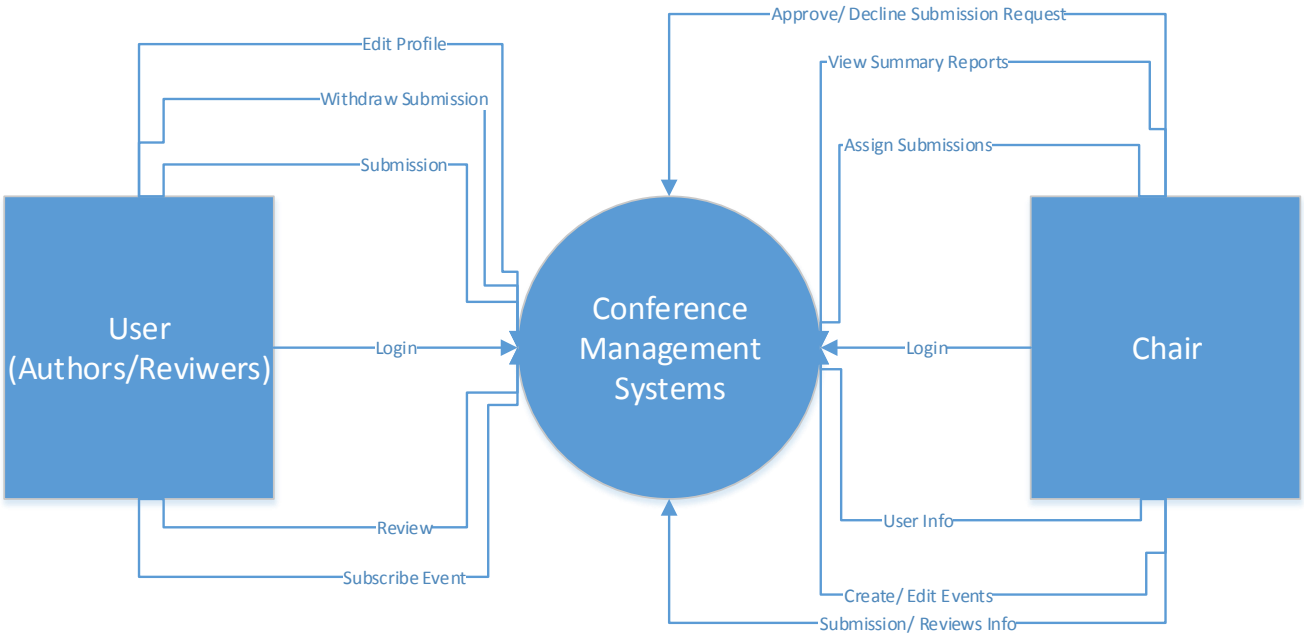

Figure 2: Context Diagram of Conference Management System

Figure (3) represents DFD Level 1 that provides a more detailed breakout of pieces of the Context Level Diagram. This represents the main functions that are carried out by the system, as the high-level process of the Context Diagram breakdown into its sub processes.

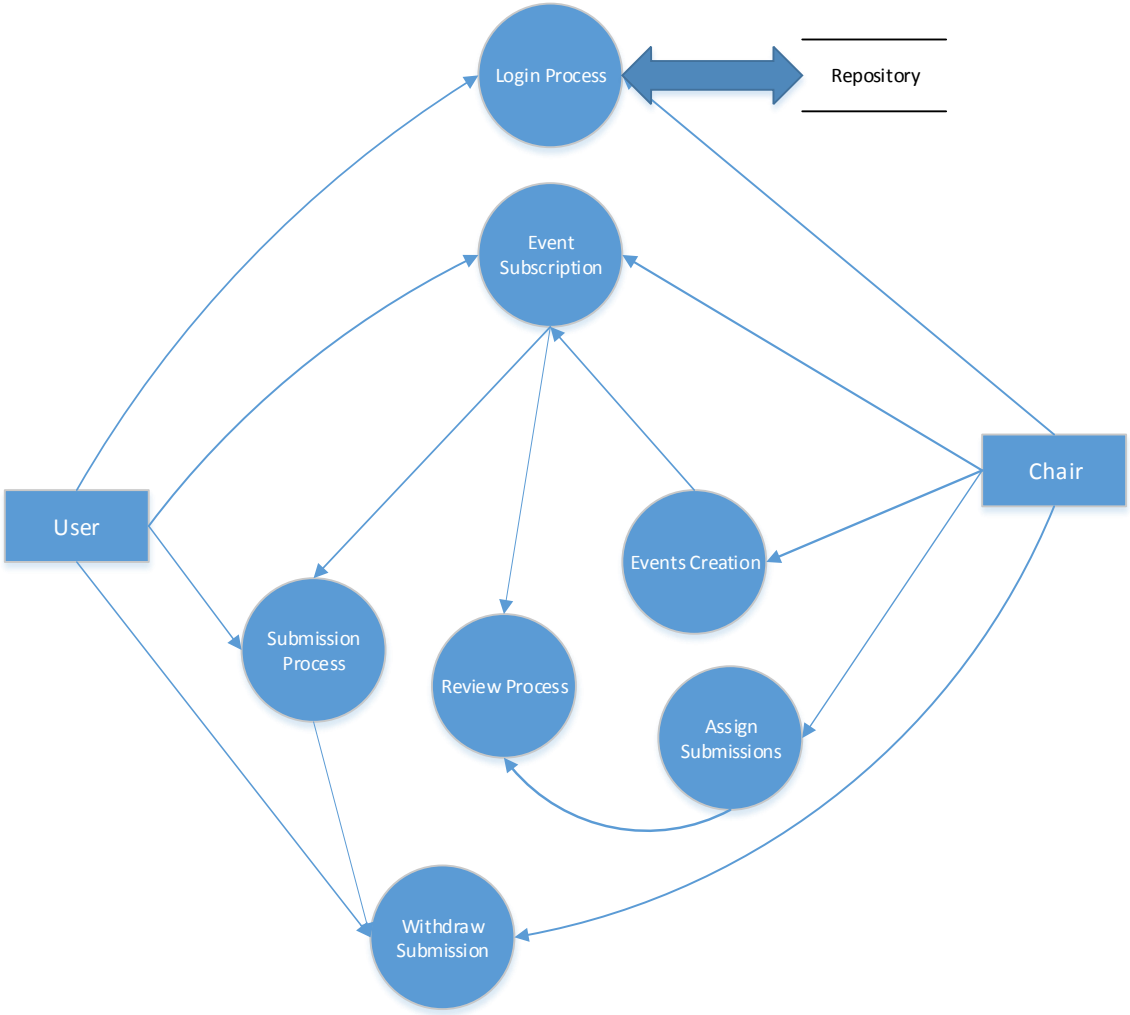

Figure 3: Data Flow Diagram (DFD1)

As represented, chair and normal user can login into the system. Chair can create new events whereas user can send request for subscription of a particular event. Chair can assign submissions to particular user for review. User can perform submission and review of assigned submissions. Both user and chair are able to withdraw submissions.

2.5. User Interface Design

Figure (4) represents user flow for login page. Each user can login into the system with unique username and password and if user is new then new registration can be done.

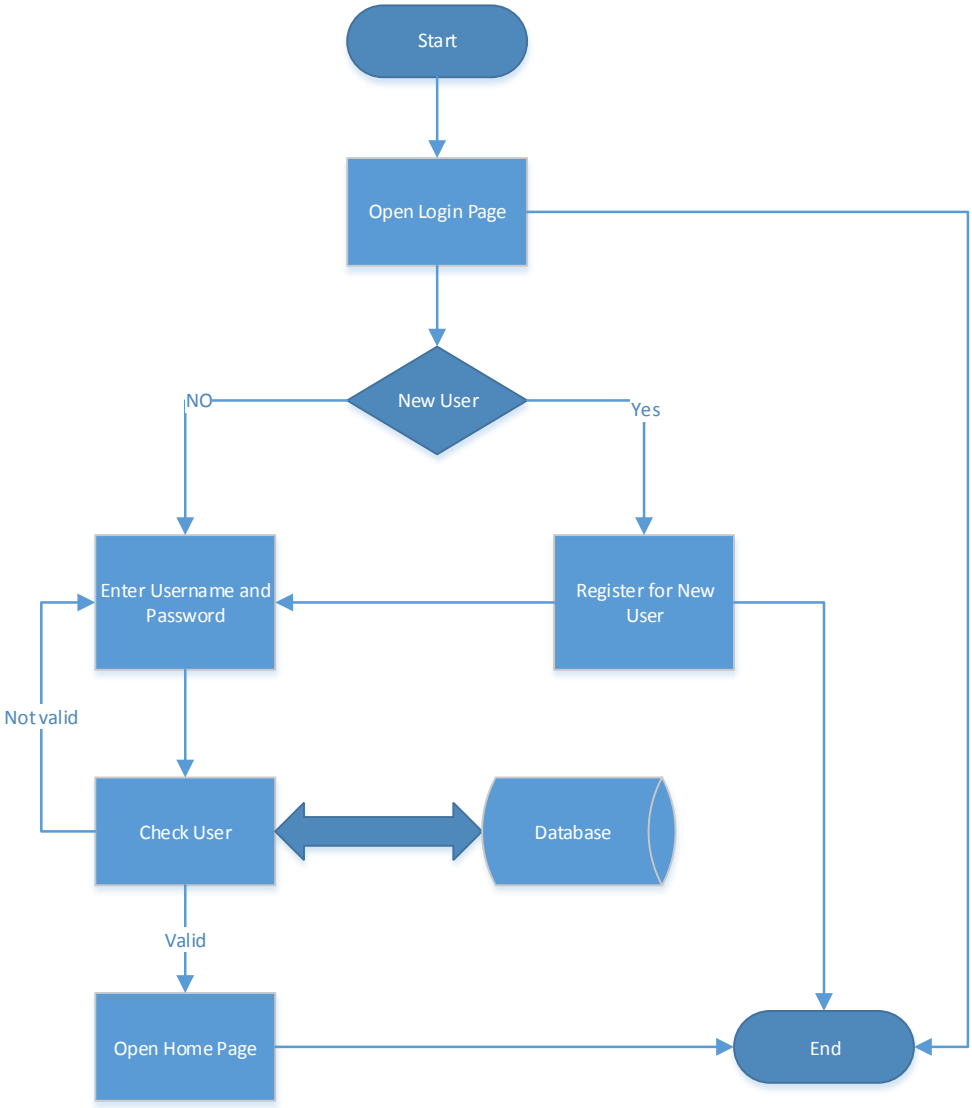

Figure 4: Login Page Flow

Figure (5) represent flow of the system as perceived by normal user where it can subscribe to a new event or can submit to an approved event. Also user can view assigned submissions and perform reviews of those submissions.

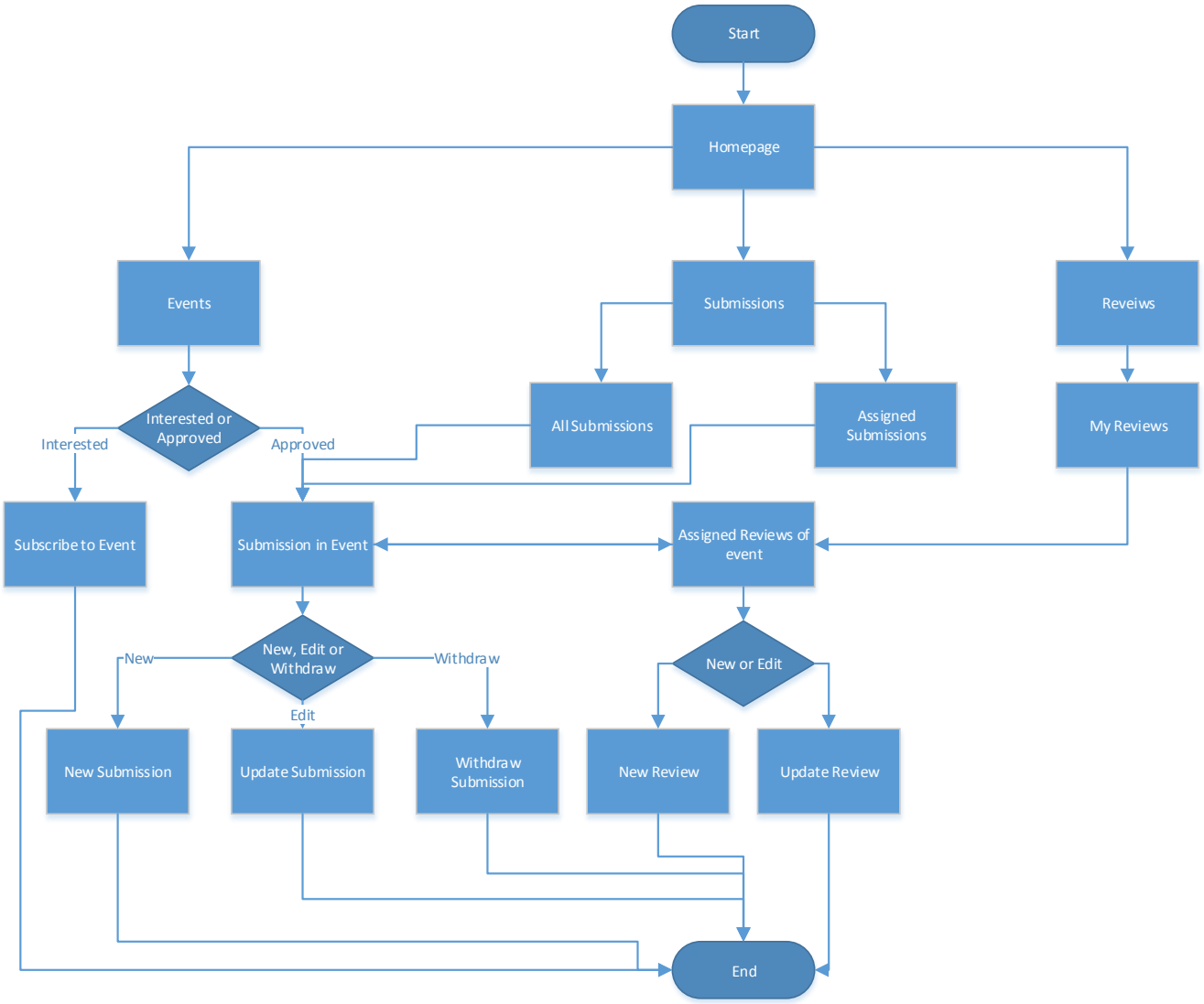

Figure 5: User (Author, Reviewer) Interface Flow

Figure (6) represent flow of the system as perceived by chair where it can create new event or edit existing events. Chair can view all submissions and reviews. Also can view submission details and assign these submissions for review. Apart from this chair can perform user request management where it can unsubscribe users or approve user's request for submissions.

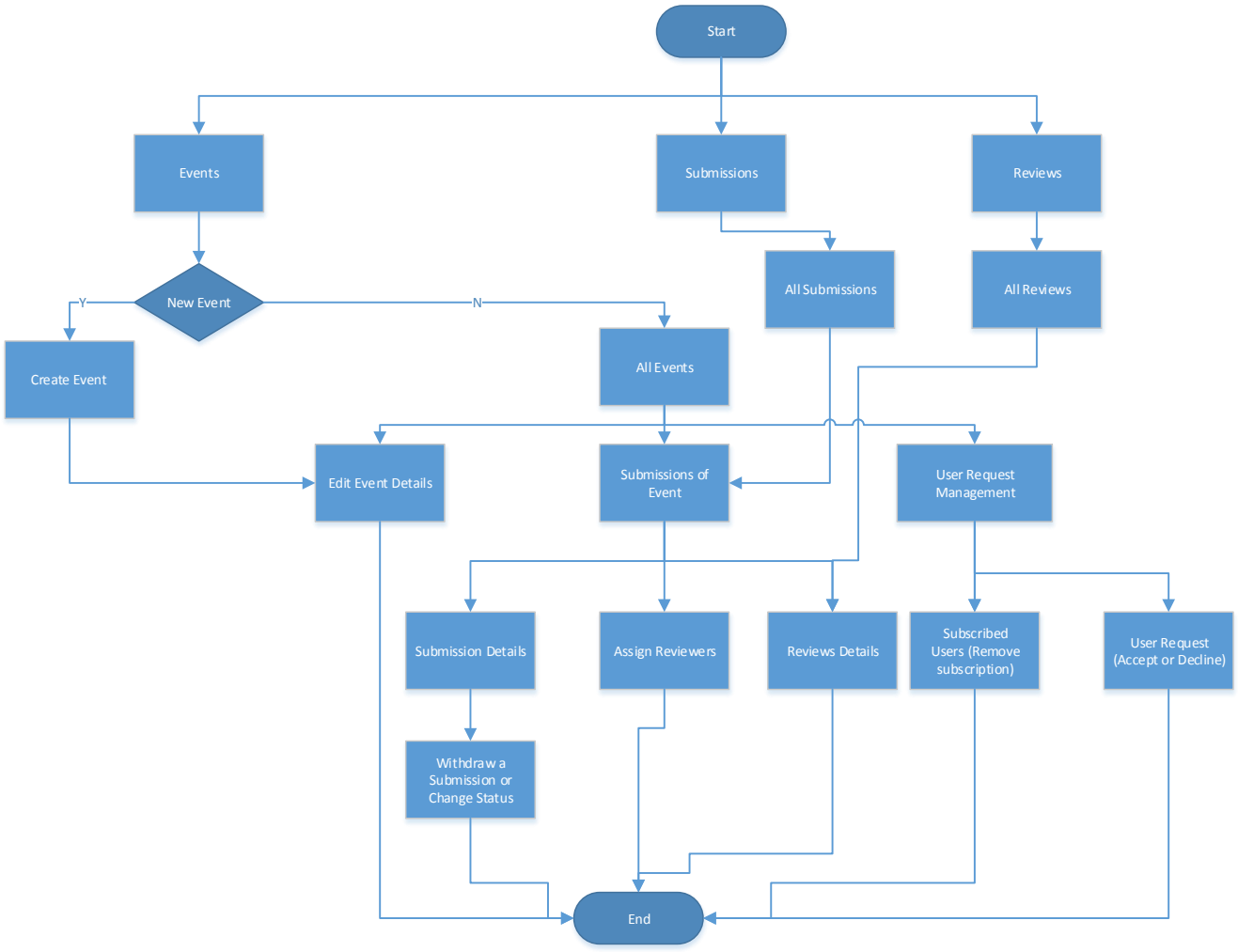

Figure 6: Chair Interface Flow

2.6. Detailed Design

In COMS different users will be able to login. The roles assigned are normal user (Authors and Reviewer) and Chair. Depending on the Role and the authentication, the home page will be showed up as represented in figures 5 and 6.
Any user irrespective of roles has to register/login and the login authentication will be authenticated by checking from database (MongoDB) and fulfilling the constraints. At the same time role mentioned of the user will be fetched from the DB and the relevant home page will be shown to user.

For each module like login, submissions, event creation and modification, reviews, subscription we created separate html pages. Root directory of the project looks like this:

- server.js - This is starting point of the app. All the modules, routes, database connections are imported here.
- app/ - Server routes, controllers, models available here
- config/ - config/ Application configuration: Mongoose, express, passport and environment
- public/ - All files publically available to server are placed here like css, js, templates
- node_modules/ - All the external modules installed using npm install are here.
- package.json - This file defines a JSON object that contains various properties of our project including things such as name and version number. It can also define what versions of Node are required and what modules our project depends on. A list of possible options can be found in npm's documentation.

Front End:

In the project we have used different angular directives we will see some of their usage.

- ng-controller: In project we have created controller for each task

  Example: for adding new project we have below expression in login.controller.js. We add ng-controller="LoginController" this in login html.

```
angular.module('loginModule',[]).controller('LoginController',
function ($scope, $location, AuthService,$rootScope, $state, $mdToast) {}
```

- ng-model: binds the value of html control like input,textare to application data.

  Example: `<input ng-model="user.password" type="password" placeholder="Password (required)" ng-required="true">`

- ng-disable: sets the disabled attribute on the element if the expression inside it evaluates to be true.
- ng-click: redirects to the code which will be executed when that it is clicked login.

  Example: `<md-button ng-click="register(user)" class="md-raised md-primary" ng-disabled="registrationForm.$invalid" data-ngcloak="switchBool('showSuccess')">Register</md-button>`

Backend Part:

Creating schemas with mongoose: We added following code in mongoose.js to connect to local instance of MongoDB.

```
mongoose = require('mongoose');
var db = mongoose.connect(config.db);
```

In our /models directory we created model.js. All the required schemas are written here.

Routes: There are some routes created which frontend client can interact. We have routes/index.server.routes.js , review.document.server.routes.js, user.server,routes.js etc.

3. Conclusion

The newly developed Conference Management System has achieved all of the goals mentioned in the project features list. It enables users to make use of system in order to submit/review the documents and to manage profile info, submissions and reviews. Also user can subscribe to any available event.

It also enables chair to create and modify events. After approval of chair, user will be subscribed to the event and will be able to submit or review the documents. From the system, chair can manage submissions, reviews, events and user info.

We suggest some changes for the future improvements:

- Security features to be added (using certificates)
- Social media integration like login via facebook or google

4. Appendix

List of Figures

Biblography

1. https://www.mongodb.org/
2. https://angularjs.org/
3. http://www.w3schools.com/js/
4. https://nodejs.org/en/
5. http://mongoosejs.com/
6. http://expressjs.com/
7. https://en.wikipedia.org/wiki/JavaScript
8. https://mean.io/

Index Directory