Analysis Me

**Fall**

2015

Technical Report

Service Oriented Computing (18-655)

Team

Enclosed in this document is the technical report of the <Project Name> sponsored by <Sponsor Name>.

# Table of Contents

1. **Introduction**
2. **Motivation**
3. **Related work**
4. **System design**
5. **System implementation**
6. **Experiments and analysis**
7. **Conclusions and future work**
8. **Contribution of each team member**
9. **Tutorial**

**1. Introduction**

Climate service management is a project for collecting a bunch of climate web services and building a mash up to publish those services.

To provide a user friendly web service, we implement 7 basic functionalities and 3 extra features as follows:

1). Implement a Climate Service main page to list all climate services.

- Show climate service URL, name, and brief description.

- When one user clicks the name of a service, e.g. “ServiceMap 2D”, the website will jump to the chosen climate service page.

- Under this climate service page, user can comment this page.

- The posts carry information like post contents, post time, author etc.

2). A user can discover a climate web service

- User can search for a climate service using keywords.

- The first 3 search results will be listed.

3). A user can view a climate web service (version control)

- All the previous versions are shown this Climate Service’s page.

4). A user can evaluate a climate web service

- User can grade a climate web service (scale: 1-5; 1 is lowest and 5 is highest).

- Climate Service public main page shows the overall grade of every service.

5). Show popular climate services in Climate Service public main page

- Rank the top 3 grades climate service.

- Rank the top 3 most recently used climate service.

6). Within a user’s comment, a user can mention a friend using sign “@”

-e. g. When user input “@”, all of his/her friends will be listed and user can choose one of his/her friend

7). Within a user’s comment, a user can mention a climate service using sign “#”

- User can search for a hashtag e.g. search “@cmu” in search box.

Three extra features:

1). A user can sign up an account and sign in.

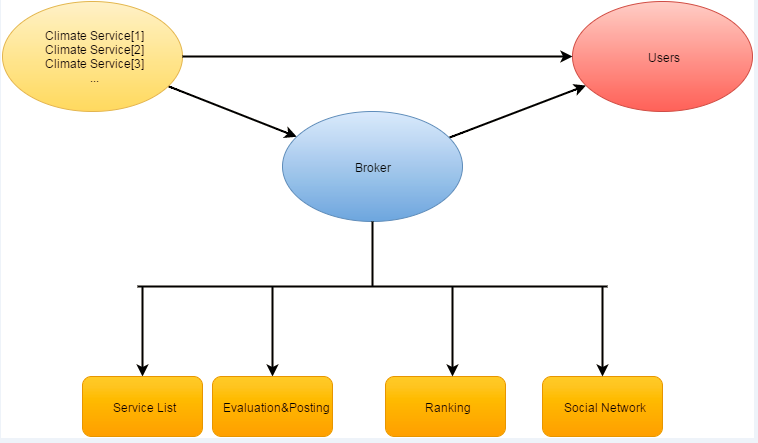
2). A user can add a friend when he/she knows another user’s account e-mail.

3). Service can be ranked based on popularity, in another word, clicking frequency.

**2. Motivation**

There are many different climate services. These services or applications can be provided by different service providers, with different business models, technologies, and different user experience. This requires a user switch between different user interfaces in a complex way to compare services.

As the Web 2.0 mashup technology is rising and becoming popular, we build our service as a mashup. Users like normal customers, scientists can now only visit one portal, which integrates a various services from different providers, without knowing where the services come from. The service is thus provided in a consistent way.



As shown in the figure above, our service can be seen as a broker, providing all climate services in one stop.

There are a lot of benefits to provide such a web service. First, it is an information-centralized system. Instead of publishing all of the services in different places, we collect them together, thus providing more information and utility to users. Second, since the services are collected in one place, it will be more efficient to manage them. Maintenance cost will be greatly deducted in this way. Most importantly, it is a user-friendly system, because a user can now have an easier way to access multiple services in one stop.

**3. Related Work**

1). Set up Docker environment

2). Install IntelliJ IDE

3). Database design

4). Implement the Climate Service public main page to list all climate services sorted by grades in front-end server.

5). Implement API for providing a list of all services to the front-end server.

6). Implement service discovery and searching function and related controllers and web pages in front-end server

7). Implement API for searching top 3 grades services and 3 most recently used services in back-end server.

8). Implement login and sign-up functions and related web pages in front-end server and relate communications mechanism with back-end server.

9). Implement APIs for responding users’ login and sign-up requests, and related controllers to store the users’ information in the database.

10). Implement climate service version control in front-end server.

11). Implement functions for ranking the top 3 grades climate services and most recently used services in front-end servers.

12). Implements APIs for responding users’ searching requests and related controllers’ functions.

13). Implements APIs’ structure design for communicating with front-end server.

14). Implement the comment area in which user can post comment and rate a service with star rating and related communication mechanism with back-end server.

15). Implement the function that a user can mention a friend using sign “@”within the user’s comment in front-end server.

16). Implement the function that a user can mention a climate service using sign “#” within the user’s comment in front-end server.

17). Implement the function for displaying comments in searching result area that mention a service with “#” in searching result in front-end server.

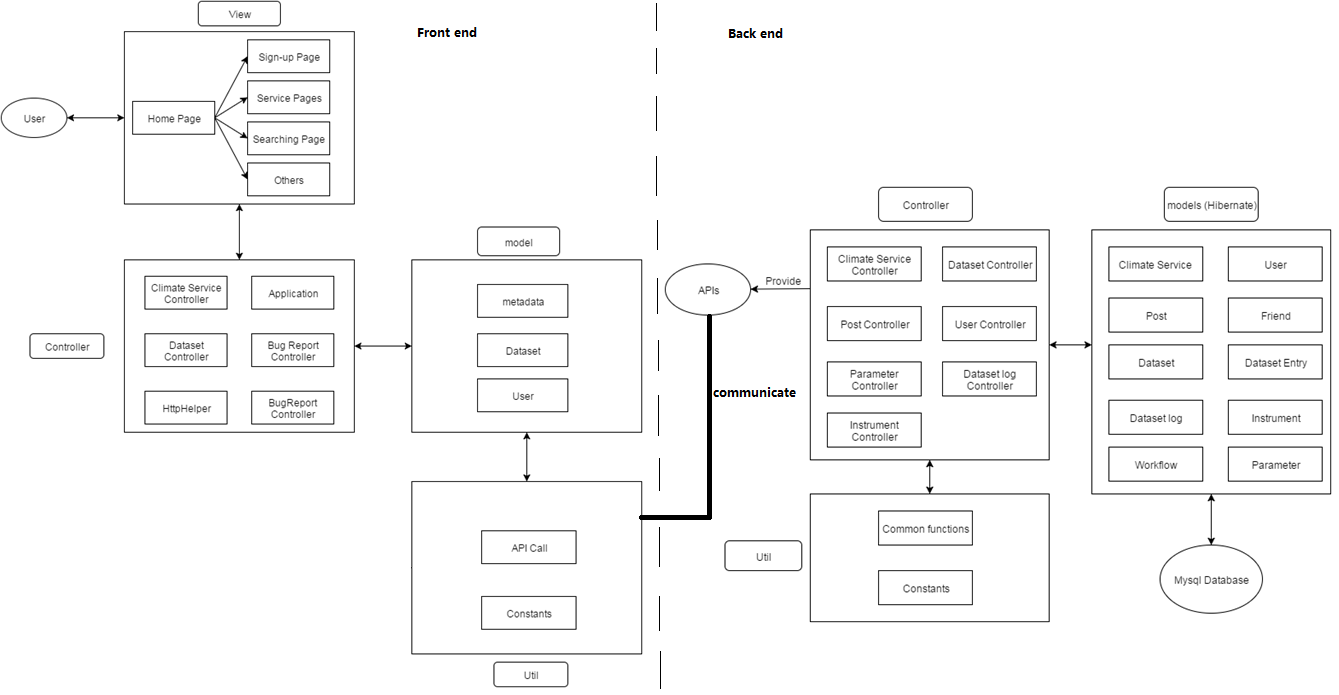
18). Implements API for managing all comments and related controllers for manage the list of comments of each climate service.

19). Implement adding-friends function in front-end server and communications mechanism for interacting with back-end server.

20). Implements API for a user’s adding friend request and related controllers’ functions and database operations.

**4. System Design**

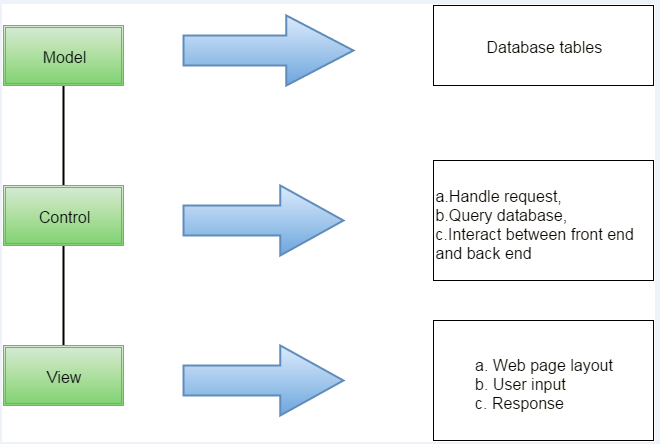
For simplification, the whole system is designed to be two parts, front end and back end, classified by its functionality.



Front end is mainly about user interface, not only includes the graphical web page, but also includes some functionality such as comment, “@” tag, and communication with back end.

Back end receive data from front end and store them in to data base. When receiving requests, it fetches data from data base and give back to front end.

Both in front end and back end, we further divide the design into Model, Control, and View.



As illustrated in the graph, MVC design divide the work into three independent part.

Model layer is mainly about database structure design and CRUD method implementation. All database information are managed in this layer.

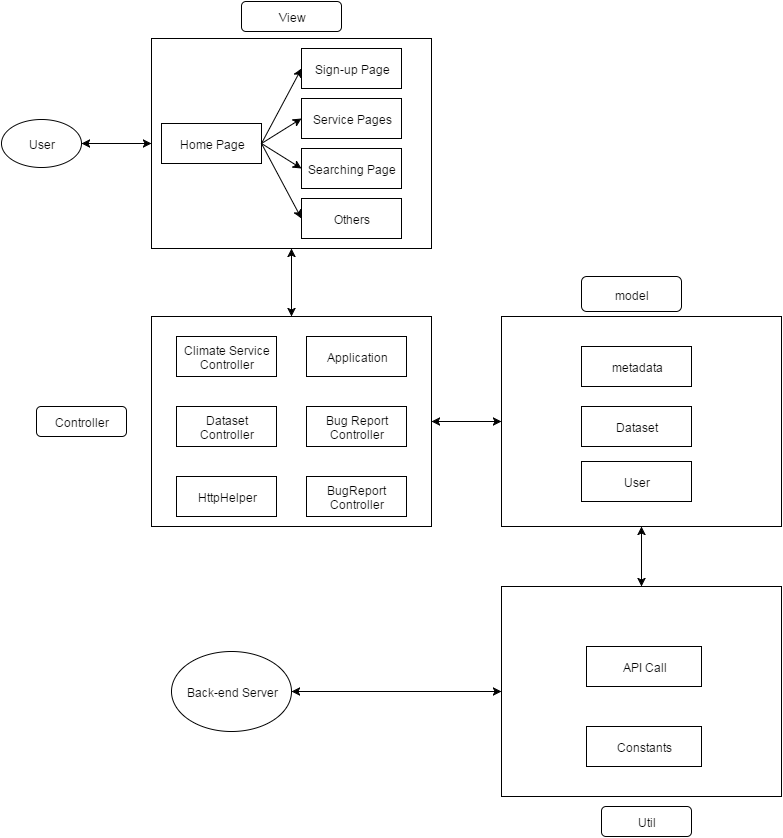
Control layer is the interaction part for front end and back end. It basically handle requests and query database for information.

View mainly deals with the interaction with users. Web page layout is designed in this part. Further, it receives the user inputs and send back the response after business logic.

**5. System Implementation**

1) Front-end server

Front-end server is for collecting user input information, which is built on Play framework. The following is the system diagram of our front-end system:



View part implementation:

In view part, to give a beautiful and comfortable user interface, we used Bootstrap’s star rating plug-in to implement our grade rating part. We also used bootstrap’s other components to build our searching, navigation bar, and comment box in corresponding pages. Javascript and JQuery are used to implement some dynamic contents such as “@” and “#” functions.

Controller:

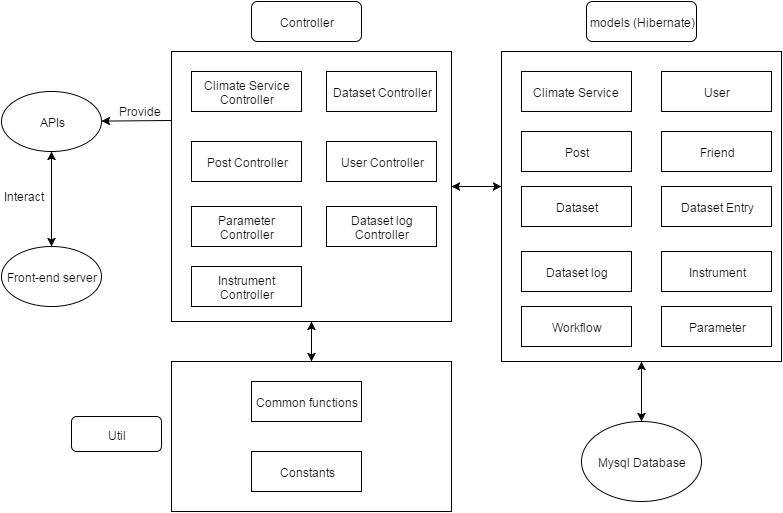
Controller is heart of the front-end server, which will pass the data collected from View part to Model part, and finally send the data to the database in back-end server. Controllers in this system is implemented with a number of specific actions, using java code.

Model and Util:

Model and Util are responsible for the communication java functions between front-end server and back-end server. Model contains the important data class definition for viewing and Util has API calling function for getting or posting data to back-end server.

2) Back-end server

Back-end server is for storing data collected from frond-end server, including dataset, climate services and user information. The system diagram is as follows:

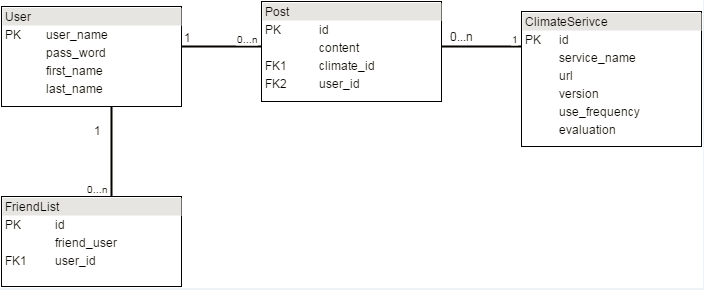


Controller:

When the front-end server sends a request for some information through APIs, controller will access the Mysql database to create, retrieve, delete or update the data and response the front-end server accordingly.

Model:

We use Hibernate to implement our database access functions. There are 10 kinds of data type which we will save in a Mysql database. The key database tables design is as follows:



Util:

Provide some common functions and constants which might be used in controllers.

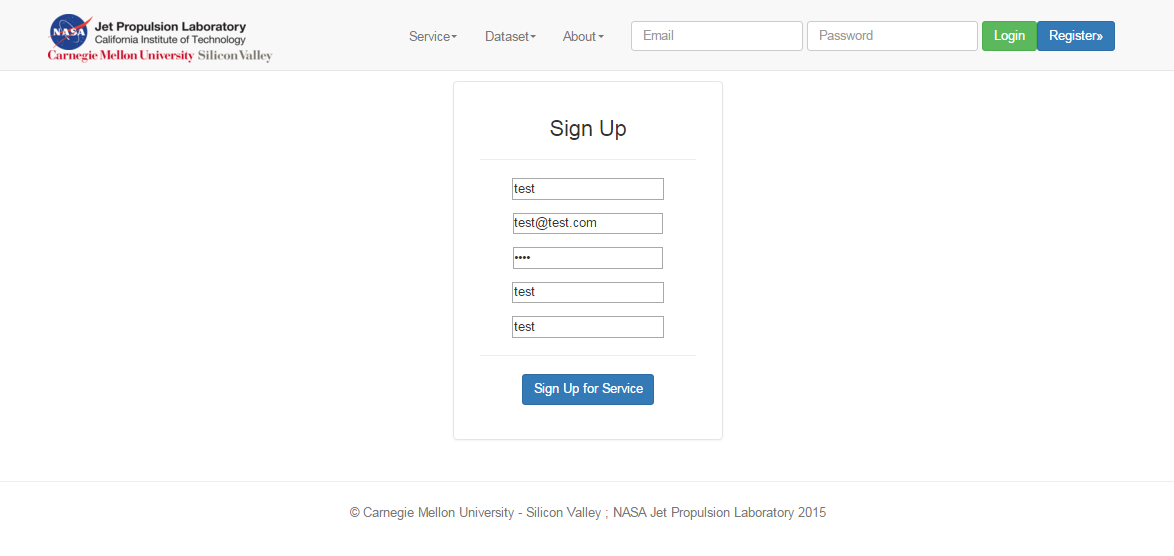
**6. Experiments and analysis**

We will test the following functions of our website and analyze the result:

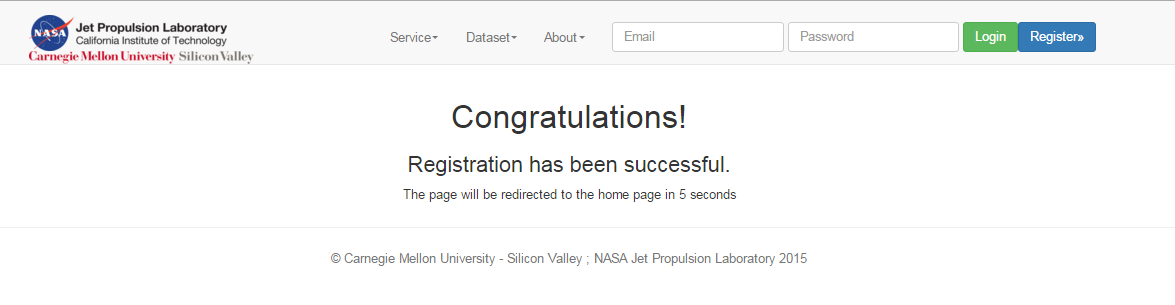
* Sign-up and login.
* Comment a service with rating star grades in comment area.
* Add a friend
* Use “@” hashtag to mention a friend
* Search a hashtag “#” after mention a service
* Search a service
* Get top 3 graded services
* Get most recently used service

***1) Sign-up and login experiment:***

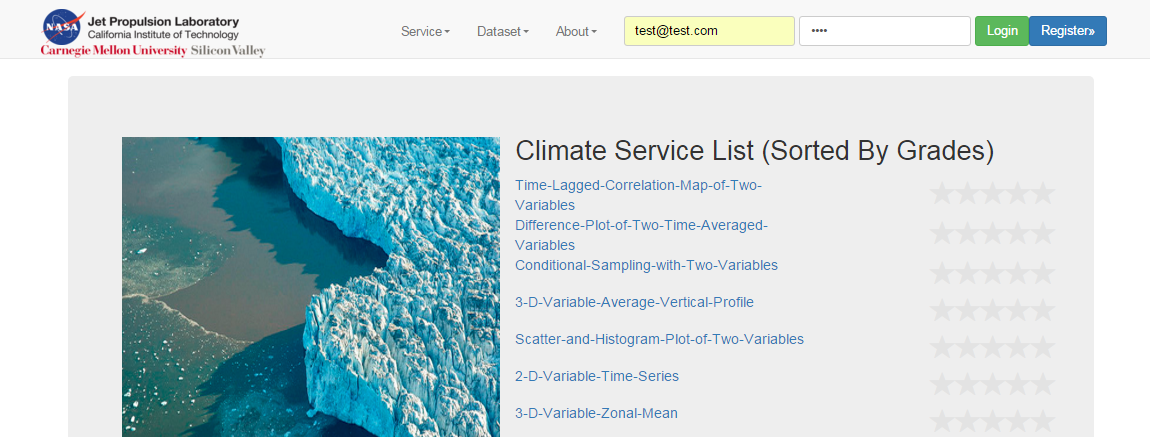
Sign-up page



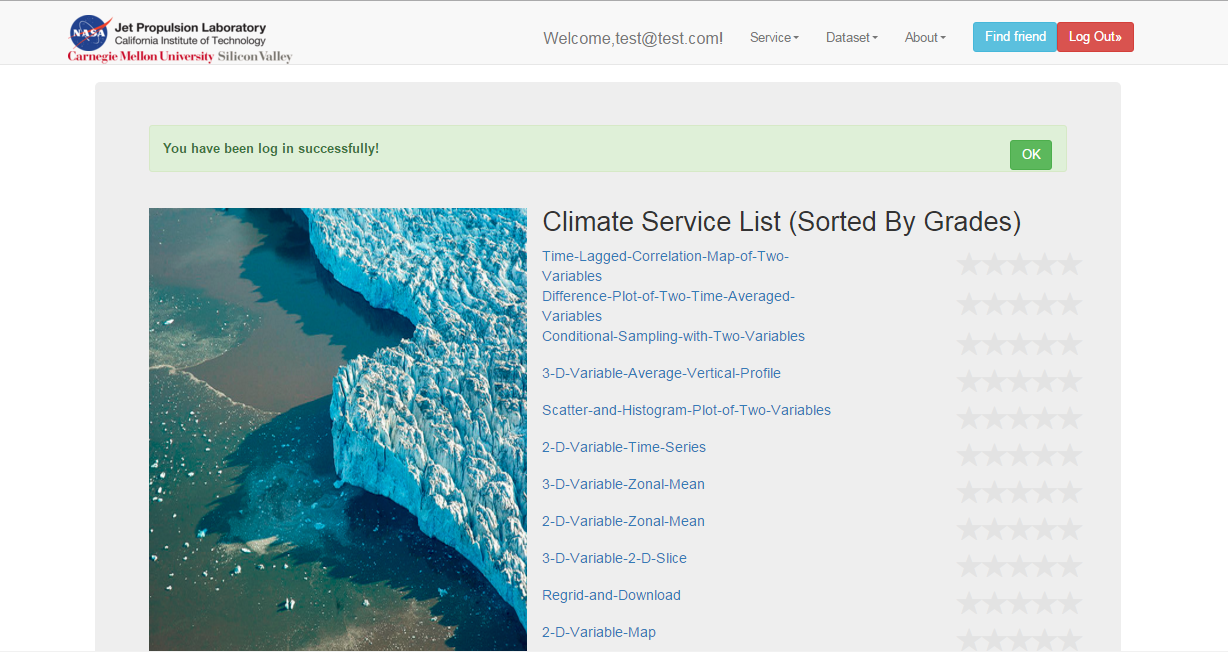
Registered successfully



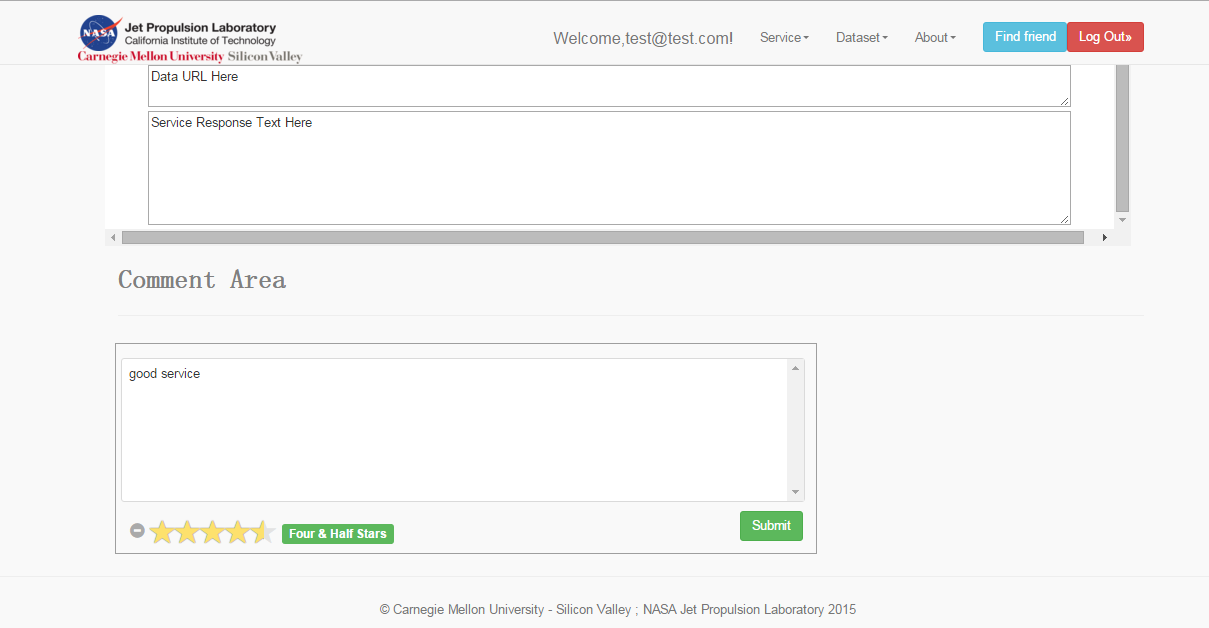
Login test:



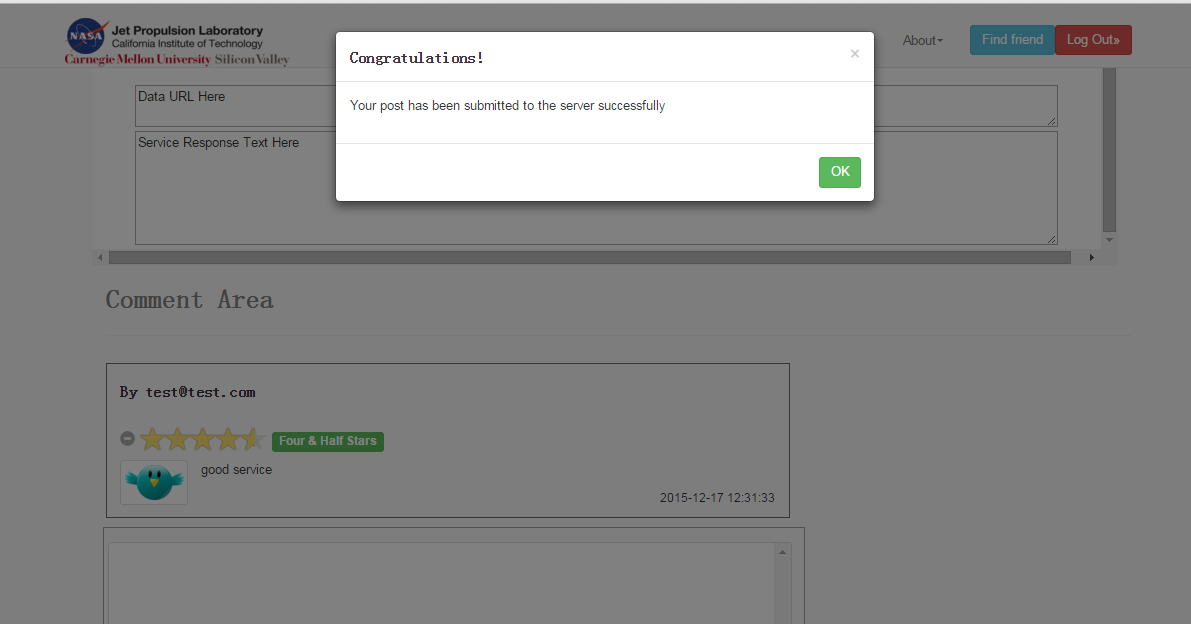
Successfully login the website:



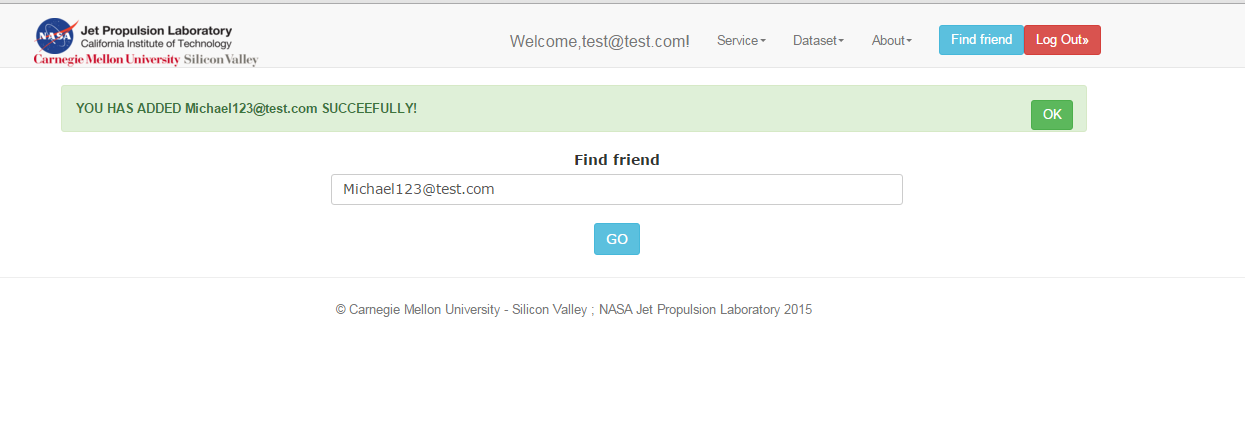
***2) Experiment of commenting a service with rating star grades in comment area.***



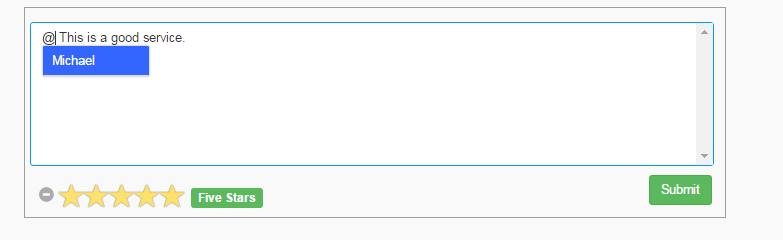
After commented a service successfully:



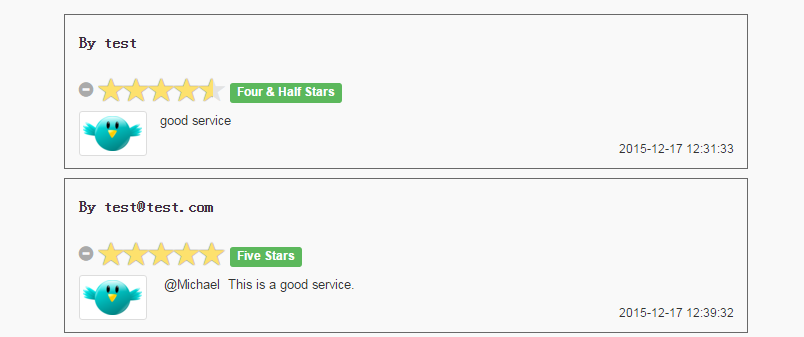
***3) Experiment - Add a friend***



***4) Experiment - Use “@” hashtag to mention a friend***

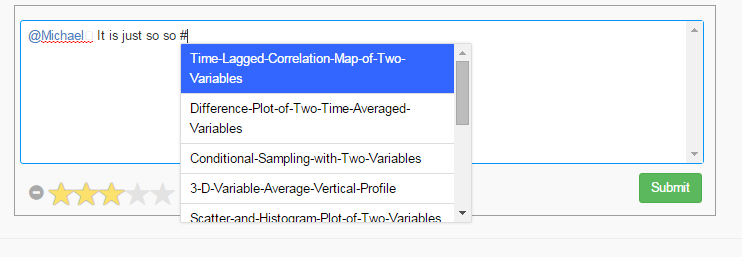


Comment area after successfully commented this service:

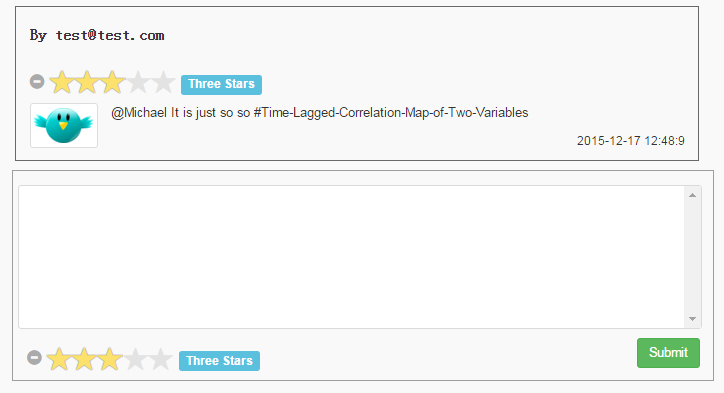


***5) Experiment - Search a hashtag “#” after mention a service***

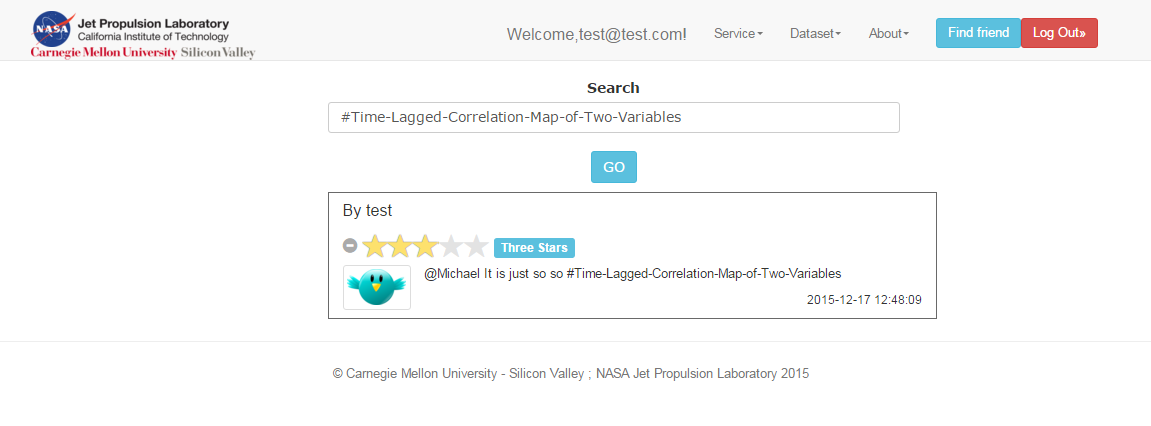
Using “#” to mention a service:



Comment area after using the hashtag “#” to mention a service:

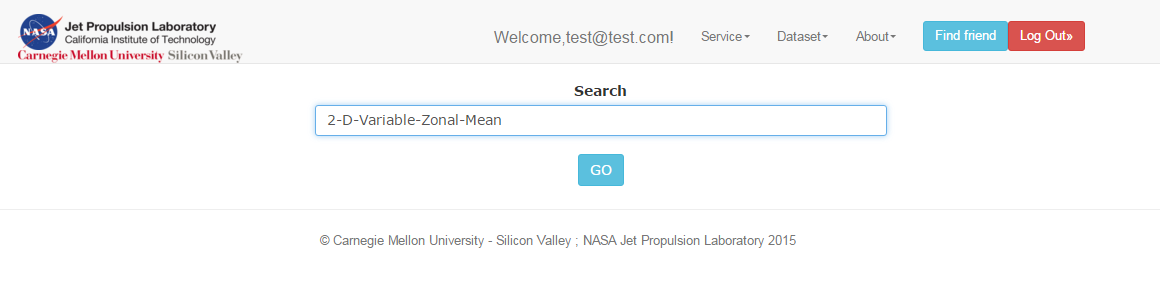


Search the hashtag with the service mentioned:

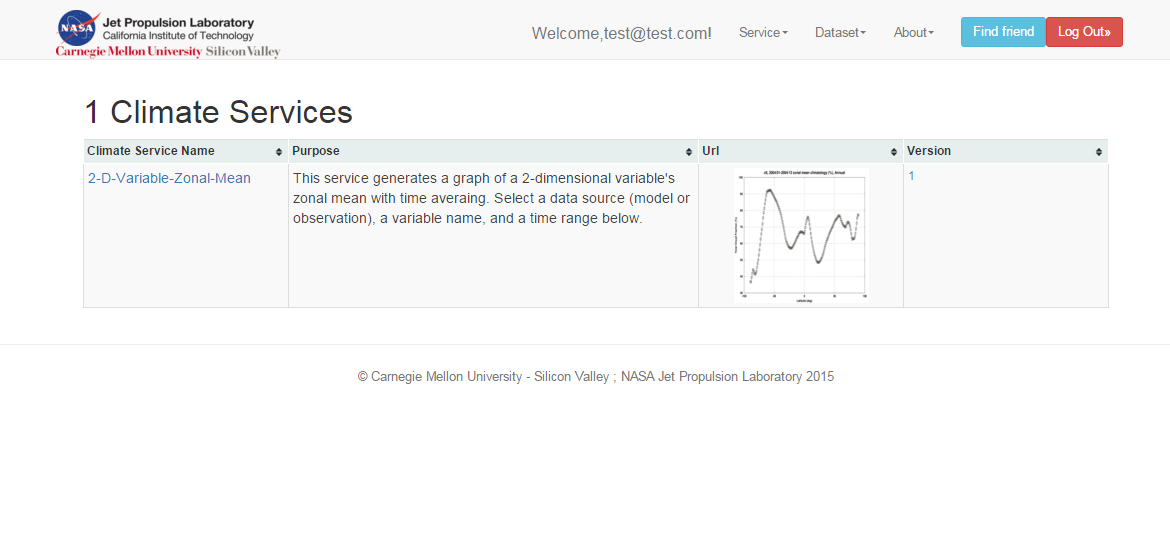


***6) Experiment - Search a service***

Search:

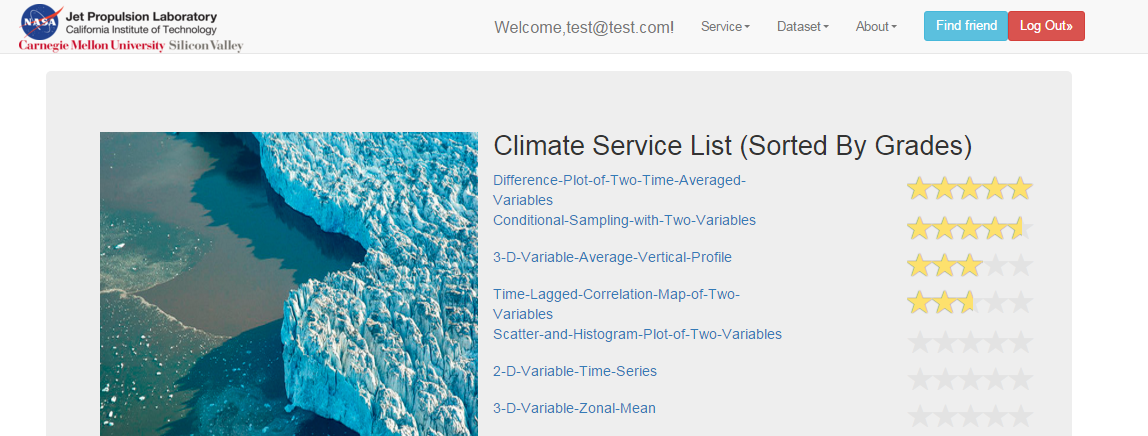


Result:

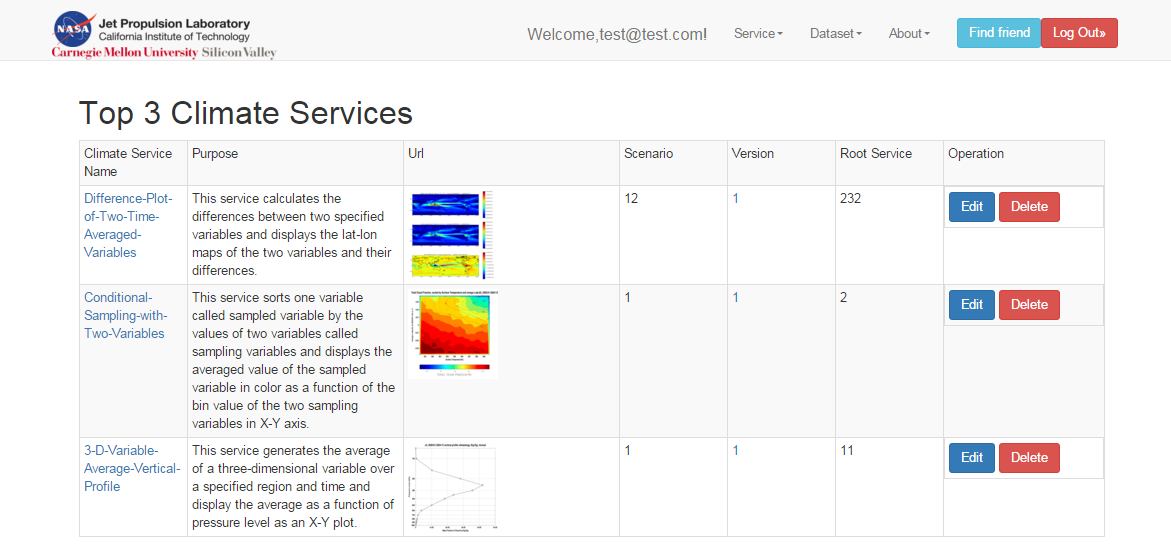


***7) Experiment - Get top 3 graded services:***

Home page service list sorted by grades:

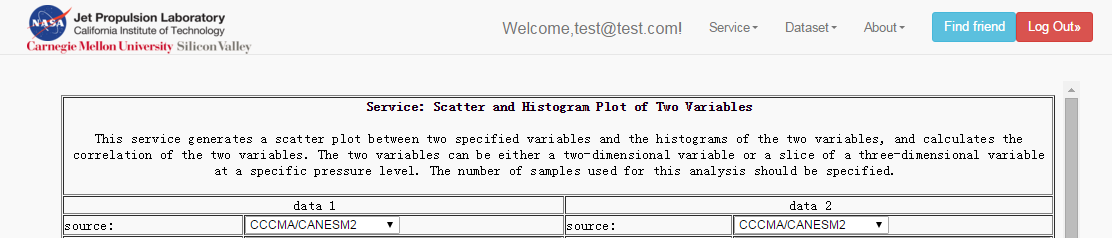


Results of getting the top three grads link

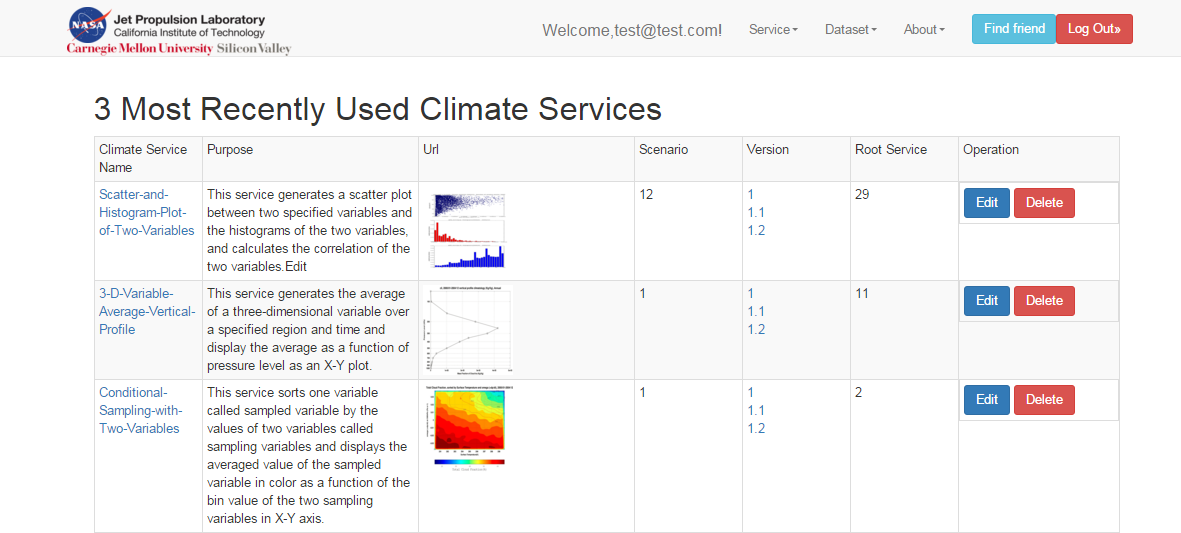


***8) Experiment - Get most recently used service***

First we will click one service and we will check if it is the most recently used service:



Result of getting most recently used service:



As you can see, now Scatter-and-Histogram-plot-of-Two-Variables is on the top of this list.

**7. Conclusions and future work**

***Conclusions***

For this project, we have implemented all of the requirements as follows:

* Implement a Climate Service public main page to list all climate services
* A user can discover a climate web service
* A user can view a climate web service (version control)
* A user can evaluate a climate web service
* Show popular climate services in Climate Service public main page
* Within a user’s comment, a user can mention a friend using sign “@”
* Within a user’s comment, a user can mention a climate service using sign “#”

In addition, we also implemented three additional functions as follows:

* Build sign-up and login pages and related functions by ourselves.
* Adding friends function.
* Get top 3 clicks climate services in a list.

***Future work***

To improve our website in future work, there are three possible ways:

* Build a home page for each registered user.
* User can send messages to each other.
* System can notice a user that when he or she has been mentioned with “@” by a friend.
* Users can manage the climate services that they have posted.

**8. Contribution of each team member**

**Front-end server:**

*Jiyu Shi*

* Implement service discovery and searching function and related controllers and web pages in front-end server.
* Implement login and sign-up functions and related web pages in front-end server and relate communications mechanism with back-end server.
* Implement adding-friends function in front-end server and communications mechanism for interacting with back-end server.
* Implement climate service version control in front-end server.
* Implement functions for ranking the top 3 grades climate services and most recently used services in front-end servers.

*Zhongao Tang*

* Implement the Climate Service public main page to list all climate services sorted by grades in front-end server.
* Implement the comment area in which user can post comment and rate a service with star rating and related communication mechanism with back-end server.
* Implement the function that a user can mention a friend using sign “@”within the user’s comment in front-end server.
* Implement the function that a user can mention a climate service using sign “#” within the user’s comment in front-end server.
* Implement the function for displaying comments in searching result area that mention a service with “#” in searching result in front-end server.

**Back-end server:**

*Yuan Liu*

* Implement data model class design and corresponding CRDU methods for a climate service and a user.
* Implement API for providing a list of all services to the front-end server.
* Implement API for searching top 3 grades services and 3 most recently used services in back-end server.
* Implement APIs for responding users’ login and sign-up requests, and related controllers to store the users’ information in the database.
* Implements API for managing all comments and related controllers for manage the list of comments of each climate service.

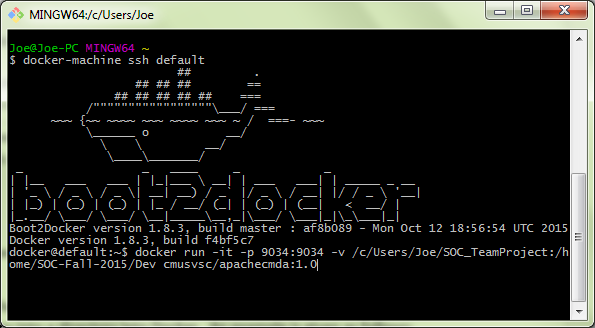
*Xiaoyu Wang*

* Implement data model class design and corresponding CRDU methods for users’ comment and users’ friend list.
* Design the database table structure in back-end server.
* Implements APIs’ structure design for communicating with front-end server.
* Implements APIs for responding users’ searching requests and related controllers’ functions.
* Implements API for a user’s adding friend request and related controllers’ functions and database operations.

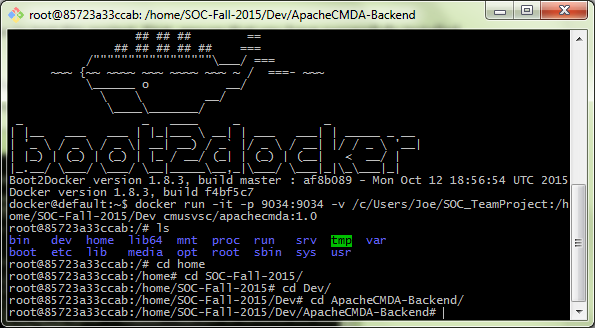
**9. Tutorial**

We have provided a tutorial for how to test this project. First, ensure that you have successfully installed Docker and related software.

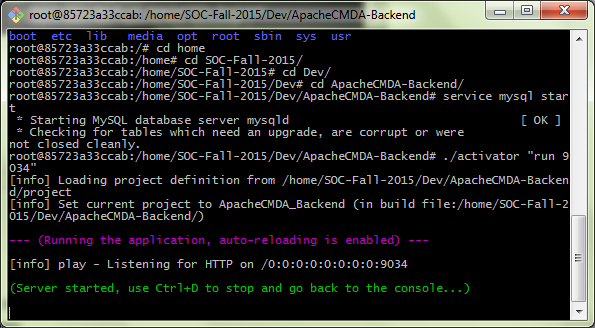
1) Load the provided image which has already configured the environment for this project. In addition, mount the back-end server directory into a directory into Docker. An example is given as follows:



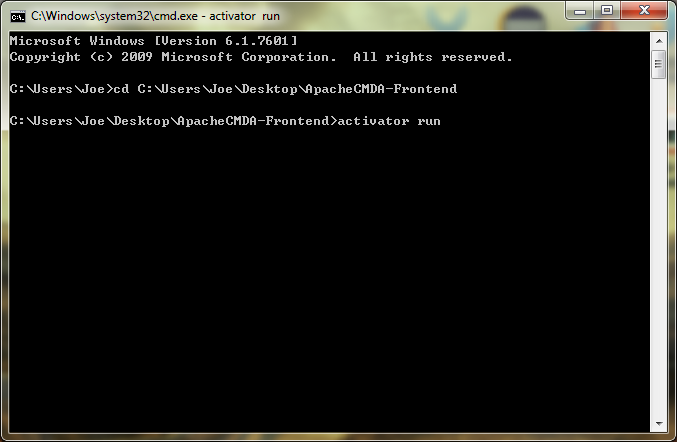
2) Change the directory in the Docker image to the directory mapped to the back-end directory in local machine. An example is given as follows:



3) Start the mysql service and start the back-end server at port 9034 with the command: ./activator “run 9034”. An example is given as follows:



4) Start front-end server in its directory at local machine, and an example is given as follows:

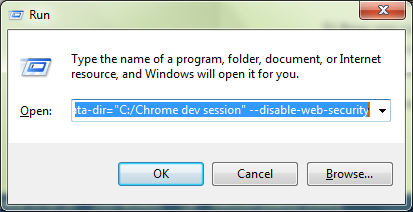


5) Now you can test our website in local port “9000”, please disable the web-security and enable cross-origin resource sharing of your browser because it might block our local AJAX request to back-end server. An example is given as follows:

Run the chrome browser in windows system:

chrome.exe --user-data-dir="C:/Chrome dev session" --disable-web-security

Following is the example graph:



6) Test the website at the browser and the URL is <http://localhost:9000/>, and the following is an example:

