
Internet Praktikum TK 2016

Conference Management System

Team Whisky

Auel, Tarek,
Sahin, Huzeyfe
Schanz, Markus



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Contents

I. User Documentation	3
1. Introduction	3
2. User	3
2.1. Create an Account	4
2.2. Application Login	4
2.3. Application Logout	4
2.4. Manage Profile	4
2.5. List Attended Conferences	5
2.6. Create a Conference	5
2.7. Select a Conference	5
3. Attendee	5
3.1. Join a Conference	5
4. Author	6
4.1. Create a Submission	6
4.2. Edit and Delete a Submission	6
5. Reviewer	6
5.1. Create a Review	6
6. Chair	6
6.1. Assign User Roles to Attendees	7
6.2. Assign Reviewer to Submissions	7
6.3. Statistics	7
II. Technical Documentation	8
1. Introduction	8
2. Architecture	8
2.1. Overview	8
2.2. Third Party Software	9
3. Implementation	9
3.1. Project Directory Structure	9
3.2. Entry Scripts	10
3.2.1. Server Application	10
3.2.2. Web Application	10
3.3. StrongLoop	11
3.3.1. Creating and Updating Models	11
3.3.2. File Upload	11
3.3.3. Generate AngularJS Services	11
3.4. The AngularJS Client Application	12
3.4.1. The app Module	12
3.4.2. Controllers	12

3.5. Role Based Access Control (RBAC)	13
3.5.1. Roles	13
3.5.2. The angular-permission library	13
3.5.3. StrongLoop Access Control List	13
4. Hands On: Extending the Application	13
4.1. Creating a View and Controller	14
4.2. Adding a new Route Definition	14
4.3. Add a Link to the Navigation Bar	15
4.4. Test the new extension	15
III. Appendix	16
A. Application Installation	16
A.1. Prerequisites	16
A.2. Installation	16
B. Database Entity Relationship Model	17
C. Application Screenshots	18
D. (Un)Implemented Features	28

Part I.

User Documentation

1 Introduction

TKonference is a web-based conference management system that offers all required features for managing attendees, authors, reviewers, and chair(s) of a conference. The application is open-source and highly extendable. If you are a server administrator who wants to install this software as a self-hosted solution, please refer to appendix A for the required prerequisites and installation instructions. If you are a developer who wants to extend the application, please refer to part II of this documentation which provides an overview of the software's architecture and implementation.

This part of the documentation is meant as a guide for users of the application. It describes the usage of the application and its provided features. The remainder of the guide is divided into five sections, each dedicated to one of the following user roles, respectively:

- **User:** Basically, a user is someone who is visiting the web application with a web browser.¹ This is the most restricted account type which is only allowed to perform basic actions (see chapter 2).
- **Attendee:** An attendee is a user who is actively participating at a specific conference. A user can become an attendee of a conference by joining it (see chapter 3).
- **Author:** An author is an attendee of a conference who has the permission to submit a conference-relevant paper (PDF), thus applying as a speaker (see chapter 4).
- **Reviewer:** Reviewers are persons with great expertise in the research area that is targeted by the conference. They are responsible to review submitted papers of authors, i.e., provide them with feedback, and rate the quality and relevance of the submitted paper (see chapter 5).
- **Chair:** The chair is responsible for managing the conference. He has the power to name authors, reviewers, and further chairs from the list of attendees. When the review phase is over, the chair decides, based on the given reviews, which papers to approve and therefore which authors to invite as a speaker (see chapter 6).

2 User

The basic abilities of a user comprise of 2.1) registering of a new account, 2.2) log in to an account, 2.4) manage the profile of an account, and 2.6) create own conferences.

For the remainder of this chapter, except for 2.1 and 2.2, it is assumed that the user is logged in to the system and has already opened the start page of **TKonference** in a web browser (e.g. `http://localhost:3000`, when setup locally).

¹ Strictly speaking, a user is someone who authenticated himself against the application with a valid username and password. This distinction is only relevant if you are a developer (see part II, section 3.5.1).

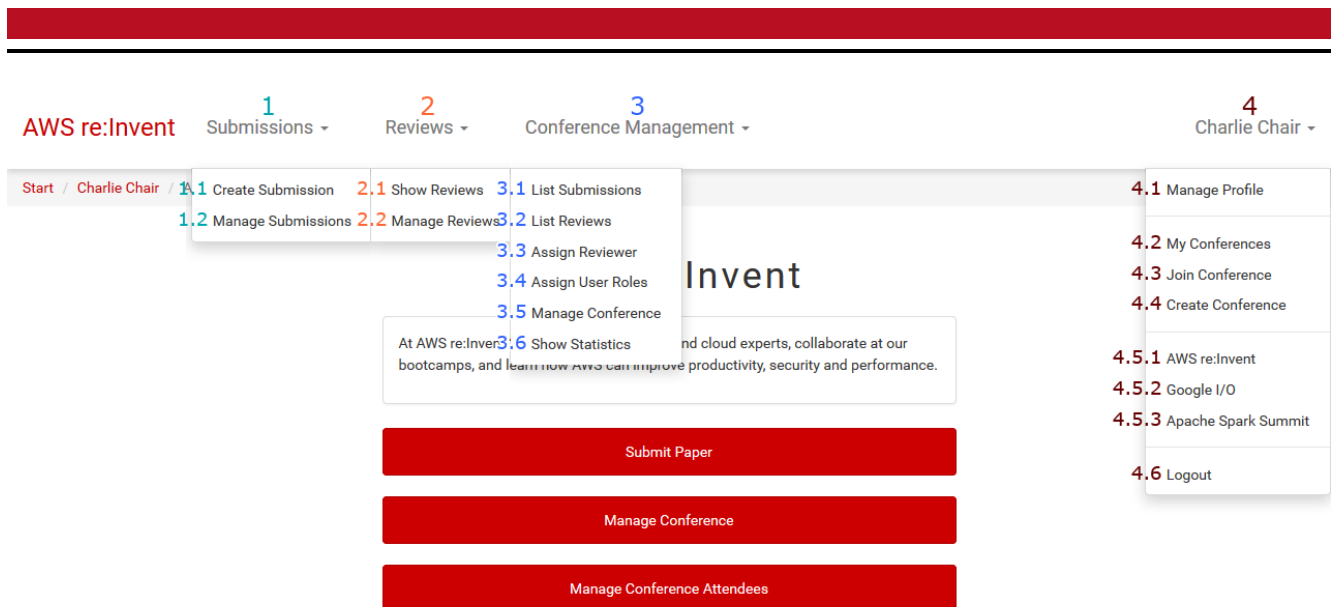


Figure 1.1.: Application UI - Logged in, menus opened

2.1 Create an Account

1. Click on the **Create new account** link (Figure C.1).
2. Fill out the name-, email-, username-, affiliation-, and password-fields, respectively (Figure C.2)
3. Click the **Submit** button to complete the registration process.

On success, the user is redirected to the login page.

2.2 Application Login

1. Fill out the username-, and password-field of the login form (Figure C.1).
2. Press the **Login** button to authenticate yourself against the system.

On success, the user is redirected a) to his account overview (Figure C.3), or b) to the conference overview of his default conference, in case he has set one (Figure C.4).

2.3 Application Logout

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, click on the **Logout** link (Figure 1.1, #4.6).

The log out operation is important whenever multiple persons have access to the computer.

2.4 Manage Profile

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, click on the **Manage Profile** link (Figure 1.1, #4.1).
3. Choose amongst the items in the top menu according to the options below (Figure C.5).
 - a) Edit your profile (default):
Adapt your profile as desired and confirm the changes by pressing the **Edit Profile** button.
 - b) Change your password:
Enter your new password in both fields to avoid misspelling and confirm it by pressing the **Change** button.

-
- c) Delete your account:

Delete your account by pressing the **Delete** button. Read the displayed instructions carefully and confirm your decision by entering your username in the form field and pressing the **I understand the consequences, delete this profile** button.

2.5 List Attended Conferences

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, click on the **My Conferences** link (Figure 1.1, #4.2).

This will list all conferences that the user attends, as well as some related quick links (Figure C.6). In case that the user does not attend any conference, he is provided with two quick links to create a conference (2.6), or join a conference (3.1), accordingly.

2.6 Create a Conference

TKonference is able to manage multiple conferences. All users are allowed to create their own conference which automatically makes them the chair of it.

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, click on the **Create Conference** link (Figure 1.1, #4.4).
3. Fill out the form with a conference name, description, important deadlines, and a shared secret (Figure C.7). This secret must be known by other users if they want to join your conference as an attendee.

2.7 Select a Conference

Because a user is able to attend multiple conferences, he has to select an 'active' conference. The conference name, and links displayed in the header navigation bar are valid for the active conference only.

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, select one of the conferences that you attend (Figure 1.1, #4.5.1, #4.5.2, #4.5.3).

You are now redirected to the conference overview (Figure C.4). Also, the navigation bar is updated and shows the conference name and deadlines.

3 Attendee

An attendee is a user who is a member of a conference. In order to become an attendee, a user has to join a conference.

3.1 Join a Conference

1. Navigate to the drop-down menu located in the top right corner and open it by clicking on **<your name>** (Figure 1.1, #4)
2. Inside the newly opened menu, click on the **Join Conference** link (Figure 1.1, #4.3).
3. Select the conference that you want to join from the drop down list and fill out the shared secret that you received from the chair or other attendees of the conference.
4. Confirm by pressing the **Join Conference** button.

On success you are redirected to the conference overview page (Figure C.4) which now also displays the joined conference.

4 Author

Those who attend a conference can be assigned the author role by the chair of the conference. An author is allowed to submit papers for the conference during the submission phase.

4.1 Create a Submission

1. Navigate to the submission creation page by clicking **Submissions** > **Create Submission** (Figure 1.1, #1.1).
2. You are now presented with the submission creation page (Figure C.8). The provided form allows you to select multiple co-authors from the list of conference attendees. The abstract field does support markdown syntax. You may provide multiple keywords for your paper. Please note that the uploaded paper must be provided as PDF.
3. Confirm your submission by pressing the **Submit Paper** button.

The newly created submission is automatically created as draft. You may edit the submission until it is marked as final (see next)

4.2 Edit and Delete a Submission

1. Navigate to the submission overview (Figure C.9) by clicking **Submissions** > **Manage Submissions** (Figure 1.1, #1.2).
2. Press the **Edit (Withdraw)** button to edit (delete) the corresponding submission.

To finalize your submission, press on the **Final** switch within the submission overview. Once a submission is final, it can not be edited anymore. This allows reviewers to start the review process early on.

5 Reviewer

Reviewers are allowed to create reviews for their assigned submissions during the review phase. A review consists of the expertise of the reviewer (1-5), the overall rating of the submitted paper (1-5), and a textual feedback for each of the following:

- Evaluation Summary
- Detailed Comments
- Major Strong Points
- Major Weak Points

5.1 Create a Review

1. Navigate to the review overview (Figure C.10) by clicking **Reviews** > **Show Reviews** (Figure 1.1, #2.1).
2. Press the **Review** button to create/edit the corresponding review.
3. Fill out the form accordingly (Figure C.11).
Note that all textual inputs on this page support markdown syntax.
4. Save your review by pressing the **Submit Review** button.

6 Chair

The chair of the conference is responsible to manage the conference by providing the submission- and review deadlines, assign author and reviewer roles to attendees and assign submissions to reviewers. He is also able to view the conference statistic page that visualizes and aggregates useful information.

The process of listing submissions and reviews (Figure 1.1, #3.1 and #3.2) does not require any further explanation and therefore, no dedicated section exists to explain these features. The process of managing a conference (Figure 1.1, #3.5) is equivalent to that of creating a conference which was explained previously in section 2.6.

6.1 Assign User Roles to Attendees

By default, members who join a conference have no privileges. The chair can empower attendees of the conference to gain the privileges of an author, reviewer, or chair which allows them to execute the actions described in chapter 4, 5, and 6, respectively.

1. Navigate to the user management (Figure C.12) by clicking **Conference Management > Assign User Roles** (Figure 1.1, #3.4).
2. In the list of conference attendees, each user is displayed in a separate panel. Each panel contains three switches for the author, reviewer, and chair role, respectively. Click on any of the switches to toggle the user role. No further confirmation is necessary.

6.2 Assign Reviewer to Submissions

Once an author create a submission, the chair is able to assign any number of reviewers to it. These reviewers will then have the ability to write a review for this submission (individually).

1. Navigate to the review assignment page (Figure C.13) by clicking **Conference Management > Assign Reviewer** (Figure 1.1, #3.3).
2. Each available submission is now displayed in its own panel. You may assign reviewers by one of the following methods:
 - a) **Manual assignment:** Within the individual panels, select the reviewers manually from the pre-populated drop down lists. They hold a list of all known reviewers of the conference.
 - b) **Automatic assignment:** For an effortless mass-assignment of reviewers, use the **Auto Assignment** button on top of the page. It will randomly assign an equal number of reviewers to each submission (if possible). However, it wont assign reviewers to submissions of which they are a co-author. When using this function, be aware of the fact that manual assignments will be overridden.
3. Click on the **Save Changes** button to confirm the assignments.

6.3 Statistics

The statistics (Figure C.14) are reached by navigating to the corresponding page via **Conference Management > Show Statistics** (Figure 1.1, #3.6). This page aggregates available information about the conference in five diagrams:

- **User Role Distribution:**
Shows the number of attendees, authors, reviewers, and chairs of a conference as a bar diagram.
- **Author/Submission Relationship:**
A bar diagram that displays the number of total submissions and the number of total authors of these submissions.
- **Review Status:**
Shows a doughnut diagram that visualizes the relationship between reviews in draft- and finish state.
- **Submissions by Affiliation:**
Shows a bar diagram with the top 3 affiliations, i.e., the ones with the highest number of (co-)authors.
- **Submission Status:**
Shows a doughnut diagram that visualizes the relationship between submissions in draft-, finish-, and approved status.

Part II.

Technical Documentation

1 Introduction

This part of the documentation is intended for developers who want to setup the project on their local machine and contribute changes to the source code. The documentation is written such that the individual chapters can be read in arbitrary order. Each chapter is a self-contained piece of documentation. However, the software tools listed in section 2.2 are heavily used throughout the application. Therefore, the reader is advised to take a look at them before proceeding with the rest of the documentation. For further information on these tools, please refer to the individual project sites. Appendix A hold the instructions on how to setup the application on a local machine.

The remainder of this documentation is structured as follows: Chapter 2 gives an insight of how the application works conceptually and provides an overview of the used software libraries. Implementation details such as authentication, authorization, etc. are covered in chapter 3. This chapter also covers the directory structure of the project as well as important files that are involved in bootstrapping the application. Eventually, chapter 4 describes the process of extending the application, i.e., how to extend the navigation bar, add a new page, etc.

2 Architecture

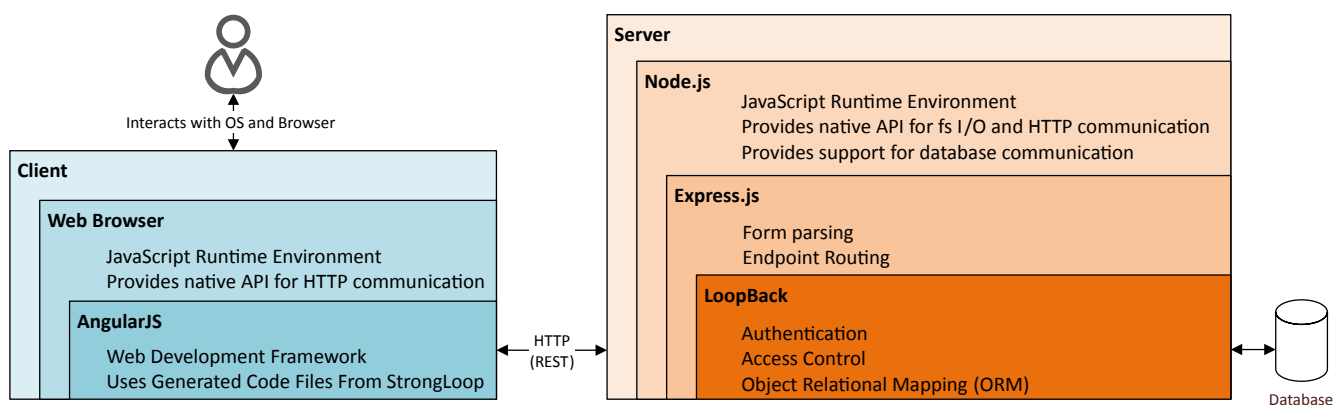


Figure 2.1.: Architecture.

2.1 Overview

On a high-level view, the application follows a client-server approach. The server part of the application primarily manages the database and provides a public REST API¹ whereas the client part runs in the browser of the user and provides an user interface (UI). The client part is responsible to translate user interactions into requests against the server API and reflect the responses in the UI.

¹ https://en.wikipedia.org/wiki/Representational_state_transfer

2.2 Third Party Software

As seen in figure 2.1, the application utilizes third party software & libraries in order to keep the codebase clean and allow for a productive development. This section gives an overview of the most important third party libraries & tools that the project relies on.

Node.js & Node Package Manager

Node.js is a standalone JavaScript runtime environment, i.e., it requires no web browser to execute JavaScript code.² In recent years, Node.js became very popular as it allows developers to use JavaScript for both server and client programming. It comes bundled together with its own package manager npm which can be used to install a wide variety of third party libraries that further ease the development process.

StrongLoop

StrongLoop is a mandatory library on which the application is primarily build upon.³ This tool is able to generate code files for both server and client that can be used to manipulate a previously defined database schema. It comes bundled with a framework called LoopBack and a library called Express.js that is used internally to parse and route HTTP requests. It can be installed (and kept up-to-date) via npm.

Bower

Analogous to npm, Bower is a package manager for JavaScript libraries.⁴ However, npm manages libraries that are required for the server part of the application (Node.js) whereas Bower focuses on the management of libraries that are used on the client side, i.e., in the browser.

AngularJS

AngularJS is a web application framework that eases web development by providing a model-view-controller (MVC) like architecture.⁵ The code that is generated by StrongLoop is also targeting AngularJS.

3 Implementation

3.1 Project Directory Structure

The top level directory of the project is organized as follows:

- /bin
- /common
- /node_modules
- /storage
- /client
- /doc
- /server

/

Besides the listed directories above, the root directory / does contain several files worth mentioning. bower.json and .bowerrc are the configuration files that are automatically found and loaded by Bower. They contain the list of dependencies and corresponding version constraints to be installed by Bower. By renaming the db.json.default file to db.json, the application can be tested with some predefined seed data. The package.json file is to npm what bower.json is to Bower. The other files in the directory are either self-explaining or irrelevant for this part of the documentation.

/bin

Contains executable scripts that are used to initialize the application database with dummy data.

/client

Resembles the root directory that is publicly accessible through the HTTP web server, i.e., it contains all the code that is relevant for the client application.

² <https://nodejs.org/>

³ <https://strongloop.com/>

⁴ <https://bower.io/>

⁵ <https://angularjs.org/>

/common

Holds the models that describe the application database schema. These models are taken into account by StrongLoop when generating the client- and server code files. The files inside this directory can either be edited manually (according to a specified standard) or through the StrongLoop console line interface (see 3.3.1).

/doc

Contains all documentation relevant files such as the documentation at hand.

/node_modules

This is where npm stores downloaded modules. Besides a .gitignore file, this directory is empty when the repository is checked out cleanly.

/server

Contains the code that is executed inside the Node.js environment on the server side. It provides the logic to manipulate data, handles authentication and access control and also provides the public REST API.

/storage

LoopBack¹ manages file uploads in so-called containers. This directory contains one directory for each container and each of these hold in turn a list of files that were uploaded to the corresponding container. The directory is initially empty when the repository is checked out cleanly. It gets filled once the conference users start to submit their papers.

3.2 Entry Scripts

Entry scripts are the first step in the bootstrapping process of the application. These are somewhat comparable to the `main()` method of a C++ program. Since the application runs in a distributed fashion, i.e., on a client and a server, it has entry scripts for both of them.

3.2.1 Server Application

As described in section 3.1, the server part of the application resides inside `/server`. When Node.js is started inside the project root directory via `node .`, it executes the script `server/server.js`, as defined in `package.json`. Like most of the files inside `server/`, this file is a slightly modified version of the LoopBack example application². It setups some preliminary routes for Express.js as well as configuring some middleware components, e.g., to parse form data. Eventually, the start-up routine is invoked and forwards all requests to LoopBack which then handles the request accordingly.

The purpose of StrongLoop is to provide a REST API to do basic CRUD³ operations on a predefined database schema. Since all the magic is done by StrongLoop, a developer usually does not need to dive further into the topic of request handling etc. Everything that a developer needs to do is defining and/or extending the models inside `/common/models` and StrongLoop will take care of the rest. The process of model creation and modification is described later in section 3.3.1.

3.2.2 Web Application

As previously mentioned (3.1), the `/client` directory is exposed through the HTTP web server. In particular, the client is eventually sending an HTTP request to the root of this directory. Because no particular file is requested by the client, the server will respond with the contents of the `index.html` file.

The `index.html` file consists of a simple HTML5 skeleton which basically includes lots of CSS and JavaScript files. One of the libraries that gets included by the file is AngularJS. Once the browser is done with parsing the page and loading all embedded JavaScript files, the AngularJS framework is scanning for an `ng-app` attribute inside the DOM. When found, AngularJS automatically starts its bootstrap routine and eventually executes the application, which is happen to be defined inside `js/app.js`. Section 3.4 covers the contents of this file.

¹ The framework provided by StrongLoop

² <https://github.com/strongloop/loopback-example-passport>

³ Create/Read/Update/Delete operations on a persistent storage

3.3 StrongLoop

As explained in section 2.2, StrongLoop is the tool that the application uses to execute CRUD operations on the database. Besides utilizing StrongLoop as an ORM, the application makes also use of its other features such as authentication- & session management, access control and file upload support. The application is mainly managed by the command line tool, **slc**, that comes together with StrongLoop. It allows to generate and extend models, define access control rules, and more things. For an exhaustive description of the tool, please refer to the official documentation.⁴

3.3.1 Creating and Updating Models

StrongLoop does represent database tables in terms of models. A model does represent a collection of data with a given schema, e.g., a collection of users that have properties like a name, an email, a password, etc. Besides the sheer representation of data, the developer can define access rules in the model that define which data can be accessed by whom. In particular, this feature allows to restrict data-access to the owner of the data (see 3.5.3). That said, a model does also provide the methods required to retrieve, store, update and delete items of a collection.

To update or create a model, one has to decide between two basic approaches, i.e., a guided and a manual one. The advantages and disadvantages of both approaches are described next.

Guided Approach

In the guided approach, the user can use the command line tool of StrongLoop to extend or create models. In particular, the tool is capable of creating models (**slc loopback:model**), add properties to an existing one (**slc loopback:property**), associate ACL⁵ rules (**slc loopback:acl**) and add relations⁶ (**slc loopback:relation**). The main advantage of this approach is that it is easy to use and does not require the knowledge of the internal configuration format. In most parts it can be used intuitively without reading the documentation.

Manual Approach

In the manual approach, the user has to edit the model directly inside the editor. This approach is useful for advanced ACL-, property- or relation-declarations. The command line tool does only generate basic code which is not sufficient for some use cases. In these cases, the user can use a hybrid approach in which he generates the basic code with the help of the command line tool and edit it manually afterwards. Please refer to the official StrongLoop documentation for further information regarding the model format.⁷

3.3.2 File Upload

The LoopBack framework that comes with StrongLoop does provide the ability to persist uploaded files from users. It does so by managing a set of so-called containers. Basically, a container corresponds to a directory on a file system. When uploading a file, the user (or the application) has to define the container in which he wishes to upload it. The REST API generated by StrongLoop does provide special endpoints in order to download/upload files from/to a container.

Containers must be defined inside the `server/datasources.js` file. The application at hand does define a single container that is configured to store uploaded files on the local file system inside the `storage/` directory. It does also rename the files according to a unique naming schema which ensures that files, once uploaded, cannot be overridden anymore. The defined container is used to store the PDF documents that are uploaded by the authors during the submission phase.

3.3.3 Generate AngularJS Services

It is important to remember that the application is distributed into a client and a server part. When adding or changing a model definition inside the `common/models/` directory, the server part of the application just needs a restart in order to load the new model definitions. However, the part of the application that runs in the browser has no access to that directory. Instead, StrongLoop is able to generate a set of AngularJS services, one for each model, that can be used inside the client. Whenever a model is added or changed, StrongLoop has to re-generate the services. This is done by executing the following command inside the root directory of the project:

⁴ <https://docs.strongloop.com/display/public/LB/Command-line+reference>

⁵ ACL stands for access control list

⁶ Possible relation types are HasOne, BelongsTo, HasMany, HasManyThrough and HasAndBelongsToMany

⁷ <https://docs.strongloop.com/display/public/LB/Defining+models>

```
$ lb-ng server/server.js client/js/services/lb-services.js
```

Please refer to the official documentation for more information about the command and automatic service generation.⁸

Side Note

When starting to write code and using models, note that the usage of model classes on the server differs from that on the client. While the methods basically remain the same, the accepted arguments usually differ. For new developers this can be a confusing fact which is why this disclaimer has been placed here.

3.4 The AngularJS Client Application

As mentioned in section 2.2, the generated code of StrongLoop is targeting the AngularJS web framework. The choice of using StrongLoop therefore also implies the usage of AngularJS. The aim of this chapter is to provide a rough overview of how AngularJS influences the development workflow of the application. After reading this section, the reader should be familiar with the application organization and be able to understand the basic concepts of AngularJS.

3.4.1 The **app** Module

Inside the `index.html` file, the `body` tag is defined as `<body ng-app="app">`. This instructs AngularJS to invoke the module called "app" and execute it. In order to let AngularJS know what to do, the "app" module must be defined and configured in advance. This is done inside the `js/app.js` script. The task of this script is to

- Setup the routes, i.e., the different "pages" that the user can visit. A route does not only consist of a location (URL address) but also of a view⁹ and a controller that provides the logic behind the view.
- Define services that can be used by the controllers. Put simply, a service is a component with a single responsibility that can be accessed application-wide as a singleton instance. Services are most commonly used to abstract the communication between the application and the backend, e.g. to fetch or update data from an REST API. Services are identified by a unique name and can be accessed by controllers (see section 3.4.2) through that name.
- Define custom filters to transform raw data inside views into an appropriate representation. For example, a custom filter could be applied to the value 4239283 in order to interpret it as a file size, e.g., 4.04 MB. Filters help to stick to the DRY programming principle and also keep logic and complex statements separate from views.
- Configure third party services and modules. For example, the application uses the **angular-permission**¹⁰ library to handle access control inside the application according to the users competence (see 3.5.2).

For further information on the concepts of AngularJS, please refer to their official documentation.¹¹

3.4.2 Controllers

In AngularJS, a controller is usually associated with a view and is responsible to manage the logic behind it, i.e., it provides the data that is being displayed by the view and react to user interactions. The data exchange between the controller and the view happens implicitly through the built-in **\$scope** service¹² by using two-way data binding, i.e., whenever a value changes inside the view (through user interaction), the changes are reflected in the controller and vice versa. More information on controllers and the important concept of two-way data binding can be found in the official documentation.¹³

Controllers are defined inside the `client/js/controllers/` directory.

⁸ <https://docs.strongloop.com/display/public/LB/AngularJS+JavaScript+SDK#AngularJSJavaScriptSDK-GeneratingAngularservices>

⁹ A view is basically an HTML template that is embedded in the layout

¹⁰ <https://github.com/Narzerus/angular-permission>

¹¹ <https://docs.angularjs.org/guide/concepts>

¹² See section 3.4.1 for a simple explanation of a service.

¹³ <https://docs.angularjs.org/guide/controller> and <https://docs.angularjs.org/guide/databinding>

3.5 Role Based Access Control (RBAC)

Access control has to happen in both parts of the application, i.e., server and client. On the server side, the application needs to ensure that data can only be accessed and manipulated by users that are actually authorized to do so, i.e., a user should not be able to edit the submission of someone else, unless he is chair of the corresponding conference. On the client side, permissions also need to be checked, e.g., to dynamically display administration links in the navigation bar based on whether the user is chair of the conference.

3.5.1 Roles

The above described scenario is typically modeled by assigning roles to the users of a system. In the case of this application, it makes sense to define the following roles:

- **User:** This role is implicitly assigned to every authenticated user. This is particularly useful to check whether a user is logged into the system.
The following roles are assigned to users conference-wise, i.e., a user can be chair in one conference, but only an attendee in another conference.
- **Attendee:** This role is assigned to users who attend a conference. Since the system is able to manage multiple conferences, this role is useful in order to check whether a user attends a given conference.
- **Author:** This role is assigned to users who have permission to submit papers to a conference.
- **Reviewer:** This role is assigned to users who are allowed to review (assigned) submissions of authors.
- **Chair:** This role is assigned to the chair of a conference. A chair is able to assign roles to attendees of the conference, including this role. The chair is also allowed to edit conference details and deadlines as well as assign submitted paper to reviewers.

Part (I) of this documentation further explains the permissions and accessible features of the particular roles.

3.5.2 The `angular-permission` library

The client part of the application utilizes the `angular-permission` library for role and permission based access control.¹⁴ In essence, it provides two services, the `PermissionStore` and the `RoleStore`. The former service allows to define atomic permissions. A permission is defined by its name and a callback function which must return a boolean value that indicates whether the current user has the defined permission. The later service allows to define roles which are also uniquely identified by a name but consist of a set of permission names. A user is considered to have a role if he owns all permissions that are associated with that role. Each of the five roles described in 3.5.1 are defined inside the `client/js/app.js` file within the `.run()` method.

The library provides the ability to define access permissions for routes, e.g., to permit access to the conference management area to users that have the chair role. Via the `permission` and `permission-only` directives¹⁵, templates can display UI elements dynamically, based on the `permission(s)/roles(s)` of a user.

3.5.3 StrongLoop Access Control List

The server part of the application relies on the StrongLoop ACL mechanisms.¹⁶ Similar to the previous library, StrongLoop allows to specify custom user roles that can be assigned to particular users. Besides the custom roles, four additional built-in roles are defined: `$authenticated`, `$unauthenticated`, `$everyone` and `$owner`. The former two roles are mutually exclusive and state whether the request against the REST API came from an authenticated user. The latter two roles are assigned to every user and the owners of model instances, respectively. For example, a record that is associated with a particular user grants that user the `$owner` role for that record.

These roles can now be used to define the access control for each of the available REST API endpoints. For example, this allows to deny access to DELETE operations to users that are not `$owner` of a record. The individual ACL definitions are found in the model definition files, located inside the `common/models/*.json` files.

¹⁴ <https://github.com/Narzerus/angular-permission>

¹⁵ The AngularJS term for user-defined HTML tags and attributes

¹⁶ <https://docs.strongloop.com/display/public/LB/Authentication%2C+authorization%2C+and+permissions>

4 Hands On: Extending the Application

This chapter describes the process of extending the web application by a new page that displays the current time to the user. This process involves 4.1) the creation of a view and its corresponding controller, 4.2) the creation of a route and 4.3) the modification of the navigation bar to include a link to the newly created route. The instructions in this chapter will help to understand the basic patterns that are used throughout the whole application. Once understood, the application can be modified and extended with ease.

4.1 Creating a View and Controller

The most natural way to start with the extension of the application is to create a new template that is going to display the current time to the user. All application templates are located inside `client/views/`. Because the view is not going to require any special permission other than an authenticated user, it should be created inside `client/views/user/`. While it is not required to follow any particular naming schema, it is common practice to name the view after the controller that is managing it. As our controller is going to be named `TimeController`, the view file should be named `time.html`. Speaking of which, the controller should reside inside the `client/js/controllers/user/` directory and be named `time.js`.

Template Content

Inside the newly created template, the following lines have to be added:

```
<div class="row">
  <div class="col-md-6_col-md-offset-3">
    <p>Hello, it is {{ time | date:'medium' }}</p>
  </div>
</div>
```

The two `div` containers are plain HTML syntax. The application uses the grid system of the bootstrap CSS framework¹. The `div` containers simply reduce the width of the content area in half. The curly brace syntax (`{{ time | date:'medium' }}`) is recognized and interpreted by AngularJS. It basically outputs the value of the `time` variable after passing it to a filter named `date`. More on AngularJS template syntax and filters can be found in the official documentation.²

Controller Content

The controller does only require a few lines of code as well. The following is enough to assign the desired value to the `time` variable:

```
angular.module('app').controller('TimeController', function ($scope) {
  $scope.time = new Date();
});
```

These lines add a controller named `TimeController` to the `app` module that is managed by the angular framework. The second parameter of the `controller()` method defines the function to be executed whenever this controller is invoked by angular. Also note the specified `$scope` parameter inside that function. This parameter will automatically be populated by AngularJS with the instance of the service with the same name. As mentioned earlier (3.4.2) `$scope` is the name of a built-in service in AngularJS. It resembles the namespace that is implicitly made available to the view. More on controllers and the `$scope` service is found on the official documentation pages.³

4.2 Adding a new Route Definition

Now that the view and controller is created, AngularJS must be made aware of the fact that they belong together and also through which route (URL) they can be invoked.

¹ <http://getbootstrap.com/css/#grid>

² <https://docs.angularjs.org/guide/templates> and <https://docs.angularjs.org/guide/filter>

³ <https://docs.angularjs.org/guide/controller>

Reference the New Controller Inside index.html

In a first step, the newly created controller must be included by the browser when loading the application. Therefore, the following line has to be added to the index.html file, alongside with the include statements of the other controllers:

```
<script src="js/controllers/user/time.js"></script>
```

Add Route Definition to app.js

The first `.config()` call inside `app.js` configures the `$stateProvider` which is part of the **ui-router** library.⁴ A lot of `state()` calls on the `$stateProvider` object configures the available routes of the application. On the bottom of this method chain, add the following lines:

```
.state('app.protected.user.time', {  
  url: '/time',  
  templateUrl: 'views/user/time.html',  
  controller: 'TimeController'  
})
```

This defines a new state which is inheriting any configuration options from its parent states, i.e. from the `app`, `app.protected` and `app.protected.user` state. In particular, since it is defined as child of the `app.protected` state, it inherits the access permission (`permissions: { only: ['USER'] }`) (see 3.5.2). Therefore, the newly created state will be available only to authenticated users. Also the defined url under which the state is going to be reachable will be expanded automatically to `/user/time`. For more information on states and configuration options, refer to the official documentation.⁵

4.3 Add a Link to the Navigation Bar

Finally, the time has come to add a link to the navigation bar which then leads us to the newly created page. Find the `<ul class="nav navbar-nav">` element inside `client/views/header.html` and add the following content to the next line:

```
<li ui-sref-active="active" permission="USER">  
  <a ui-sref="app.protected.user.time">Show Time</a>  
</li>
```

The `permission="USER"` attribute ensures that the new link is only displayed for authenticated users. The `ui-sref-active="active"` attribute automatically adds the `active` class to the `` tag whenever the related state defined by `ui-sref="app.protected.user.time"` is detected to be active. The latter attribute automatically adds an appropriate `href` attribute to the `<a>` tag that refers to the url of the given state (`/user/time`).

4.4 Test the new extension

Now it is time to test the created extension. Simply start the server by typing `npm start` inside the project root directory and navigate to `http://localhost:3000`. After logging in with a valid user account, the new link will become visible and, once clicked, will lead to the newly created page which shows the current date and time, nicely formatted.

⁴ <https://github.com/angular-ui/ui-router>

⁵ <https://github.com/angular-ui/ui-router/wiki>

Part III.

Appendix

A Application Installation

The application does not enforce the usage of a specific operating system. However, we recommend to use a Linux system with a package manager in order to install the required prerequisites to make the setup process as smooth as possible.

A.1 Prerequisites

Before starting with the installation process, please make sure that your system does met the following prerequisites:

- Git $\geq 2.x$
- Node.js $\geq 4.2.x$
- npm $\geq 3.x$ (part of Node.js)

The project heavily relies on strongloop for generating API code as well as on bower to fetch 3rd party java-script libraries. To install both on your system execute the following command:

```
# npm install -g strongloop bower
```

A.2 Installation

Use the git command to fetch a copy of the source code repository.¹ The command should be executed by the user who is supposed to run the application and inside an empty directory which is going to hold the application source code.

```
$ git clone ssh://git@scm.informatik.tu-darmstadt.de/iptk-ss2016/iptk-ss2016-  
  ↪ team-whiskey.git .
```

For further installation instructions, please stick to the readme.md file which is now found inside the directory.

¹ Requires read access to the repository.

B Database Entity Relationship Model

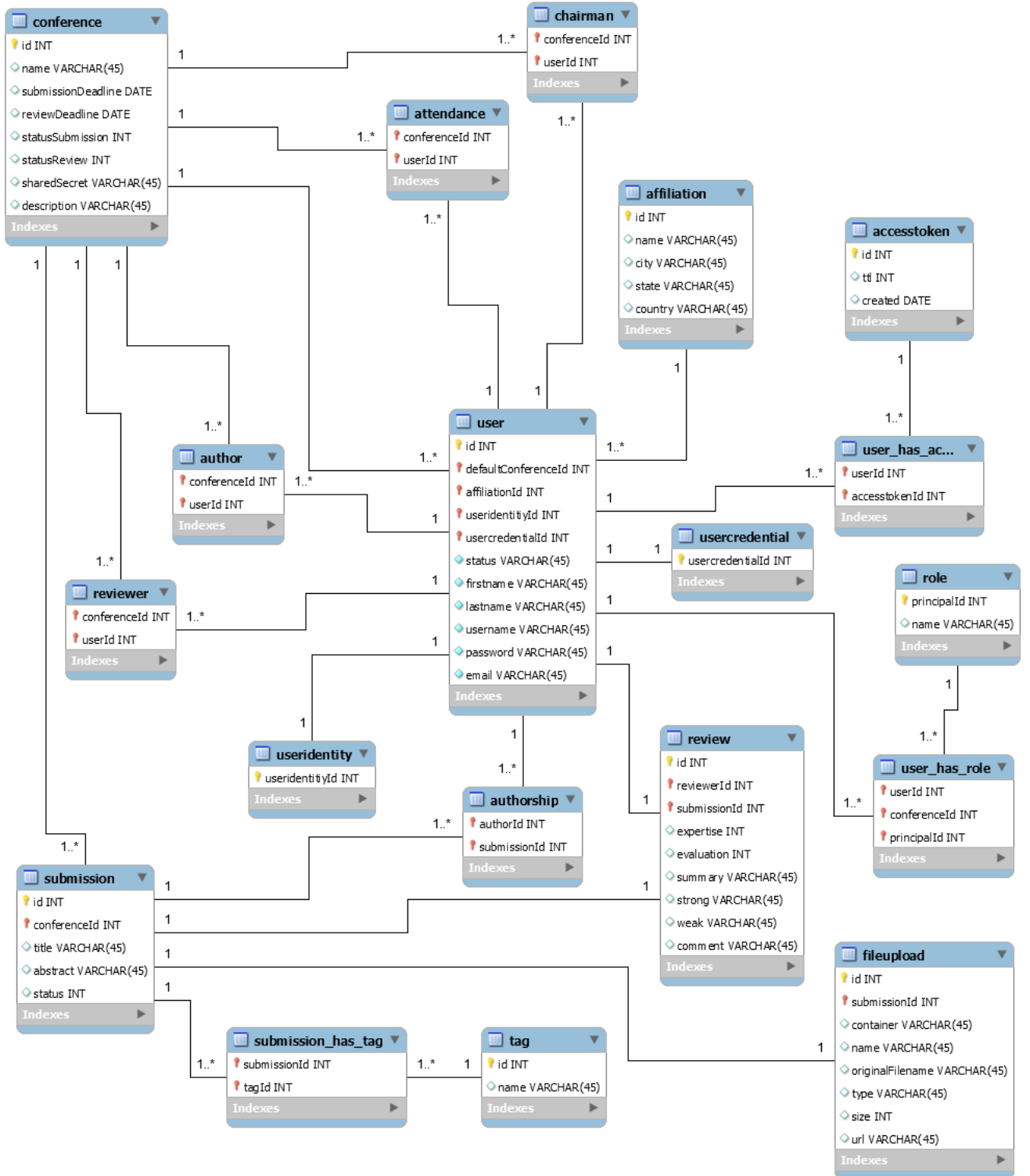


Figure B.1.: The entity relationship model of the application's database.

C Application Screenshots

TKonference - Team Whisky

Start

Login

Username

Password

Login

[Create new account](#)

Figure C.1.: Login page

TKonference - Team Whisky

Start / Create Account

Register

First Name	Last Name
<input type="text" value="Firstname"/>	<input type="text" value="Lastname"/>
E-Mail	Username
<input type="text" value="E-Mail"/>	<input type="text" value="Username"/>
Affiliation	
<input type="text" value="Select or search an affiliation"/>	
Password	Confirm Password
<input type="password" value="Password"/>	<input type="password" value="Confirm Password"/>

Submit

Figure C.2.: Register page. Affiliation field is pre-populated with a list of known universities (~300).

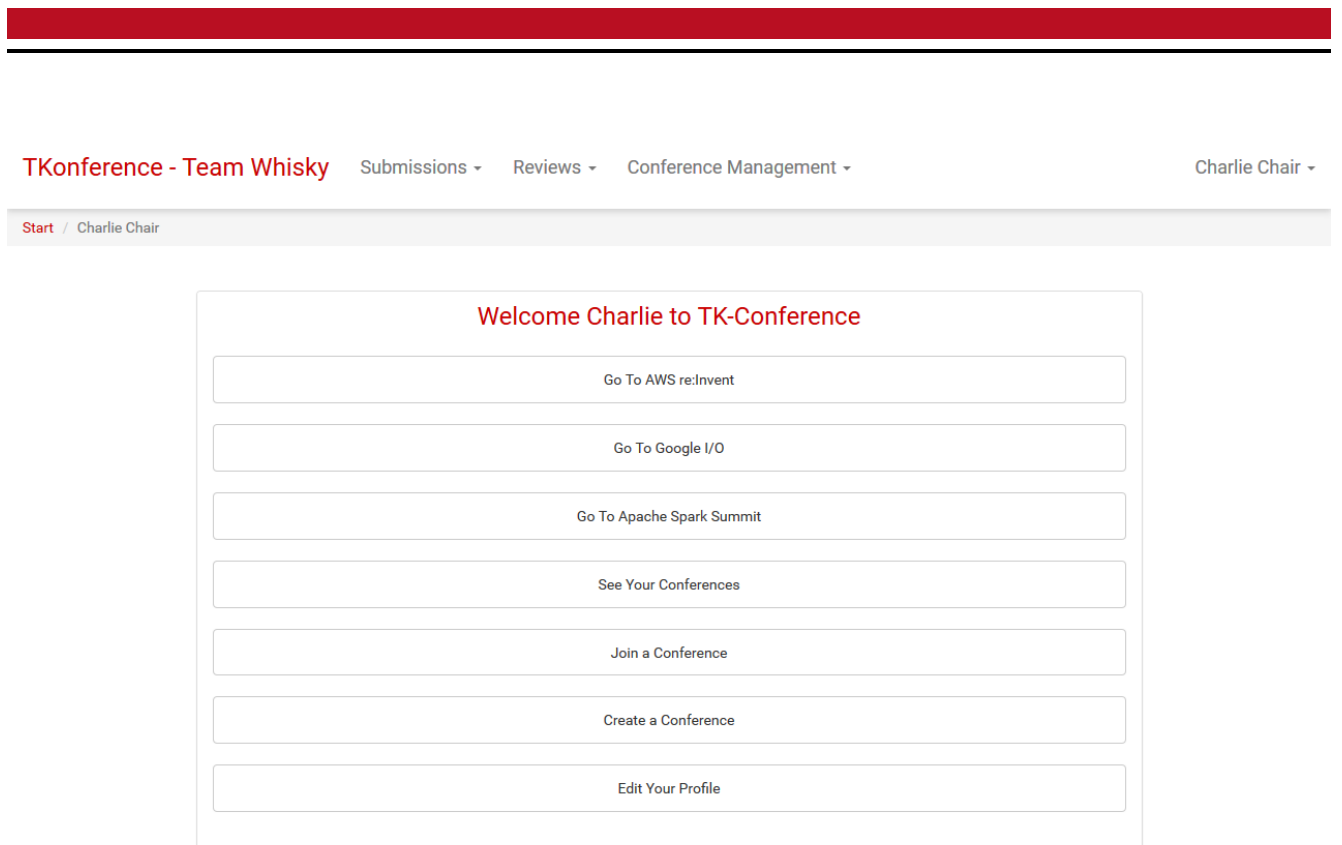


Figure C.3.: Account overview. Provides important quick links

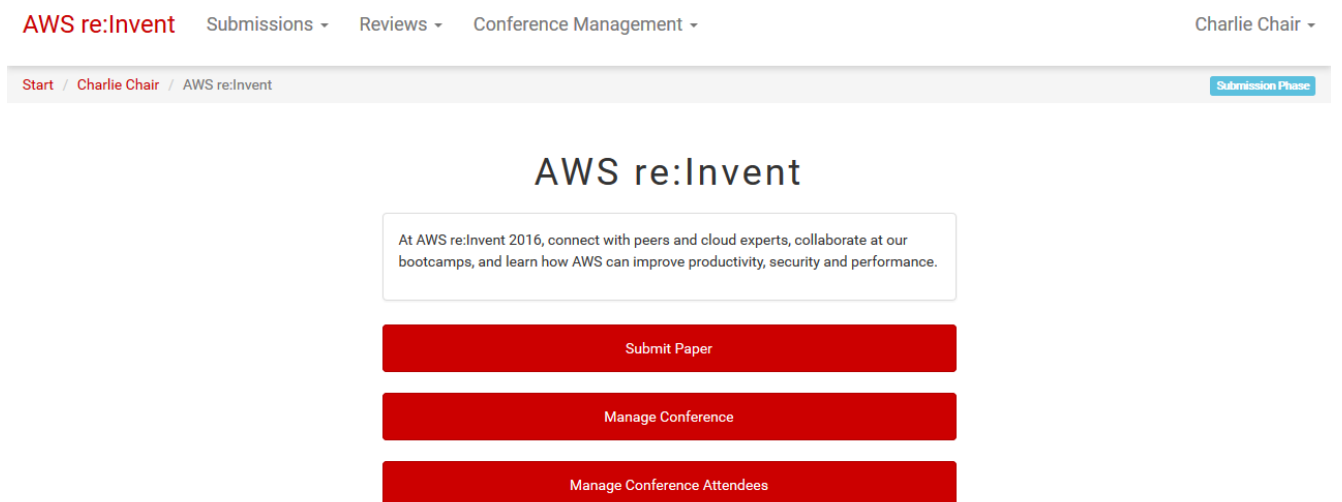


Figure C.4.: Conference overview. Provides some quick links according to the user permissions.



TKonference - Team Whisky Submissions ▾ Reviews ▾ Conference Management ▾ Charlie Chair ▾

Start / Charlie Chair / Edit Profile

EditChange PasswordDelete Account

First Name

Charlie

Last Name

Chair

E-Mail

chairman1@chair.de

Username

chair

Street

Röhnring 15

Postal Code

64283

City

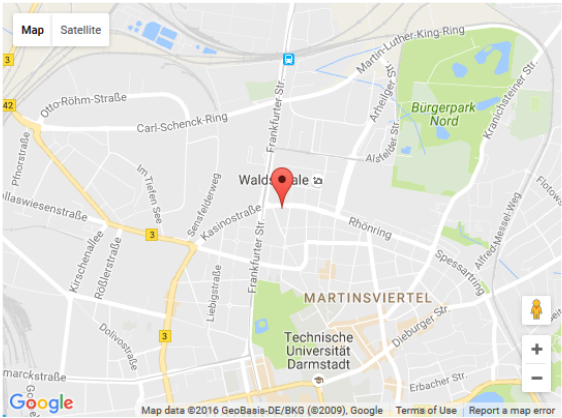
Darmstadt

State

HE

Country

Germany



Map Satellite

Map data ©2016 GeoBasis-DE/BKG (©2009), Google Terms of Use Report a map error

Affiliation

Christian-Albrechts-Universität Kiel

Edit Profile

Figure C.5.: Manage profile. Allows the user to edit his profile, change his password, and delete is account.

20

TKonference - Team Whisky

Submissions ▾

Reviews ▾

Conference Management ▾

Charlie Chair ▾

Start

Charlie Chair

My Conferences

AWS re:Invent Default

You are: Chair, Author, Reviewer
Submission Deadline: September 5th 2016, 12:12 pm
Review Deadline: September 6th 2016, 12:01 am
At AWS re:Invent 2016, connect with peers and cloud experts, collaborate at our bootcamps, and learn how AWS can improve productivity, security and performance.
[Switch to AWS re:Invent](#)
[Show Submissions](#)
[Manage Users](#)
[Manage Conference](#)

Google I/O

Submission Deadline: September 7th 2016, 10:21 pm
Review Deadline: September 7th 2016, 10:21 pm
Google I/O (simply I/O) is an annual developer conference held by Google in San Francisco, California. I/O showcases technical in-depth sessions focused on building web, mobile, and enterprise applications with Google and open sources such as Android, ChromeCandhrome OS, APIs, Google Web Toolkit, App Engine, and more
[Set Default](#)
[Switch to Google I/O](#)

Apache Spark Summit

Submission Deadline: September 7th 2016, 10:21 pm
Review Deadline: September 7th 2016, 10:21 pm
Since our inaugural event in 2013, thousands of developers, scientists, analysts, researchers and executives from around the globe have flocked to Spark Summit to better understand how big data, machine learning and data science can deliver new insights. Whether you're new to Apache Spark or a hardcore enthusiast, Spark Summit is the place to meet with leading experts to share knowledge, receive training and foster valuable connections.
[Set Default](#)
[Switch to Apache Spark Summit](#)

Figure C.6.: List attended conferences. Provides some quick links according to the user permissions.

TKonference - Team Whisky

Submissions ▾

Reviews ▾

Conference Management ▾

Charlie Chair ▾

Start / Charlie Chair / Create Conference

Create Conference

Conference Name

AWS re:Invent

Submission Deadline

09/05/2016 10:23 PM

Review Deadline

09/16/2016 10:23 PM

Allow Submissions

Date

Force Allow

Allow Reviews

Date

Force Allow

Shared Secret

aws

Conference Description

B I H

“ ”

≡ ≡

🔗 🖼️ 👁️

At AWS re:Invent 2016, connect with peers and cloud experts, collaborate at our bootcamps, and learn how AWS can improve productivity, security and performance.

lines: 1 words: 25 0:160

Create Conference

Figure C.7.: Conference creation. Allows to set the name, description and deadlines of a conference. The submission and review phase can be force opened to allow submissions/reviews outside of the defined intervals. The shared secret is used by attendees to join the conference.

AWS re:Invent

Submissions ▾

Reviews ▾

Conference Management ▾

Charlie Chair ▾

Start / Charlie Chair / AWS re:Invent / Submissions / Create

Submission Phase

Create Submission

Title

The title of your paper

1. Author

Charlie Chair <chairman1@chair.de>

2. Author

Select Author... ▾

Abstract

B I H [link] [list] [table] [code] [image] [video]

The abstract of your paper

lines: 1 words: 0 0:0

Keywords

Keywords, e.g., html5, javascript, google-chrome, ...

Enter any numbers of keywords. Press ENTER to confirm each of them.

Paper (PDF)

Drop files here to upload

Submit Paper

Figure C.8.: Submission creation. Allows to select multiple authors from the list of conference attendees. The abstract supports markdown syntax. Keywords are auto-suggested and created on-the-fly. Paper must be provided as PDF.

AWS re:Invent

Submissions ▾

Anton Author ▾

Start / Anton Author / AWS re:Invent / Submissions

Submission Phase

Loop Recognition in C++/Java /Go/Scala

Final

Tags: c++, java, scala, go, performance

Authors: Anton Author, Paisley Herrera, Piper Woodard, Vivian Cole

Status: Draft Unapproved

Abstract: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Attached file: 37122.pdf (310.9 kB)

Edit

Withdraw

A Field Guide to Personalized Reserve Prices

Final

Tags: computer-science, testing, web

Authors: Anton Author, Skyler Lowery, Isabella Gilmore, Claire Jacobs

Status: Draft Approved

Abstract: We study the question of setting and testing reserve prices in single item auctions when the bidders are not identical. At a high level, there are two generalizations of the standard second price auction: in the lazy version we first determine the winner, and then apply reserve prices; in the eager version we first discard the

Figure C.9.: Submission overview. Lists created submissions and shows a summary and the current status. The user can choose to finalize the submission which allows reviewers to review the submission early on.

AWS re:Invent

Reviews ▾

Liliana Marshall ▾

Start / Liliana Marshall / AWS re:Invent / Reviews

Review Phase

A Field Guide to Personalized Reserve Prices

Tags: computer-science, testing, web

Show

Review

Figure C.10.: Review overview. Lists reviews of assigned submissions and allows to edit them during the review phase.

AWS re:Invent

Reviews

Liliana Marshall

Start

Liliana Marshall

AWS re:Invent

Reviews

Edit Review: A Field Guide to Personalized Reserve Prices

Review Phase

Create Review

Submission Title

A Field Guide to Personalized Reserve Prices

Submission Abstract

My Expertise

Very Good	Good	OK	Less	None
-----------	------	----	------	------

Overall Evaluation

Very Good	Good	OK	Less	None
-----------	------	----	------	------

Evaluation Summary

B I H " " " " " "

lines: 1 words: 0 0:0

Major Strong Points

B I H " " " " " "

lines: 1 words: 0 0:0

Major Weak Points

B I H " " " " " "

lines: 1 words: 0 0:0

Detailed Comments

B I H " " " " " "

lines: 1 words: 0 0:0

Draft

Finished

Submit Review

Figure C.11.: Review creation. Allows for an abstract rating and textual feedback (markdown support).

AWS re:Invent

Submissions ▾

Reviews ▾

Conference Management ▾

Charlie Chair ▾

Start / Charlie Chair / AWS re:Invent / Manage Users

Review Phase

Filter

Author

Reviewer

Chair

Author, Anton

Author ☒

Reviewer ☐

Chair ☐

Chair, Charlie

Author ☒

Reviewer ☒

Chair ☒

Herrera, Paisley

Author ☒

Reviewer ☐

Chair ☐

Lynch, Layla

Author ☐

Reviewer ☒

Chair ☐

Kent, Kylie

Author ☐

Reviewer ☒

Chair ☐

Rich, Violet

Author ☒

Reviewer ☒

Chair ☐

Woodard, Piper

Author ☒

Reviewer ☒

Chair ☐

Knight, Lillian

Author ☐

Reviewer ☒

Chair ☐

Cole, Vivian

Author ☒

Reviewer ☐

Chair ☐

Hill, Evelyn

Author ☐

Reviewer ☒

Chair ☐

Johnson, Reagan

Author ☐

Reviewer ☒

Chair ☐

Fuller, Alaina

Author ☒

Reviewer ☐

Chair ☐

Figure C.12.: Role Assignment. Allows the chair to manage the roles of the conference attendees. The filter on top allows to filter for attendees that have a specific role or name.

AWS re:Invent

Submissions ▾

Reviews ▾

Conference Management ▾

Charlie Chair ▾

Start / Charlie Chair / AWS re:Invent / Assign Reviewer

Review Phase

Save Changes

Auto Assignment

Title	Authors	Reviewer
Loop Recognition in C++/Java/Go/Scala	Anton Author Paisley Herrera Piper Woodard Vivian Cole	Skyler Lowery <Skyler.Lowery@gmail.com> ▾
		Paisley Scott <Paisley.Scott@gmail.com> ▾
		Callie Cook <Callie.Cook@gmail.com> ▾
		Select reviewer... ▾
A Field Guide to Personalized Reserve Prices	Anton Author Skyler Lowery Isabella Gilmore Claire Jacobs	Layla Lynch <Layla.Lynch@gmail.com> ▾
		Liliana Marshall <Liliana.Marshall@gmail.com> ▾
		Vivian Ryan <Vivian.Ryan@gmail.com> ▾
		Select reviewer... ▾

Figure C.13.: Reviewer Assignment. Allows the chair to assign one or more reviewers to a submission. This can be done either manually or automatically, or in a hybrid approach.

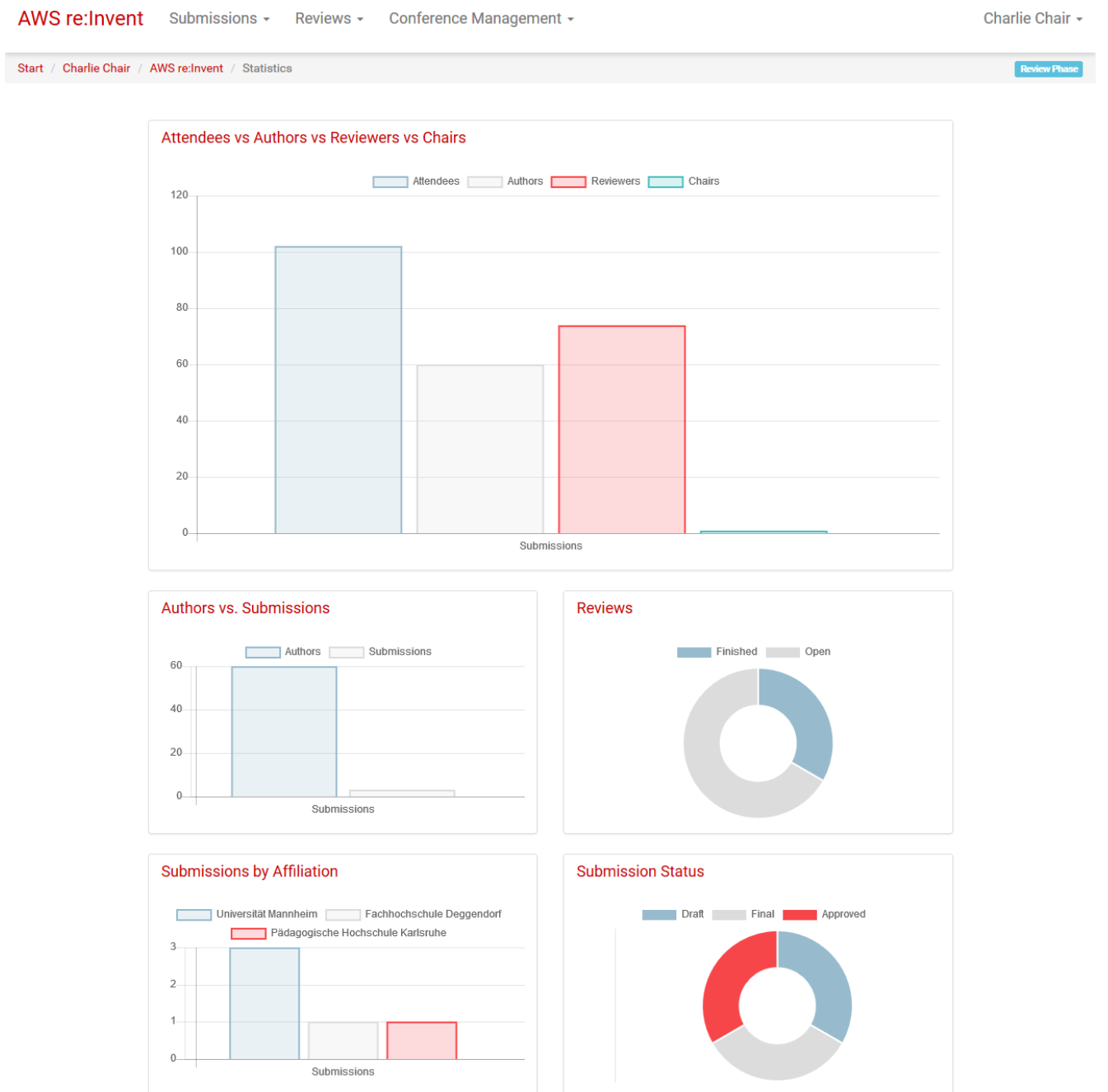


Figure C.14.: Conference Statistics. See 6.3 for more information on the different statistics.

D (Un)Implemented Features

Below is the list of (un)implemented features, copied from the SCM wiki page.¹ A checked box (☑) means that the feature is implemented whereas an unchecked box (☐) means that the feature is (partly) unimplemented

- Roles
 - ☑ Authors [0..*]
 - ☑ Reviewers [0..*]
 - ☑ Chair [1..*]
- General
 - ☑ responsive design: access the web application with different kinds of devices (mobile, tablet, pc..)
- As a user, I can
 - ☑ register to the system at least with email, password
 - ☑ emails are unique
 - ☑ users can have multiple roles (e.g., author and reviewer)
 - ☑ additional information (see slides)
 - ☑ e.g., basic profile (given name, family name, postal address)
 - ☑ e.g., affiliation (institution, city, state, country)
 - ☑ login to the system at least with registered email, password
 - ☑ edit my profile (password, additional information)
 - ☑ remove my profile
 - ☑ logout
- As a author, I can
 - ☑ create a new submission (if submission is open)
 - ☑ title
 - ☑ registered authors [1..*] (given name, family name, email affiliation)
 - ☑ abstract
 - ☑ keywords (multi-selection) that describe the submission (e.g., research fields)
 - ☑ upload the paper (.pdf)
 - ☑ access current submissions (overview)
 - ☑ status: incompleted, completed, closed, accepted, rejected
 - ☑ look at a submission
 - ☑ edit a submission (if submission is open)
 - ☑ withdraw a submission
- As a reviewer, I can
 - ☑ access assigned submissions (overview + status)
 - ☑ look at an assigned submission
 - ☑ details, pdf download or view
 - ☑ make a review > template (if review is open)

¹ https://scm.informatik.tu-darmstadt.de/projects/iptk-ss2016/wiki/Conference_management_system_-_Features

-
- ☒ reviewer expertise: (1) not familiar w/ the topic - (5) expert
 - ☒ overall evaluation: (1) strong reject - (5) strong accept
 - ☒ summary
 - ☒ major strong points
 - ☒ major weak points
 - ☒ detailed comments
 - ☒ edit a review (if reviewing phase is open)
 - As a chair, I can
 - ☒ list all paper submissions
 - ☒ look at a submissions (details, pdf download or view)
 - ☒ withdraw a submission
 - ☒ list all authors
 - ☒ look at detail information
 - ☒ list all reviewers
 - ☒ look at detail information
 - ☒ list all reviews
 - ☒ look at a review
 - ☒ assign papers to reviewers
 - ☒ conflict avoidance: an author is not allow to review his own paper
 - ☒ do the schedule management
 - ☒ automatic: set close deadlines for submissions and reviews
 - ☒ manual: (re-)open/close submissions and reviews
 - ☒ view fancy summary charts or reports (e.g., total submission, acceptances, topics, countries etc.)
 - Bonus features (OPTIONAL)
 - ☐ users must confirm their registration by clicking on a registration link sent via email
 - ☐ email notification to submitters and reviewers (status changes)
 - ☐ chat with other online authors
 - ☒ automatic assignment (even distribution) and conflict resolving
 - ☒ a user can create multiple conference events and becomes the chair (=administrator) of them²

² Implementing this feature sadly broke the server side ACL mechanisms. In **LoopBack**, ACL roles can only be defined globally, i.e. a user can either have the role **chair** or not, but not selectively for only some database records. We noted this after implementing this bonus feature and decided to leave it as is. This means that server side access checks are currently not working and a workaround must be found when used in production mode.