Team 5

Fall | 2015

Technical Report

Data Intensive Workflow Development for Software Engineering (18-656)

Table of Contents

Introduction

DBLP

DBLP is a well-known service that provides open bibliographic information on major computer science journals and proceedings.DBLP indexes more than 3 million publications' metadata.

Neo4j

Neo4j is an open-source graph database implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint.The developers describe Neo4j as an ACID-compliant transactional database with native graph storage and processing.Neo4j is the most popular graph database.

Motivation

Create the data analytics service for DBLP:

Build a network between authors and publications

Develop a user-friendly interface

Help users get what they want effectively, efficiently and accurately:

Improve functionality of queries

Visualize the answer

Related Work

Graph Database:

A database management system with CRUD options working on a graph data model Good at handling data relationships

MySQL:

Use relational database to store user information

Spring framework:

Provides a comprehensive programming and configuration model for modern Java-based applications

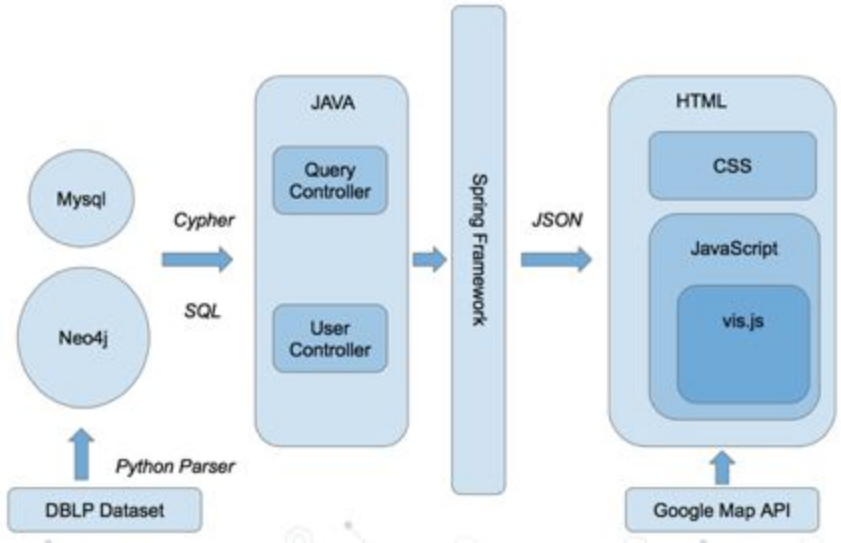
vis.js:

Display dynamic, automatically organised, customizable network views.

System Design

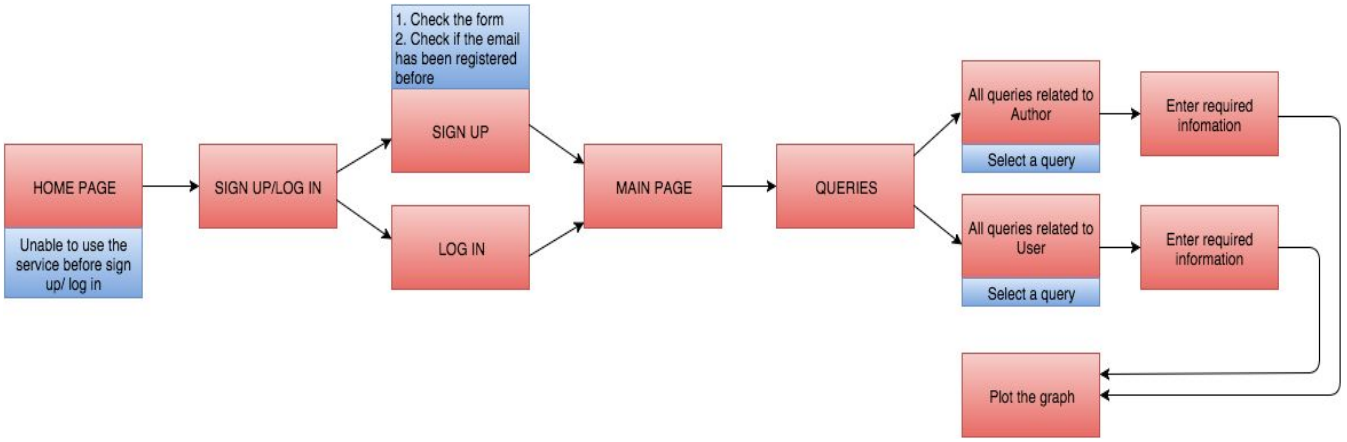
-System Architecture

Multiple technologies are utilized and integrated in the system. From a high-level perspective, raw data is extracted from DBLP Dataset with a python parser, two different kinds of databases are used as data sources, a Java backend serves to transform and compute all queries and a user-friendly front end is provided to fetch all user queries.



-Front-End

Page flow:



According to the requirements of the project. We plan to design a home page first. A home page will contain a “home” button which redirect the page to the home page.

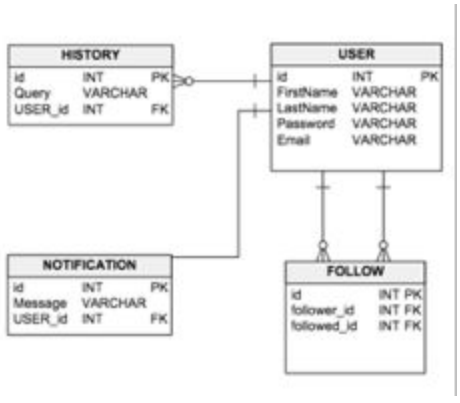
To implement user feature requirement, there should be a sign up/log in button on top-right side of the layout, and after user sign up/log in, the user's email will shown in the same place. Then the second part will be the sign up/log in page. After sign up or log in, the main page will be shown which contains the username, two buttons that will help user to find a specific query. And then, choosing a category of queries like queries related to author and queries related to paper will navigate the user to to another page which contains all the queries in a drop-down list. Choose one of the queries will show the result in a new page which contains the result graph, recommended queries and knowledge card (when user click a node, a knowledge card regarding this node will be shown).

-Back-End

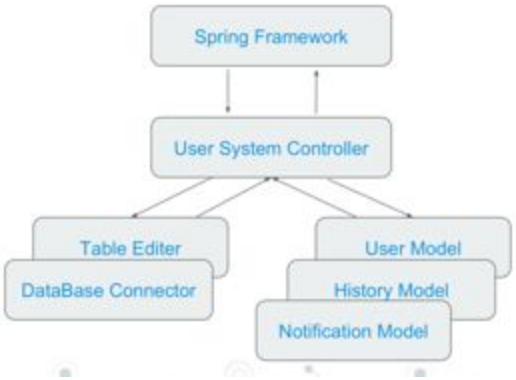
- User Part
- Query Part
- Load Data

User Part

A separated user system is designed for all user-related queries and functionalities. Basically, MySQL database is used to store all user information. Four tables are used to represented User, History, Notification, and follow. The schema is provided as followed:

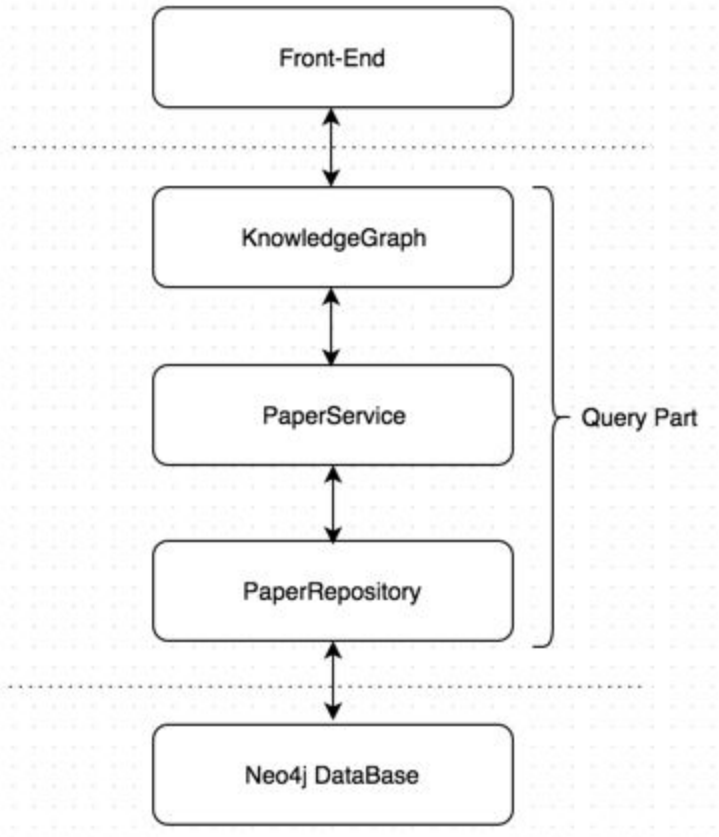


A User Controller is used as the dispatcher to distribute inputs from Spring Framework to different APIs to query from MySQL database.

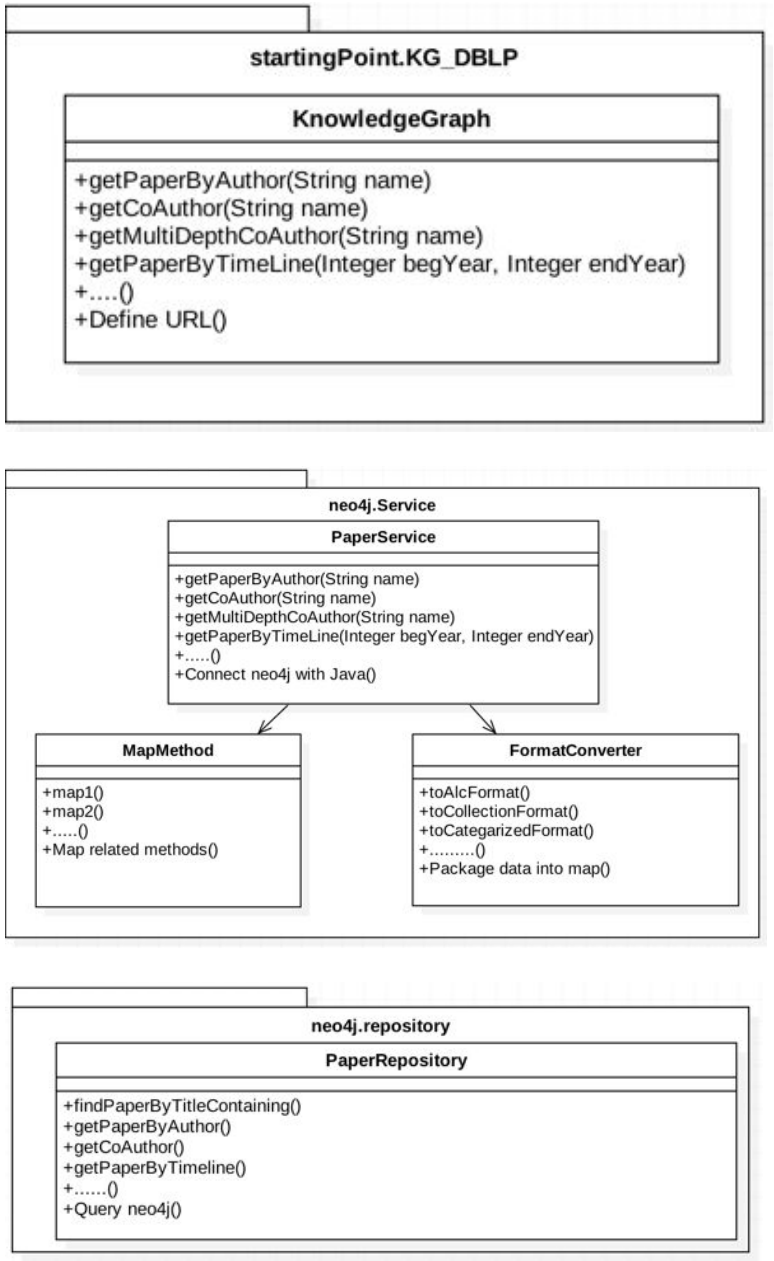


Query Part

We follow the structure that TAs offered. The whole structure is as follows --- three layers structure. The first layer of query part is KnowledgeGraph in which we define the URL that Front-End can call directly. The second layer is PaperService in which we pass the query from KnowledgeGraph to the next layer and package the data returned from the next layer. PaperRepository is the last layer in which we access the Neo4j database.

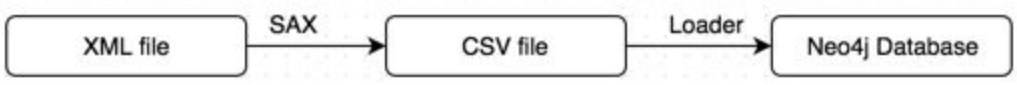


Their corresponding class diagrams are as follows.



Load Data

The load process is as follows. The original database is XML file which we cannot load into Neo4j directly. First step is to parse the XML file using SAX parser into CSV file. The last step is to load CSV file using py2neo python loader into Neo4j database.



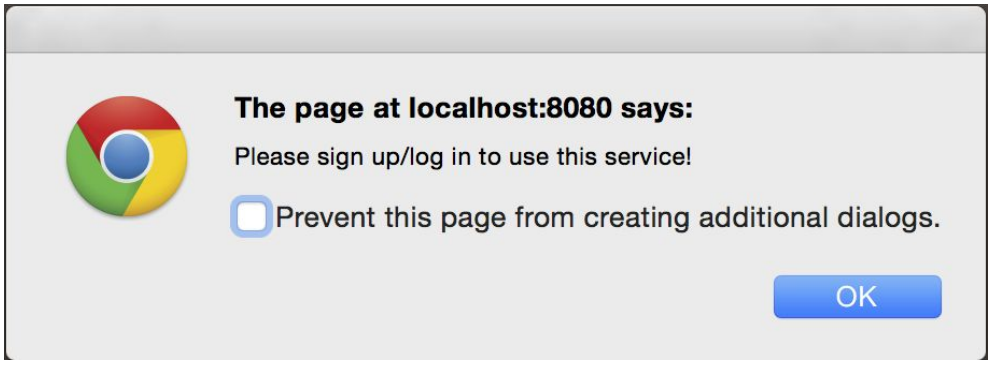
System Implementation

Front-End

- HOME PAGE

For home page, we will implement two functions :
1. For user to sign up or log in.
2. Before signing up/logging in, users can not use this service.

Using javasript, when detected users click on “GET QUERIES OF PAPER” or “GET QUERIES OF AUTHOR”, an alert will be popped up to encourage users to sign up or log in.



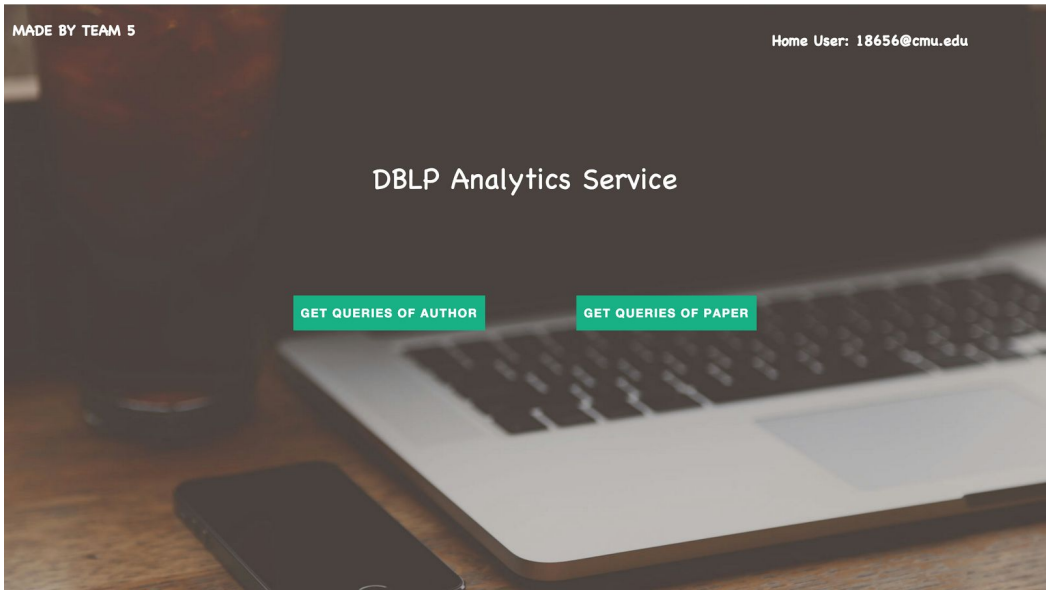
- SIGN UP/LOG IN PAGE

For sign up/log in page, we implement several useful functionalities:

1. Check if all the required information is entered or not.
2. Check if the email format is correct or not. (Email format should be String@String)
3. When signing up/logging in, query the back-end databases and find information about whether the email has been registered before and whether the password matches the email.

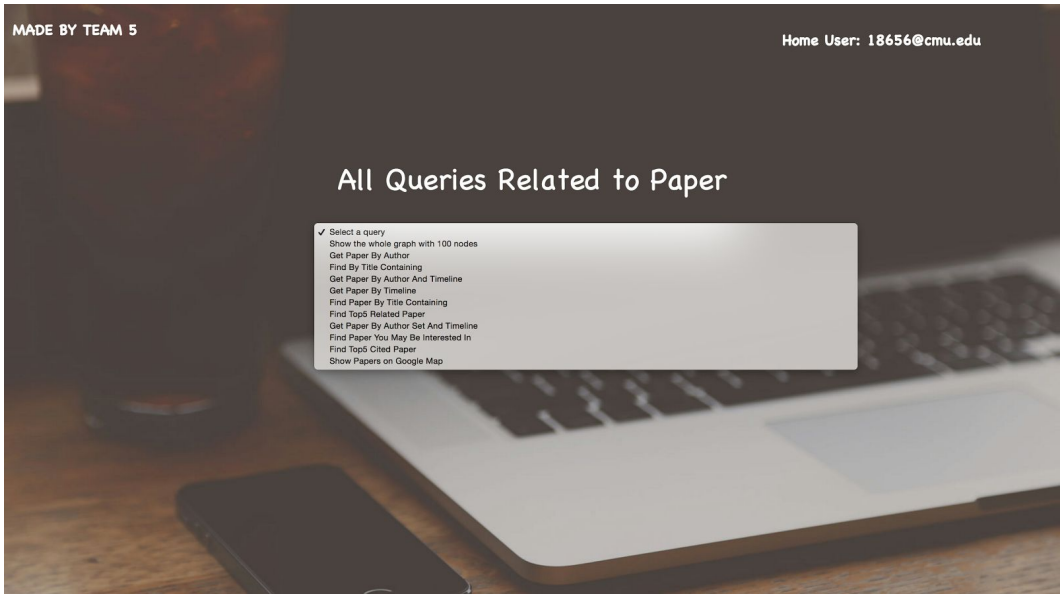
- MAIN PAGE

  Successful signing up or logging in will jump to another page called main page. Main page displays all the information about this user.



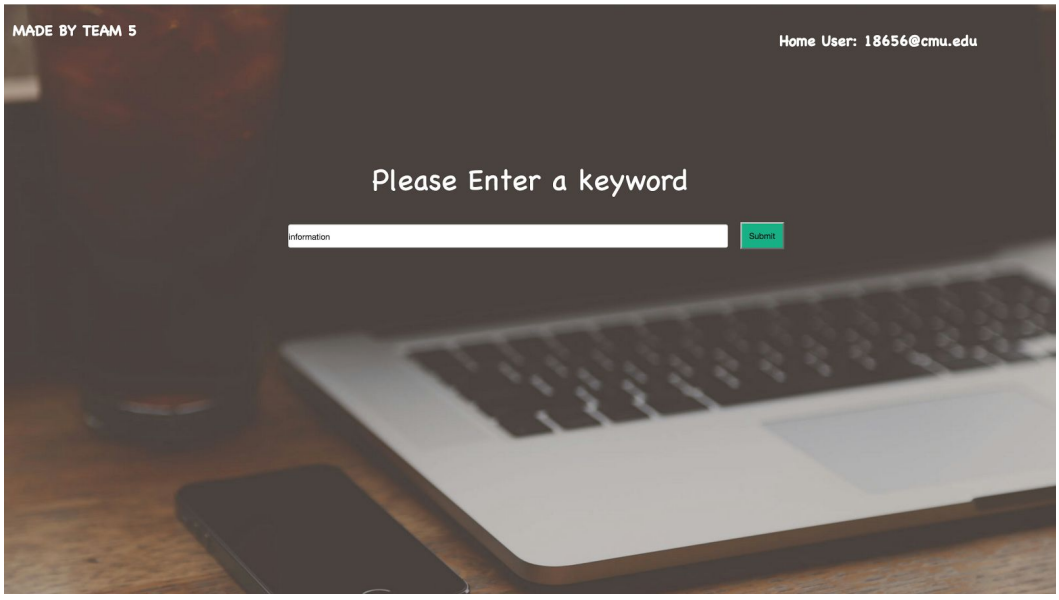
(In this example, the registered user email is 18656@cmu.edu)

- QUERY PAGE
  In the main page, users can choose queries from the two categories.
  For example, choose “GET QUERIES OF PAPER”



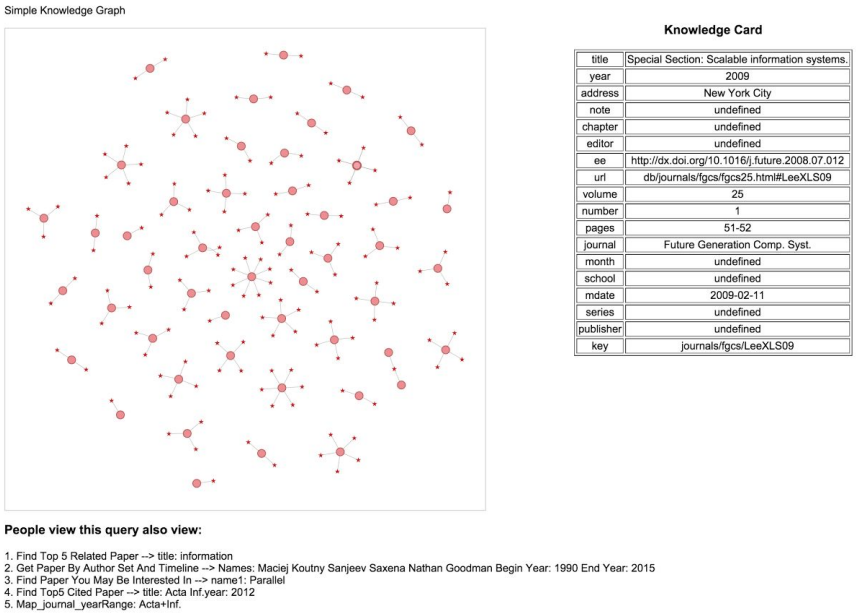
  A drop down list contains all the queries will be shown. Choose a specific one will jump to the input page which is designed for users to enter some information.

- INPUT PAGE



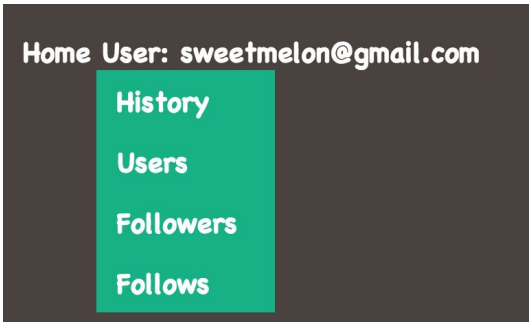
- RESULT PAGE
  The result page contains three part:
  1. The result graph.
  2. Give information about people view this query also view other

queries
3. When click a node, a knowledge graph will be shown on the right side.

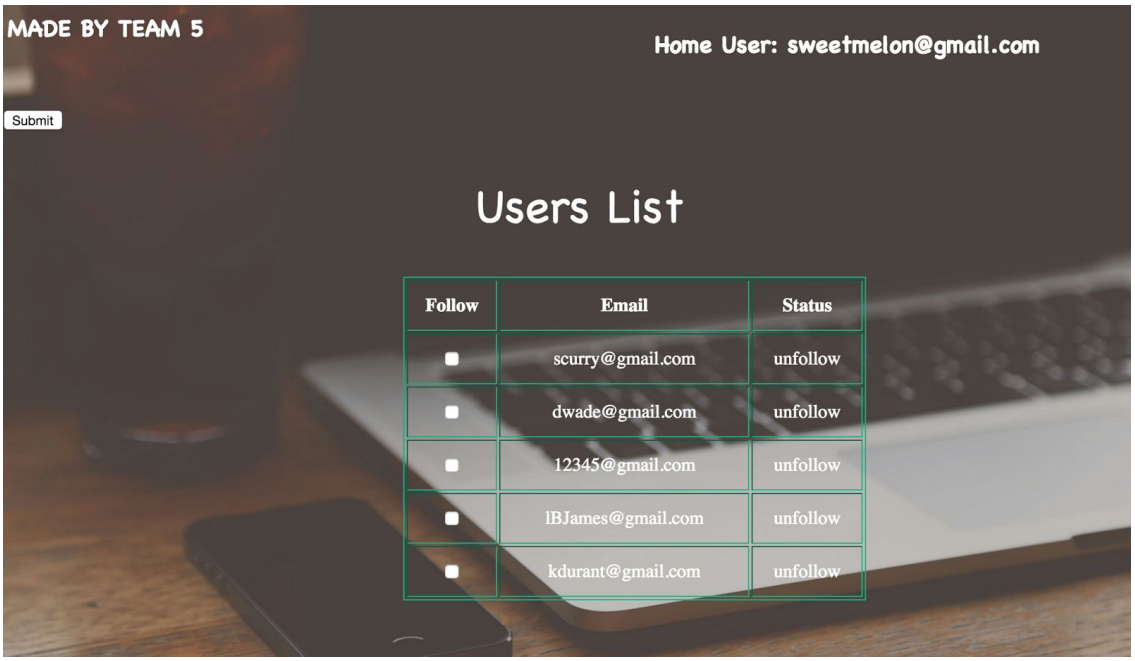


User Part

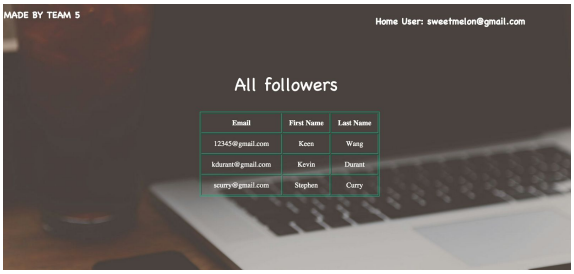
Basically we achieved four functionalities based on user system : Follow others, Get all users/followers/followed, get histories and cache histories for query recommendations.
Basically the functionalities are presented in the pull-down menu in the rightmost corner of the main page:

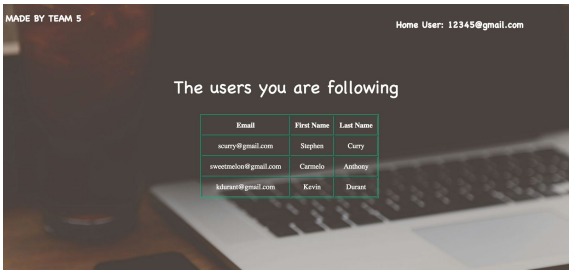


All users in the system can be found in “user” section and any user can be followed by submitting a follow request.

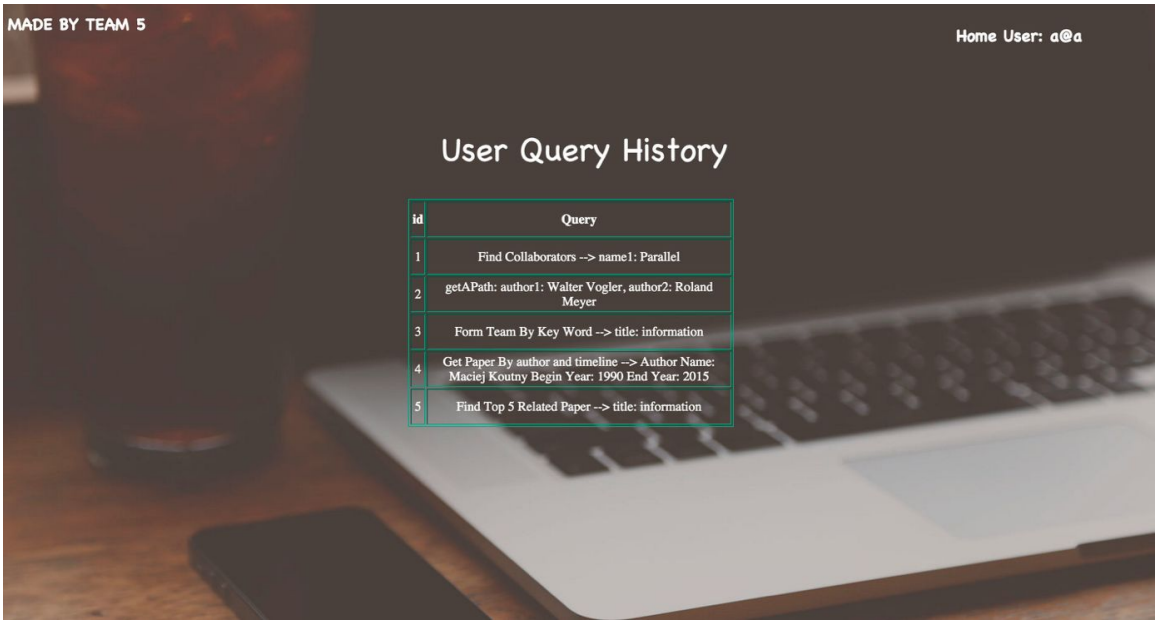


Meanwhile, all follows and followers of current user can be listed:

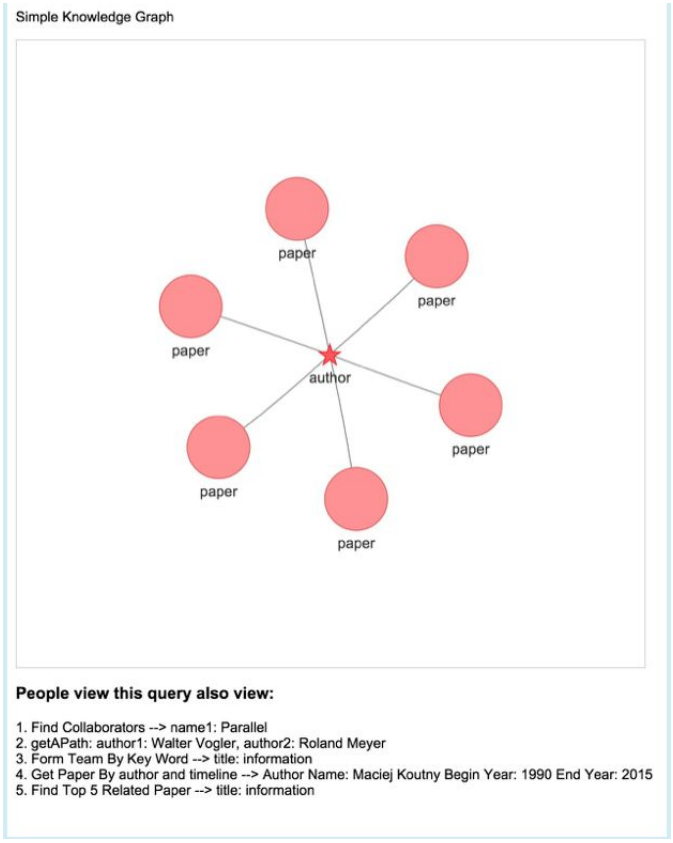




All user's query history is tracked in our system and can be listed as below:



Additionally, a list of top 5 hot queries is cache in memory to recommend to all users. A min heap is used to keep tracking of the top 5 queries of hits. And the list is always presented at the bottom-left corner.



Conclusions and future work

In this project we built a web service for scientists to collaborate and communicate. The completed features include: User authentication and authorization, User profile, Friend and subscription, User groups and the Forum. Possible future work includes: Notification features that support friend/ subscription request & approval.

Contribution of each team member

Bailiang Gong & Na Li : Backend
Baiyang Wang & Peng Tong : Frontend

Tutorial

1. Open the terminal and start Neo4j using “neo4j start”. Then open System Preference and start MySQL.
2. Change to the project directory and build the whole project using “*mvn spring-boot:run -Drun.jvmArguments="-Dusername=neo4j -Dpassword=root*"
3. Open the browser, and type “*http://localhost:8080*”
4. Before signing up or logging in, users can not use this service. So sign up or log in first.
5. After signing up or logging in, users can use the service. Move the mouse to the user email on top-right of the screen, the user can see his/her query history, check the list of all the users, follow a specific user or check the list of all the followers and all the users he/she follows. Also, users can choose either “GET QUERIES OF AUTHOR” or “GET QUERIES OF PAPER”. And then follow the instructions to use a specific query.

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)

-Check in everything onto Docker Hub under the predefined directory

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)