Analysis Me

**Spring**

2017

Technical Report

Team: Xiaohan Shen

Huiliang Ling

Shuang Zhang

Keqin Li

Instructor: Jia Zhang

Enclosed in this document is the technical report of 18655 Service Oriented Computing.

# Table of Contents

**1.**    **Introduction**

In this team project, we used SOA/SOC to develop a software system supporting technical conference management.

The software system serves for online paper submission, review, and registration.

Before paper submission opens, the technical program committee (PC) of the conference will be established: a collection of area experts will be identified led by a couple of program chairs (PC Chairs). Authors will submit their papers to this online system before paper submission deadline. First, they need to register a new paper with paper abstract and metadata(paper title, author list and affiliations etc). Second, each paper will be assigned a unique paper id in the system which will be sent back to the authors. Third, authors then use this paper id to upload their full paper to the system. A confirmation email will be sent to the authors and a link will be provided for the authors to verify their submission.

After the paper submission deadline, the PC chairs will assign papers to pc members to review. Each paper will be assigned to at least three pc members. Paper with only abstract submitted will not be assigned for review. PC members will submit their review results before review deadline. Periodically, PC chairs will send reminders to pc members who have not submitted their review results for some assigned papers. Afterwards, PC chairs will make decisions on each paper, whether to accept or reject, based on pc members’ reviews and pc chairs’ own reviews. PC Chairs will mark in the system for each paper’s status: accept or reject. Chairs’ messages are prepared in templates for accepted papers and rejected papers, respectively. Results will be communicated back to the authors of each paper through the system. Authors can go to the system to check the status of their submitted papers.

2.     **Motivation**

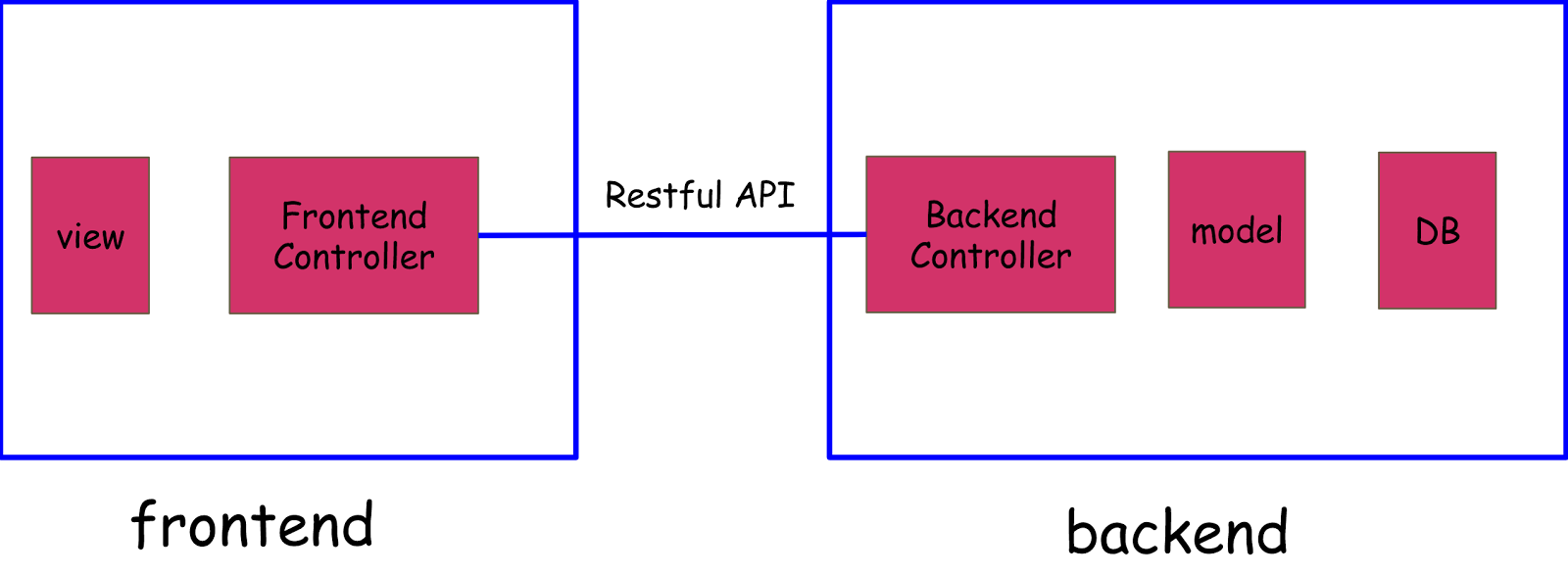
We want to deploy a easy-to-use conference management system.

3.     **Related work**

Confhub: confhub.com

**4.**    **System design**

The system design is shown as Figure 1. In this project we divide the whole project into two parts: frontend and backend, so that some people can focus on frontend/backend without worrying about backend/frontend things. What they need to do is to invoke the services provided in order to implement a whole functionality. In the frontend, there are view and frontend controller. View is how we display the website while frontend controller is to gather data using restful api from backend. In the backend, there are backend controller, model and database. Backend controller can respond to the requests from frontend. Model communicates with database.

****

**Figure 1. Overall system design**

The relation schema of this system is as Figure 2. We use

User - store the login information of registered user

Profile - store the detailed information of a user

ConferenceDetail - store the detailed information of a conference

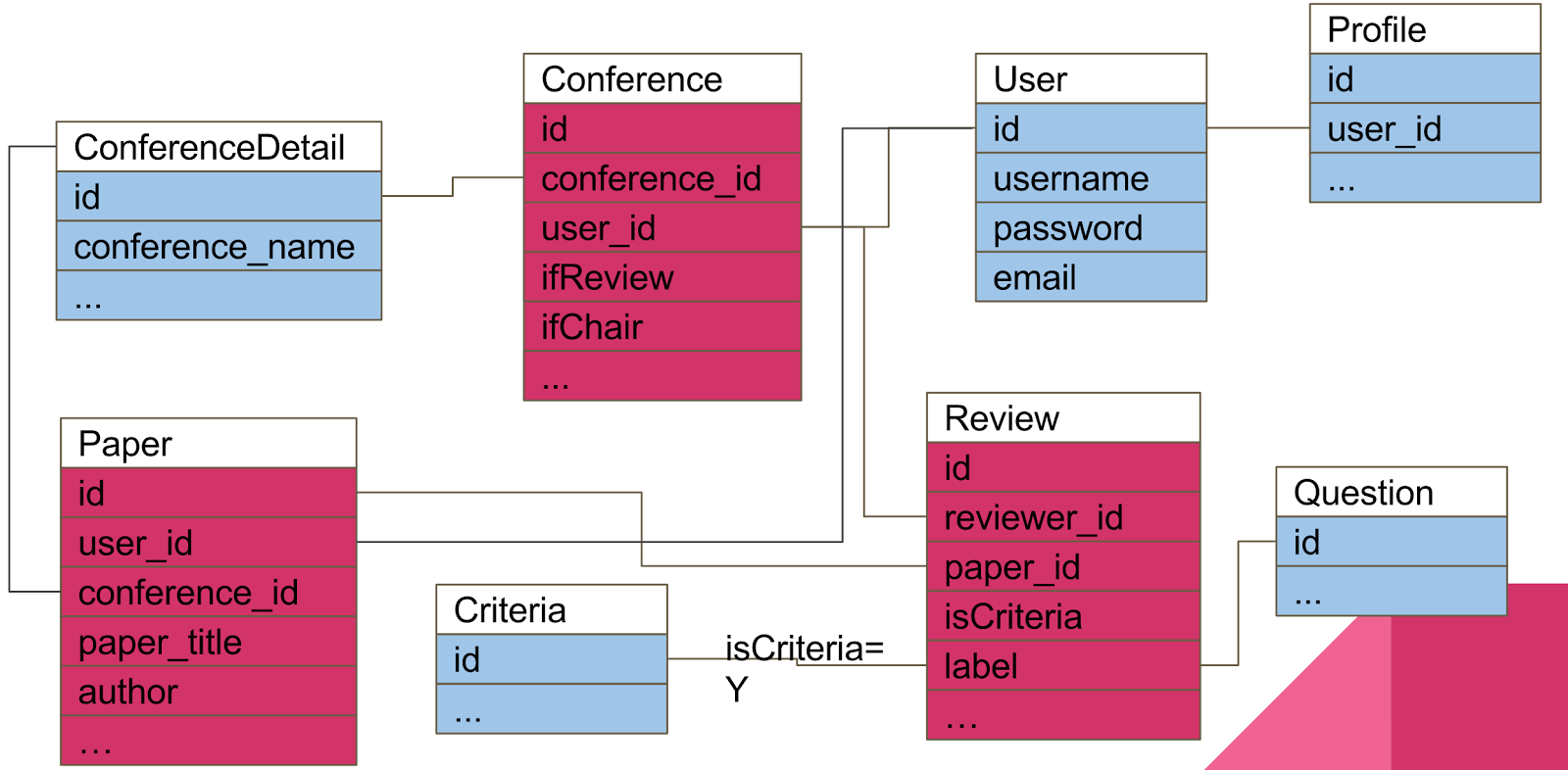
Criteria - review criteria of a certain reviewer to a certain paper

Question - review question of a certain reviewer to a certain paper

Conference - store the relation information of a user and its conference

Review - store the relation information of a reviewer to a paper

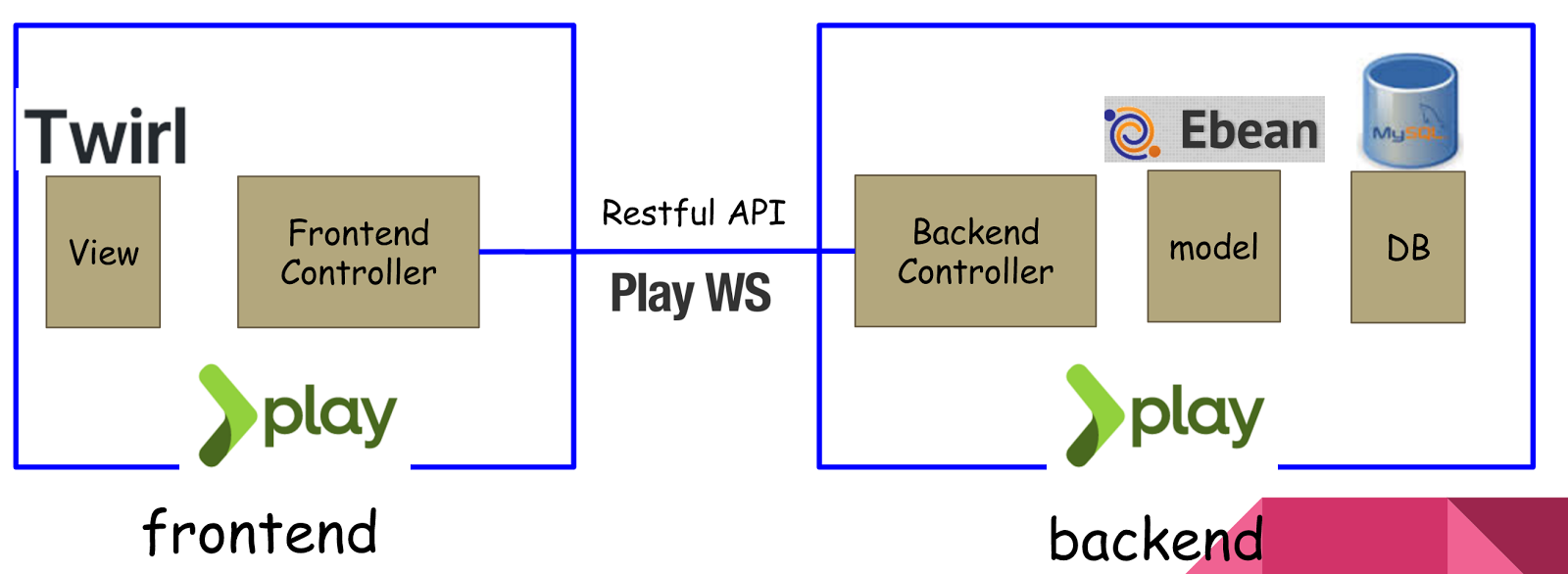
Paper - store the information of a paper(including user who submit it, conference it belongs to, etc)

****

**Figure 2. Schema Design**

5.     **System implementation**

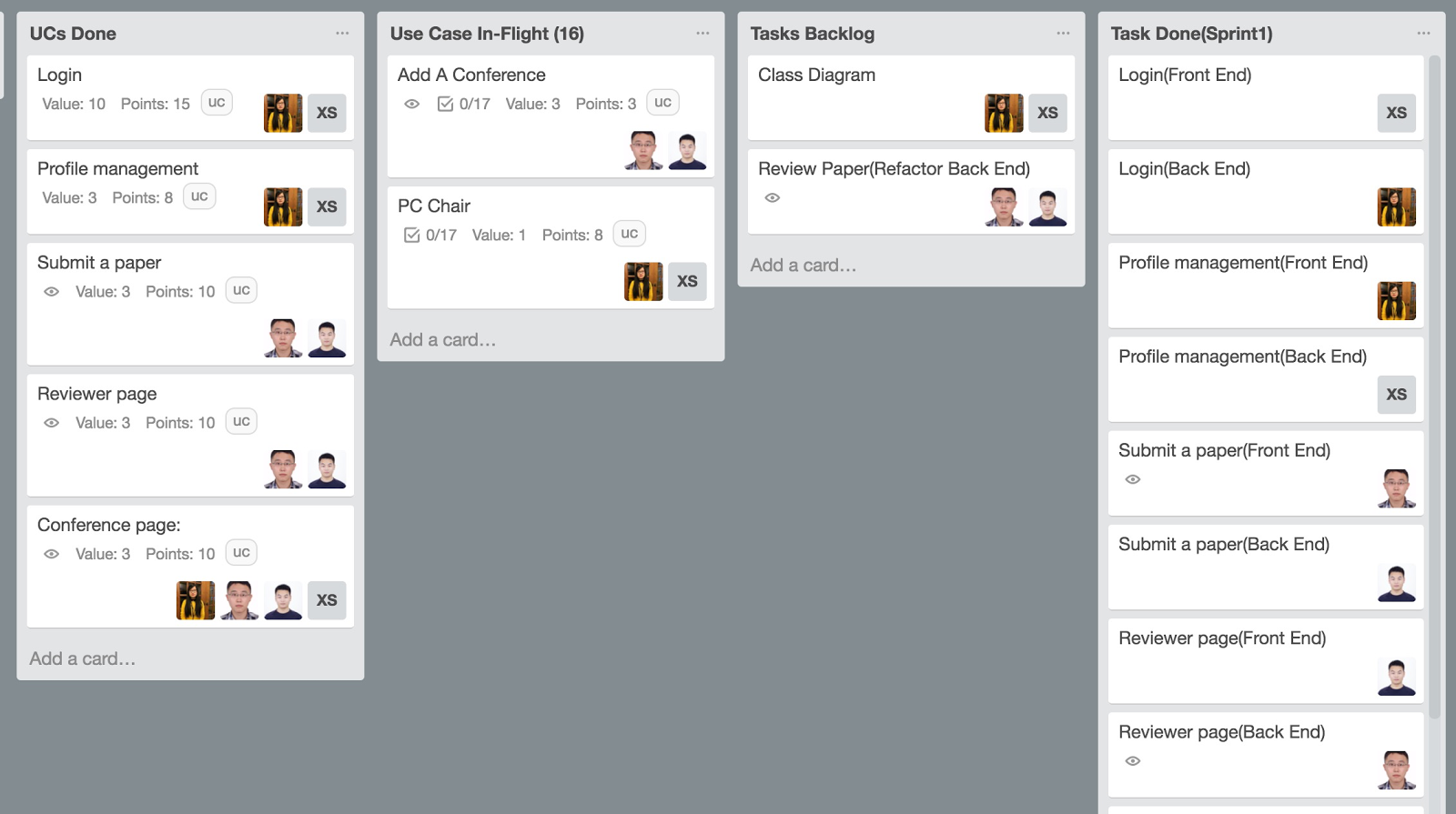
In the system we use two play framework. In the frontend, we use Twirl template engine for view, which is a powerful scala-based template engine. For frontend controller we use play, written in Java. We use Play WS library to send/respond to http request to implement restful api. For backend controller we use play framework. The database we use is relational database, because all the required field is decided. Here we use MySQL as database. We use Ebean ORM to manipulate the database operation with MySQL database. We use sbt to manipulate dependencies.

****

**Figure 3. Overall system implement**

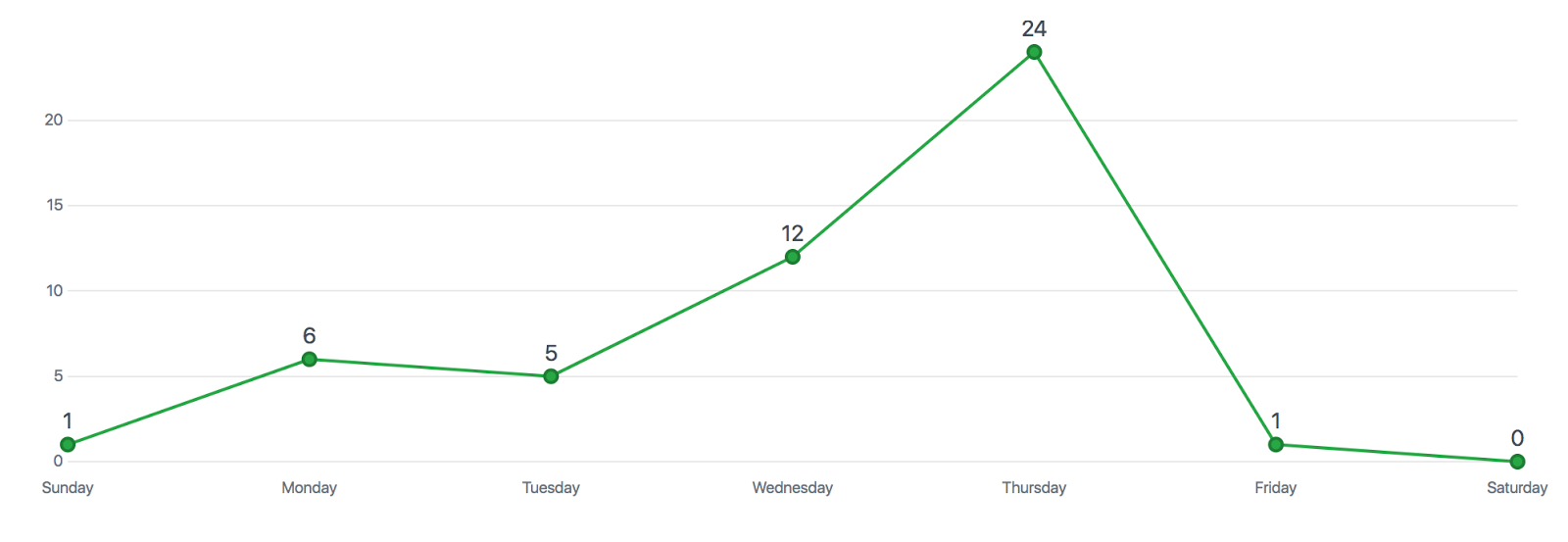
6.     **Experiments and analysis**

1. Spec-based testing: We conducted spec-based tests for all features and requirements and all these tests have passed.
2. Usability testing: We find that some features are not so usable and not easy to test. For example, “review paper” feature has too many dependencies. When a PC chair assigns a paper to some reviewers, they should be able to see this paper in their “review paper” page, otherwise they can never see this page. And when they want to update their reviews, review criteria and questions are synchronized with these in the database and might always be updated by PC chair.
3. We are still in the process of being accustomed to agile development. There are always new and changing requirements that are out of expect. We should keep improving the ability to predict future features. But also we should let our code be more flexible to try to handle requirements even we haven’t thought of.
4. We used Trello to assign the work for everyone. Generally, the assignment is based on detailed requirements.

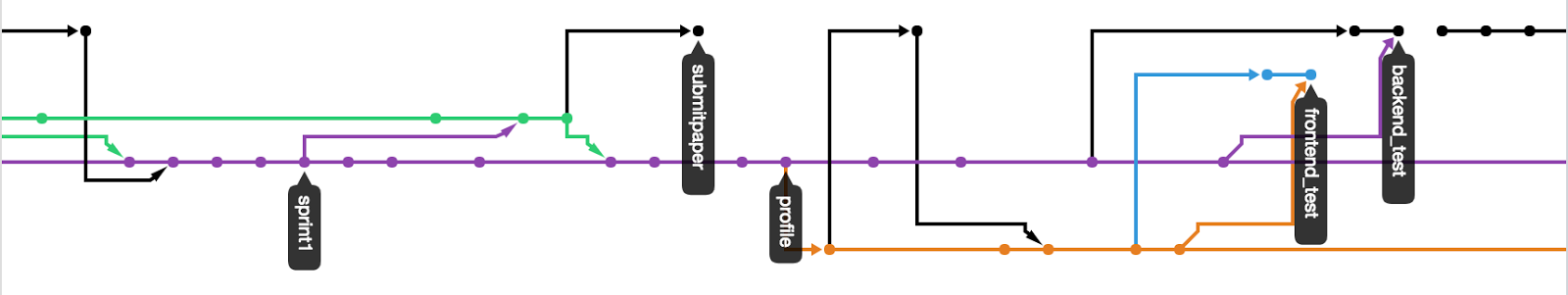


**Figure 4. Trello**

1. We use Github to get our commit distribution and branch tree. Our commit frequency is pretty high in Thursday, almost the sum of all other days because Friday is always deadline. This really reminds us to start early.



**Figure 5. Github commit distribution**



**Figure 6. Github branch tree**

7.     **Conclusions and future work**

Conclusions:

We have completed all features required in all sprints but there are still works to do for both existing functions and future ones.

Future work:

1. We can add more features, for example, managing conference.
2. The system could be more flexible through refactoring. Since there will be many new and changing features, design of database and code should be more flexible to adapt to this situation.
3. Deploy the project on server side. For now, the project is local.
4. Lessen burden of users. For now, one single page might contain too many buttons and redirects. Also, a flow might also include too many steps and pages. Users might easily get confused and make operation mistakes. As a result, we should simplify these complicated flows and pages.

**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)