|  |  |
| --- | --- |
| |  | | --- | | Team: 6  Instructor: Enclosed in this document is the technical report of 18655 Service Oriented Computing. | |

Table of Contents

**1. Introduction**

The IEEE (Institute of Electrical and Electronics Engineers) describes itself as "the world's largest technical professional society -- promoting the development and application of electrotechnology and allied sciences for the benefit of humanity, the advancement of the profession, and the well-being of our members." (ieee.org)

IEEE has forth societies where Computer Society is the largest one. IEEE organizes brand-name international technical conferences annually under her societies. IEEE conferences are considered top conferences in the fields.

IEEE conferences are run by volunteers, which are typically researchers, educators, and practitioners in the fields. Each conference decides which software they use to facilitate conference management, based on their specific requirements. From our many years of conference participation and management experiences, so far there is no optimal system that can support a variety of lifecycle conference needs while providing outstanding extensibility, flexibility, reusability, configurability, reliability, etc.

In this project, we aim to design and develop an SOA solution toward fulfilling such requirements. Our goals are to design a system with simplicity, smooth learning curve and lightweight.

2. **Motivation**

It seems like most of the users are so dissatisfied with the current conference system due to their complexity. We simplify the system making it easy to be used.

Without level by level menu, we display all functionalities in limited pages. People don’t need tutorials. Everything is natural. 80% functionalities can be achieved by 20% scale. We keep the system lightweight.

We aim to apply new technologies into this conference management system such like play frame and ORM tech. These new technologies can improve productivity significantly.

**3. Related work**

ConfHub is a traditional conference management system. It’s full-featured and powerful. The functionalities are comprehensive. There are some disadvantages including old fashioned technology framework, complicated menu directory and unfriendly interaction. We want to keep its advantages as much as possible and improve these disadvantages.

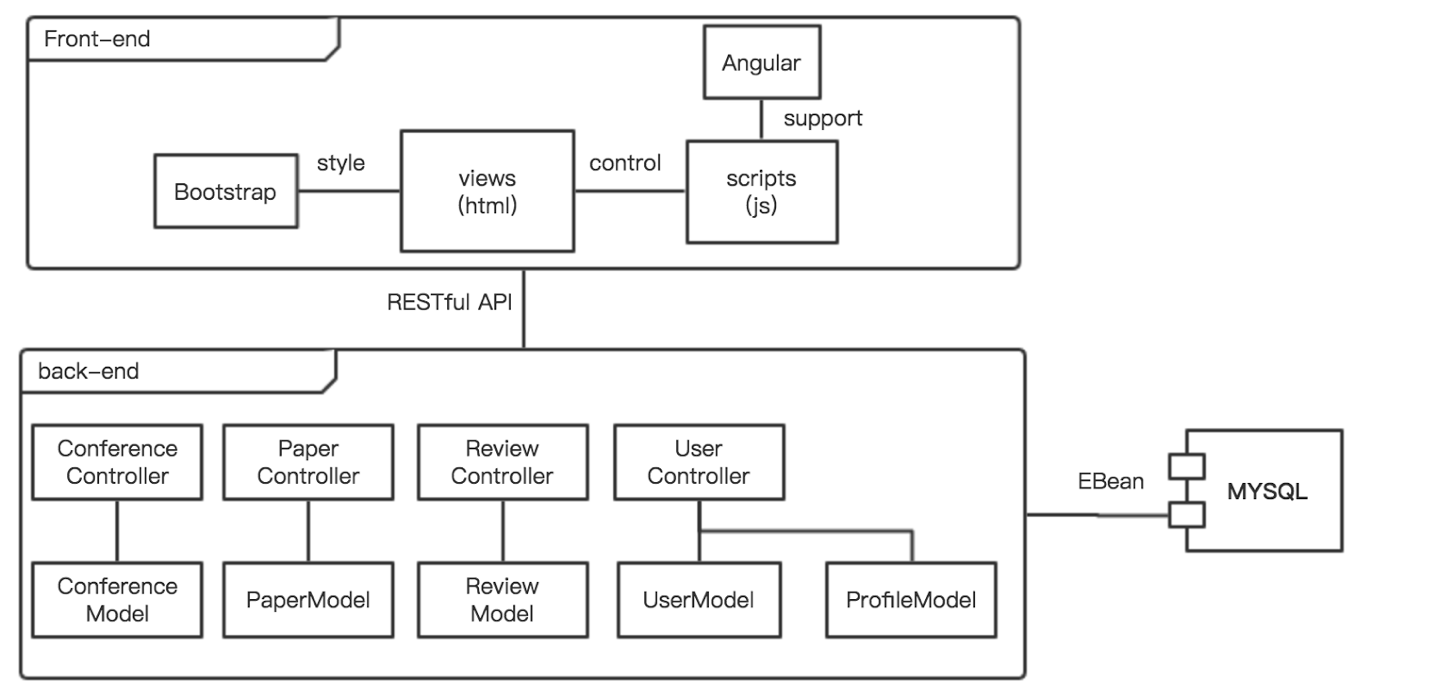
**4. System design**

We use MVC architecture in our system design. MVC stands for model, view and controller. By using it, our system has high cohesion and low coupling. Also, our team members are able to work on different parts of the system without disturbing others.

By making use of a stateless protocol and standard operations, REST systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole, even while it is running. Everyone is able to use the web service we provide.

Between the back-end and database, we use ORM. It stands for Object Relational Mapping. Although we are programmers, we do not want to write each line of code which can be automatically generated by the machine. It reduces a lot of programming time for writing those codes.

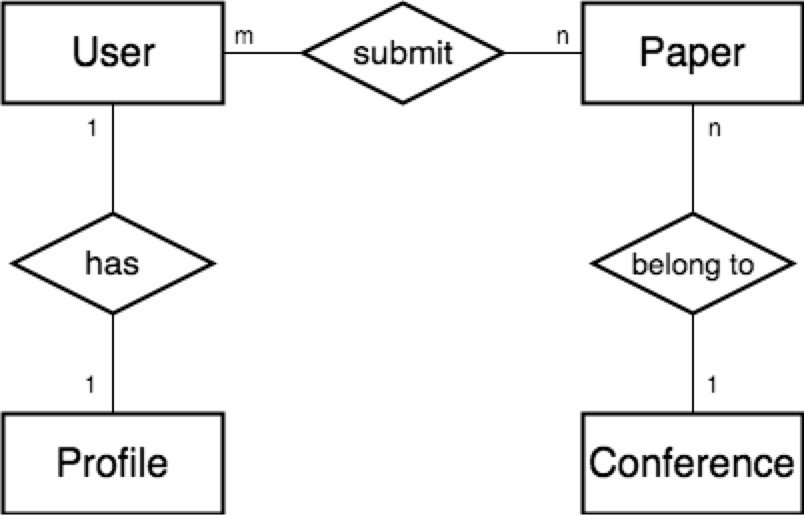
**5. System implementation**



This picture shows our MVC architecture implementation. In the front-end we use Angular.js. In its controllers we use $scope to represent models and finally the data will be used to generate web pages with the style support from Bootstrap.

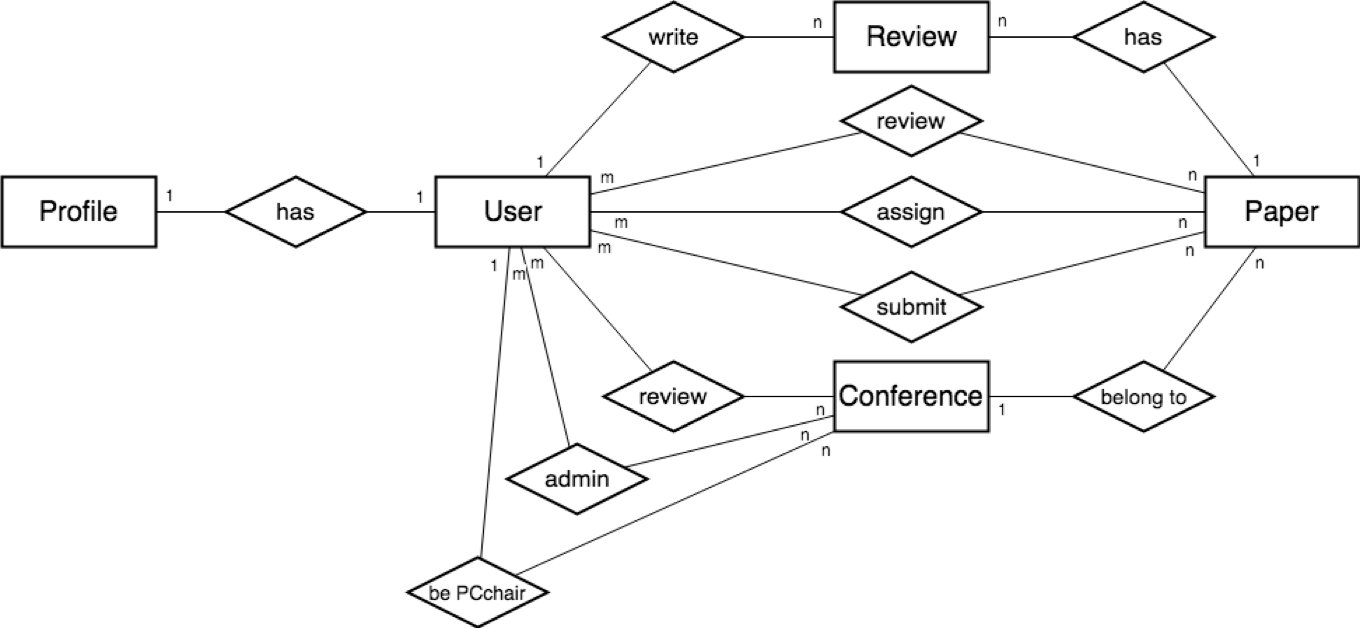
All the interactions between the front-end and back-end are through RESTful APIs. In the back-end, requests will be redirected into different controllers by the router. Then those controllers deal with models and send responses back to the front-end.

When we designed models, we started from E-R(Entity-Relationship) diagrams.



This is the E-R diagram for sprint 1. There are only few use cases, so the diagram is simple. But it contains all the three kinds of relationships: 1-1, n-1 and n-m. In the past, when we were not using ORM, we needed to transfer E-R diagram into database schema, then wrote SQL scripts to create tables depend on it. All the manipulations have to be taken with SQL scripts written by ourselves. But now, we can just write a model in a class and the ORM will help us do the mapping work, which is really convenient.

Here we design a 1-1 relationship for User and Profile. The main reason is that in the User model, it contains some private information such as password and all the attributes in the User cannot be left blank while in the Profile it can. This design incrases the security of our system.

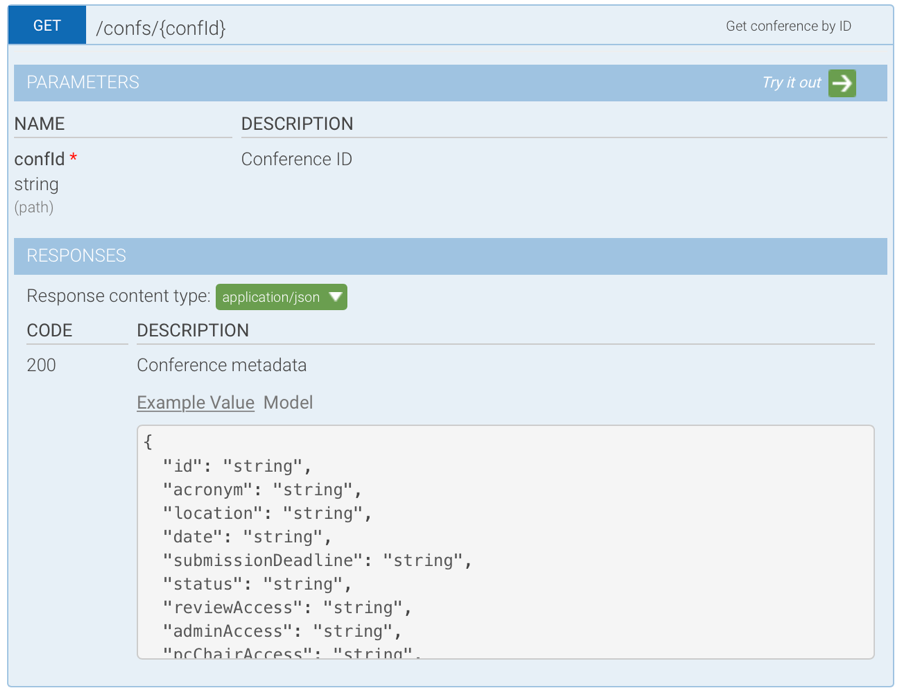


This is our final E-R diagram for the whole system. Because of various privileges, it becomes complicated. In the implementation we once faced some difficulties. In the Play Framework, we can put a JSON object into a response and it will automatically finish the transfer. But because of the relationships, there’s sometimes a recursive issue. To solve this problem, we add “@JsonIgnore” to those attributes.

**6. Technical Practices**

**API Documentation**

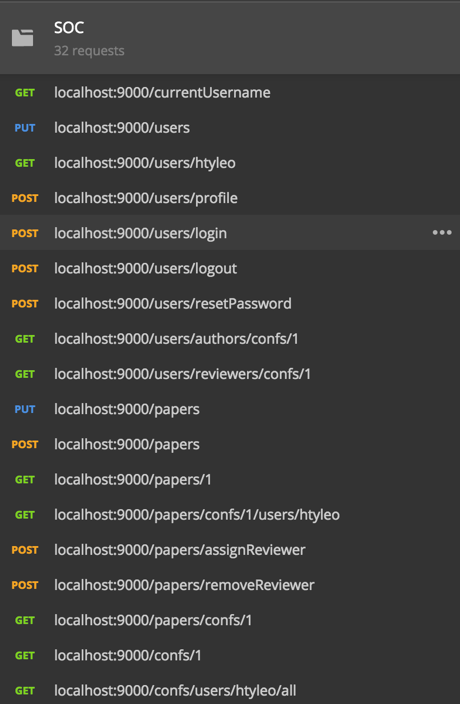
We use SwaggerHub to document all the RESTful APIs. The API document can be accessed at https://swaggerhub.com/apis/socteam6/SOC\_Team\_Project/1.4. There are totally 58 RESTful APIs. For each API, we document its HTTP method (GET, POST, PUT, DELETE), URL, brief description, input parameters with format, response code and response description. Below is the API of getting a conference information by conference’s ID, which illustrates that it takes a single parameter “confId” as the input in the URI and returns a JSON object containing the information of this conference.



In each sprint, we create a new version of API document and add new APIs according to the requirements. All the team members will make a compromise on the APIs before coding. By doing this, we can fully decouple the frontend and backend development, since RESTful API is the only connection between frontend and backend.

**API Testing**

We test RESTful API using Postman, which is a convenient tool to send HTTP requests and render the result. For each API, we commit its implementation code only if it passes the test. This makes sure that every API documented in SwaggerHub functions correctly, which reduces the cost of integration test. By using Postman, we can save the test of each API in a collection, which facilitates the regression test. The following screenshot shows part of the APIs saved in Postman.



**Database Initialization**

Because Ebean clears the database every time we modify the database schema, we need to manually insert testing data into MySQL for test. To reduce the tedious process of manual data insertion, we put the data initialization scripts at conf/evolutions/default/2.sql. The scripts are basically SQL statements, which will insert two conferences with different status into the database. This scripts will be automatically executed once we change the database schema, which facilitates the testing process.

**7. Conclusions and future work**

In conclusion, we implemented the conference management system in three sprints. Starting from March 2017 to May 2017. Our future work includes improving extensibility, simplicity and usability.

**Appendix:**

-Check in everything onto GitHub under the predefined directory including the following items

-Readme file: Describe briefly the purpose of the project, how to download and install the software, how to use the software

-API (sub-directory): instruct APIs as well as descriptions and examples

-Test Suite (sub-directory): a collection of test examples and descriptions

-src (sub-directory): include all source code categorized by packages

-lib (sub-directory): include all related library packages needed to support the project

-conf (sub-directory): include any confirmation settings and files

-app (sub-directory): any applications built on top of the APIs

-contact: please provide every team member’s contact information (cell number, personal email)

-Documents (sub-directory): in different WORD files

-access information: URL, user name/password

-download and installation documents with step-wise descriptions

-executive summary

-background and motivation

-assumptions and considerations

-design documents (architectural design documents and various diagrams e.g., UML files)

-discussions

-presentations (ppt file)

-tutorial: step-by-step usage file with screen shots included

-future work: to-do list and descriptions

-technical report

-Transit the knowledge to either Advisor or a signed student (schedule time to sit down for transition)