Database Course Project Report

Dingming Zhou (dz1108)

1. Abstract

This report is the documentation of Principles of Database Systems Course project, a website called *Jobster* with relational database system as backend focuses on graduating students looking for jobs. Students and companies can sign up and post information about themselves. Students can also apply for job positions post by companies.

2. Project Requirement

According to the info pdfs provided, the requirement of the whole website can be summarized as:

(a) Student account registration and authentication at login in; students' personal information storage, including names, universities attended, major, GPA, interests, skills, and other info; students' resume upload and storage. Furthermore, students should be able to restrict access to their GPA and resumes so that it is only visible to their friends and to those companies that they have applied to.

(b) Basic social network functionality between students, including a friend system and a message system. One can forward a job post to another.

(c) Company account registration and authentication at login in; companies' profiles including names, location of headquarter and industry info.

(d) Job announcement posting functionality. Companies can post job announcement, providing information including job title, salary, required academic background and details of job description.

(e) Students can receive job announcement by multiple ways, including browsing followed companies public job announcements, browsing job announcements forwarded by friends, and receive notifications or invitations from certain companies which may not followed by students.

(f) Companies could make public job announcements to all following students, or push private or public job announcements by running a filter condition against students' information.

(g) Students can apply to job announcements and check their application status. Companies can see list of applicants and make further actions like checking resumes and checking personal information provided by students.

(h) The basic login and logout functionality, and concurrence for multiple users using the site at same time.

(i) Appropriate measures to guard against SQL injection and cross-site scripting attacks.

3. Database Design

According to the requirements above, my initial ideas about the design is: there is one student table and one company table to store basic user information; one job table to store job announcement info; one or more tables to implement friend functionality (including friend request, etc.); one table to store students follow list; one or more tables to store messages; one table to store special job notification for students; and one table for job application.

To implement this database schema with details, I made a few assumptions about how users are going to use my website, and about the way I implement my design:

(a) All students must have an undergraduate degree at least. One student only need to provide information of (one of) the highest degree acquired, and they only need to provide one GPA along with this degree.
Explanation: In practice, one students can have one or more than one PhD degree and have several master's degrees, along with several GPAs for different degrees. In this project, for convenience, students are required to provide only one GPA and one-degree information.

(b) There is no blacklist functionality provided. One student can only have one friend invitation to another certain student at a time. If one friend invitation gets rejected, the sender can make another request, but only one at a time. Friend relation cannot be removed once the invitation is accepted.
Explanation: In real life, people wants to avoid contacts with certain individuals, and many online social network websites provide blacklist for users to prevent having unwanted interactions with other users. In my design, this functionality is simplified. One student can only make another invitation after the previous invitation to the same student gets rejected.

(c) Companies can select which level of degrees when they post an announcement and can select multiple degrees.
Explanation: In practice, a job position may open to people with different degrees. Hence, the academic bar should be a set.

(d) Messages sent between students are limited to text format. Job announcements forwarded by friends are separated from message.
Explanation: messages are limited to text format because it is easier to implement and organize backend tables and frontend representation; In real life, job forward could be sent as a shared link in messages. In this design I decide to use another table to store job notifications from friends and companies. This will help prevent cross-site attacks as well.

(e) Single student can only apply to one job position once. Company users can still see applicants' resumes even they declined their applications.

(f) Students use job feed functionality on website to see job post by companies they follow (just like Twitter). Special job notification that companies choose to send to selected students are not shown in job feed, but shown in 'Job Notification' column, which are used to show job notifications sent by companies or shared by friend.

(g) With students click to see the pages where represent messages or job notification, the status will change to 'read' from 'unread' even though in reality the user may not see the message on the page.

Then, my database schema is:

Student (sid, sname, password, degree, university, major, gpa, skill, access, resume, rgtime);

Company (cid, password, cname, location, industry, rgtime);

Job (jid, cid, jlocation, title, salary, major, acadamicbar, posttime, posttype, descrp);
FOREIGN KEY cid REFERENCES Company (cid)

Friend (sid, fid);
FOREIGN KEY sid, fid REFERENCES Student(sid)

Invitation (sd, rcv, itime, status);
FOREIGN KEY sd, rcv REFERENCES Student(sid)

Following (sid, cid);
FOREIGN KEY sid REFERENCES Student(sid)
FOREIGN KEY cid REFERENCES Company(cid)

Message (sd, rcv, mtime, content, mstatus)
FOREIGN KEY sd, rcv REFERENCES Student(sid)

JobNotification (jid, sid, mtime, source, status)
FOREIGN KEY jid REFERENCES Job(jid)
FOREIGN KEY sid REFERENCES Student(sid)

Application (jid, sid, apptime, status)
FOREIGN KEY jid REFERENCES Job (jid)
FOREIGN KEY sid REFERENCES Student(sid)

Some explanation about this design:

(a) For job table, the posttype attribute has two value, 'Public' and 'Partial'. If the company set the job post to 'Public', this job post will automatically appear on the company's followers' feed. If set to 'Partial', the company need to choose which students are going to receive this job announcement. Of course, a 'Public' job announcement could also be chosen by company to be forwarded to specific students. This table is for job announcement info and be used to generate students' feed with Following table.

(b) In Friend table, I would like double tuples for single friend relationship. For example, if A is a friend of B, then there will two tuples, (A,B) and (B,A), in the database. The reason to have two tuples is that it is easier to construct queries at frontend and make queries Processing faster despite that it has data redundancy.
After one accepts friend invitation from another, two rows with their id info will insert into Friend table, and the original status will change from 'Awaiting' to 'Accepted'. If rejected, no rows will be inserted in to friend table, and the status will be changed to 'Declined'.
(c) Message table is used to store text message between students. Logically, each message is identical by its sender, receiver and the timestamp of sending.
(d) JobNotification table is used to store the job announcements forwarded by friends or pushed by companies. Furthermore, there is an attribute 'mstatus' to represent whether the notification has been seen by receiver. If the receiver clicked the notification link, the status will change from 'unread' to 'read'. This table helps students get latest notification about job announcements.
(e) Application table is to store student application for jobs. With this table, companies can see students that applied and their resumes using joins in backend. Students can see which jobs they have applied and the status of application.

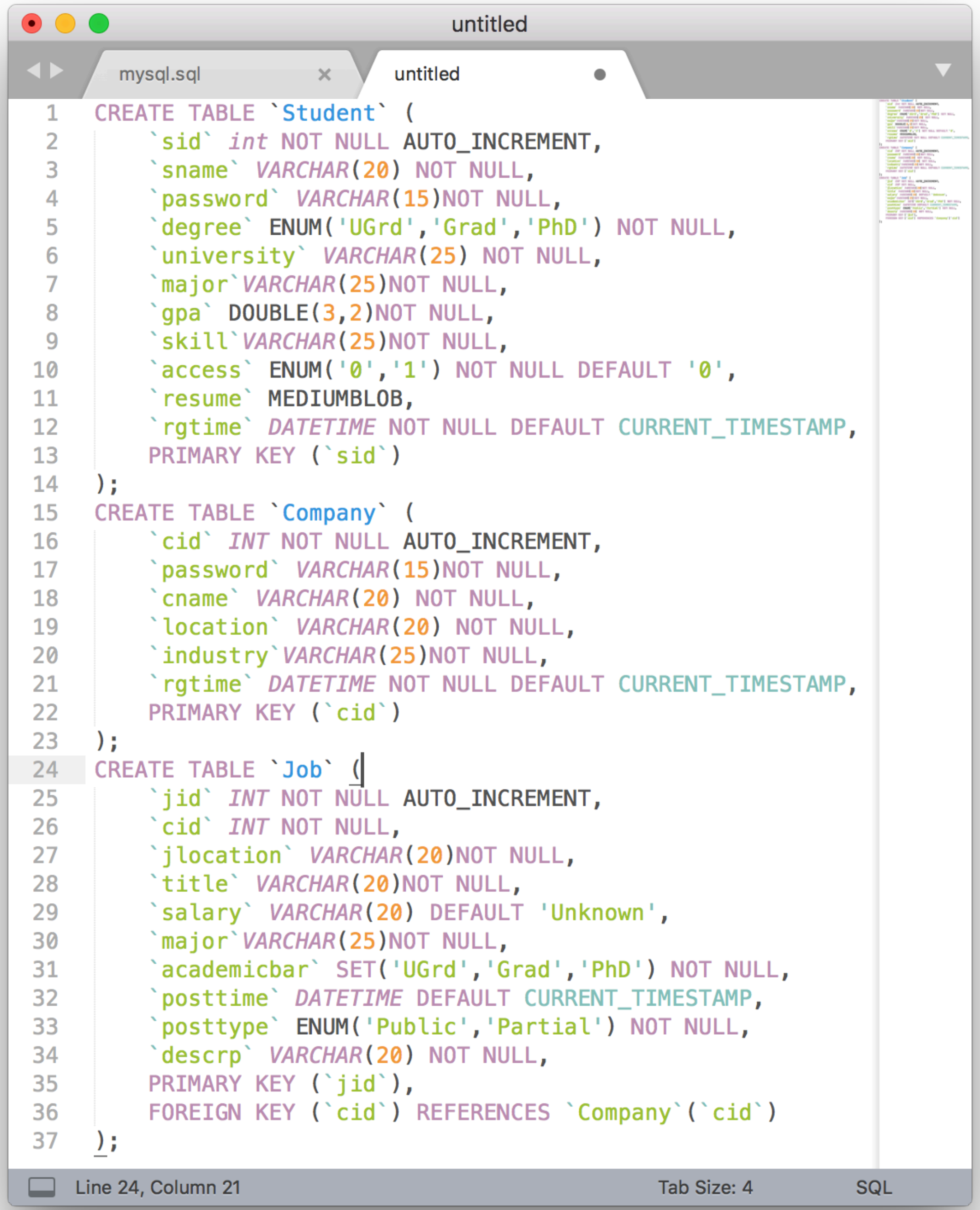
Then, I need to address some decisions I made about some attributes' type and default value:

(a) In Student table, password is stored without encryption, since this is a simple login system. Degree is an enum type since I only need to store one degree that user provide. Access is the attribute that flag whether a user want to show his/her GPA and resume to public, and '0' means only friends and companies that he/she had applied before can see his/her GPA and resume while '1' means everyone can access them on user's profile page. 'rgtime' attribute is used to store the timestamp of user sign up, and is used for user login id retrieving after user sign up: after insertion on Student table, I use a select query to get this new user's login id by finding the tuple with latest rgtime value. Technically speaking, there stands a chance that if two users created two accounts at same time, there would be mess providing correct login user id, but considering this chance is so small that we can assume that every user created account at different time.
(b) In Company table, password is stored without encryption and 'rgtime' attribute is used for record and providing login id for new users due to same reason above.
(c) In Job table, 'posttype' attribute is used for company users to decide whether certain job post could be seen in students' job feed. Sometimes companies want to offer positions to selected students without showing to public, they can set it to 'Partial' when creating job post so it will not appear in followers' job feed. For students, even though they may not be selected to receive certain jobs' notification, they still are able to see job post details with 'Partial' type because these jobs could be forwarded by selected students. I assume that companies are open on this subject because selected students may have friends that are better applicants.

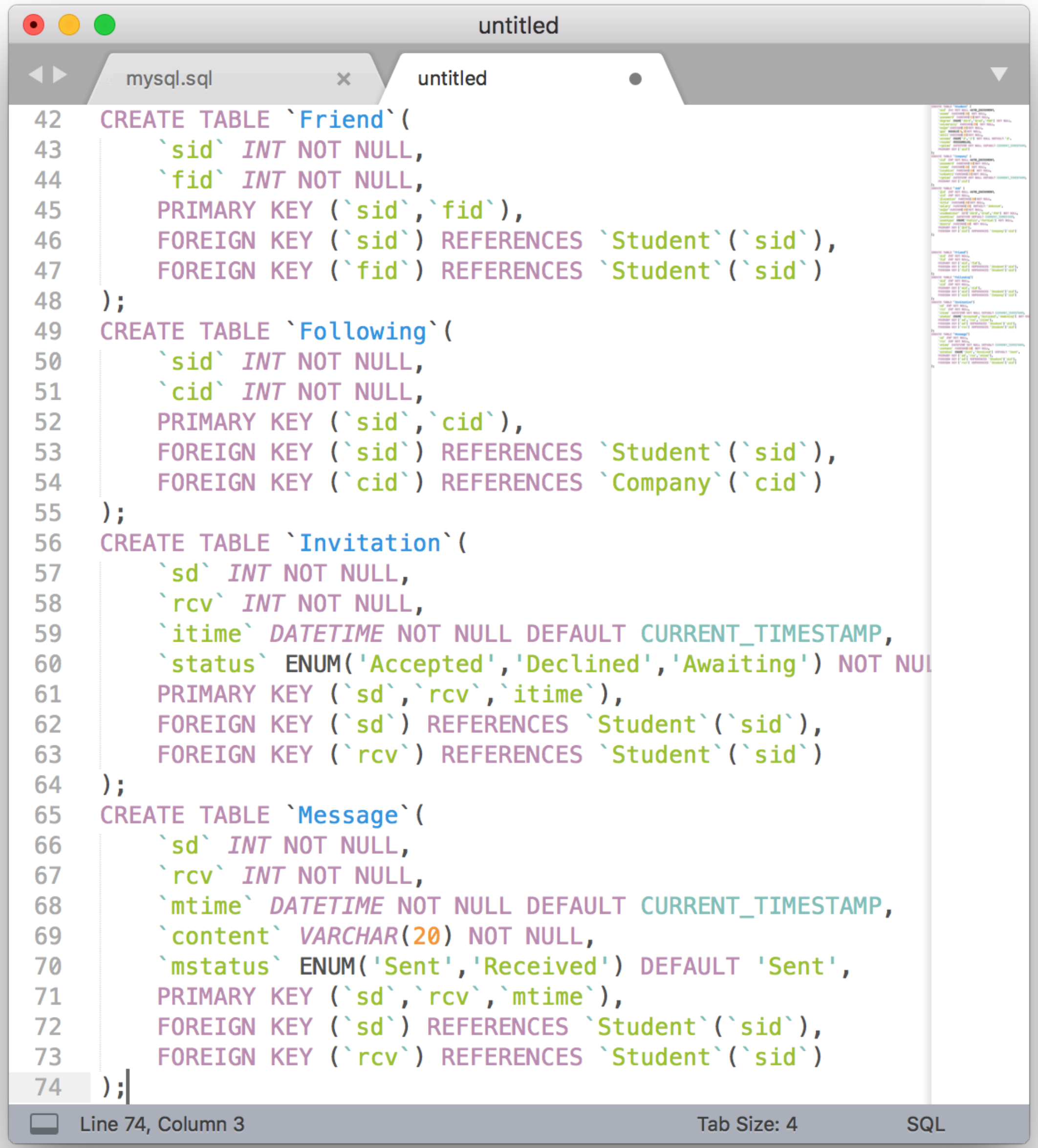
(d) In Invitation table, \`itime\` which represents when the invitation was sent should be part of the primary because one user can send invitations to same user several times (rejected and send again).

(e) In JobNotification table, ‘mtime’ should be part of primary key because one job post could be forwarded between two students and between companies and students. ‘Source’ is an enum type attribute because the notification sender could only be either company or student.

Then, the corresponding MySQL schema is:



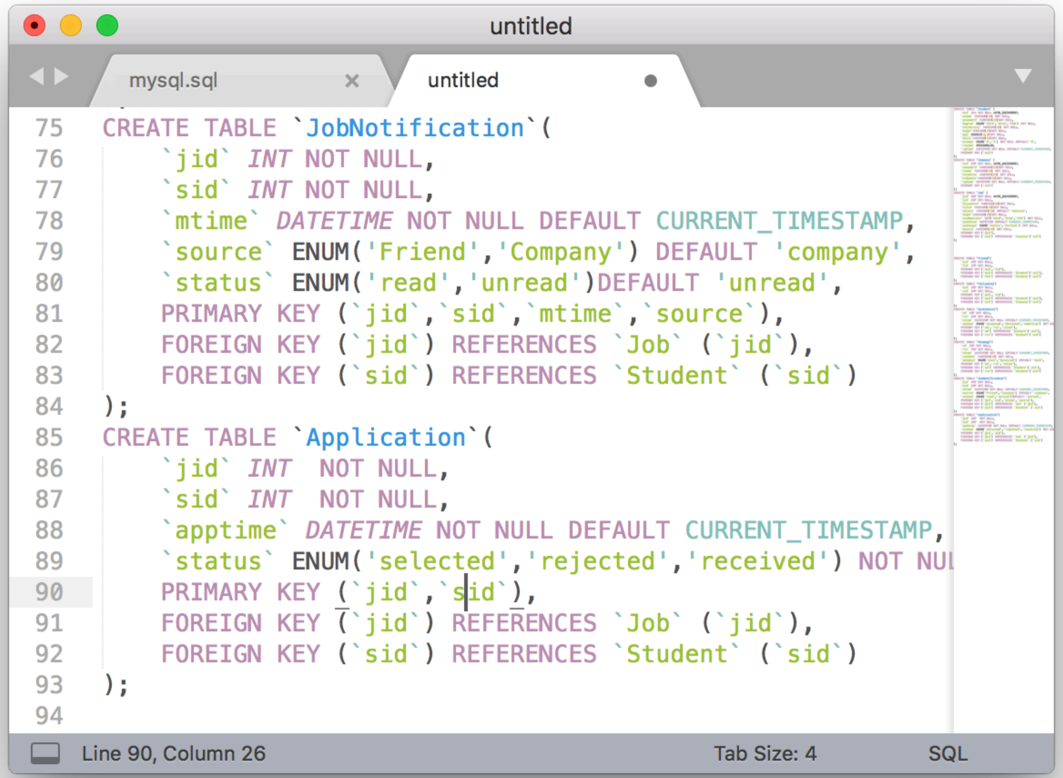
```sql
CREATE TABLE `Student` (
    `sid` int NOT NULL AUTO_INCREMENT,
    `sname` VARCHAR(20) NOT NULL,
    `password` VARCHAR(15)NOT NULL,
    `degree` ENUM('UGrd','Grad','PhD') NOT NULL,
    `university` VARCHAR(25) NOT NULL,
    `major`VARCHAR(25)NOT NULL,
    `gpa` DOUBLE(3,2)NOT NULL,
    `skill`VARCHAR(25)NOT NULL,
    `access` ENUM('0','1') NOT NULL DEFAULT '0',
    `resume` MEDIUMBLOB,
    `rgtime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`sid`)
);
CREATE TABLE `Company` (
    `cid` INT NOT NULL AUTO_INCREMENT,
    `password` VARCHAR(15)NOT NULL,
    `cname` VARCHAR(20) NOT NULL,
    `location` VARCHAR(20) NOT NULL,
    `industry`VARCHAR(25)NOT NULL,
    `rgtime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`cid`)
);
CREATE TABLE `Job` (
    `jid` INT NOT NULL AUTO_INCREMENT,
    `cid` INT NOT NULL,
    `jlocation` VARCHAR(20)NOT NULL,
    `title` VARCHAR(20)NOT NULL,
    `salary` VARCHAR(20) DEFAULT 'Unknown',
    `major`VARCHAR(25)NOT NULL,
    `academicbar` SET('UGrd','Grad','PhD') NOT NULL,
    `posttime` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `posttype` ENUM('Public','Partial') NOT NULL,
    `descrp` VARCHAR(20) NOT NULL,
    PRIMARY KEY (`jid`),
    FOREIGN KEY (`cid`) REFERENCES `Company`(`cid`)
);
```


untitled

mysql.sql | untitled

```
CREATE TABLE `Friend`(
    `sid` INT NOT NULL,
    `fid` INT NOT NULL,
    PRIMARY KEY (`sid`,`fid`),
    FOREIGN KEY (`sid`) REFERENCES `Student`(`sid`),
    FOREIGN KEY (`fid`) REFERENCES `Student`(`sid`)
);
CREATE TABLE `Following`(
    `sid` INT NOT NULL,
    `cid` INT NOT NULL,
    PRIMARY KEY (`sid`,`cid`),
    FOREIGN KEY (`sid`) REFERENCES `Student`(`sid`),
    FOREIGN KEY (`cid`) REFERENCES `Company`(`cid`)
);
CREATE TABLE `Invitation`(
    `sd` INT NOT NULL,
    `rcv` INT NOT NULL,
    `itime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `status` ENUM('Accepted','Declined','Awaiting') NOT NUL
    PRIMARY KEY (`sd`,`rcv`,`itime`),
    FOREIGN KEY (`sd`) REFERENCES `Student`(`sid`),
    FOREIGN KEY (`rcv`) REFERENCES `Student`(`sid`)
);
CREATE TABLE `Message`(
    `sd` INT NOT NULL,
    `rcv` INT NOT NULL,
    `mtime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `content` VARCHAR(20) NOT NULL,
    `mstatus` ENUM('Sent','Received') DEFAULT 'Sent',
    PRIMARY KEY (`sd`,`rcv`,`mtime`),
    FOREIGN KEY (`sd`) REFERENCES `Student`(`sid`),
    FOREIGN KEY (`rcv`) REFERENCES `Student`(`sid`)
);
```

Line 74, Column 3 | Tab Size: 4 | SQL



```sql
CREATE TABLE `JobNotification`(
    `jid` INT NOT NULL,
    `sid` INT NOT NULL,
    `mtime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `source` ENUM('Friend','Company') DEFAULT 'company',
    `status` ENUM('read','unread')DEFAULT 'unread',
    PRIMARY KEY (`jid`,`sid`,`mtime`,`source`),
    FOREIGN KEY (`jid`) REFERENCES `Job` (`jid`),
    FOREIGN KEY (`sid`) REFERENCES `Student` (`sid`)
);
CREATE TABLE `Application`(
    `jid` INT  NOT NULL,
    `sid` INT  NOT NULL,
    `apptime` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `status` ENUM('selected','rejected','received') NOT NUL
    PRIMARY KEY (`jid`,`sid`),
    FOREIGN KEY (`jid`) REFERENCES `Job` (`jid`),
    FOREIGN KEY (`sid`) REFERENCES `Student` (`sid`)
);
```

4. Frontend Design

The server code has been uploaded to GitHub (link)

(a) Website structure

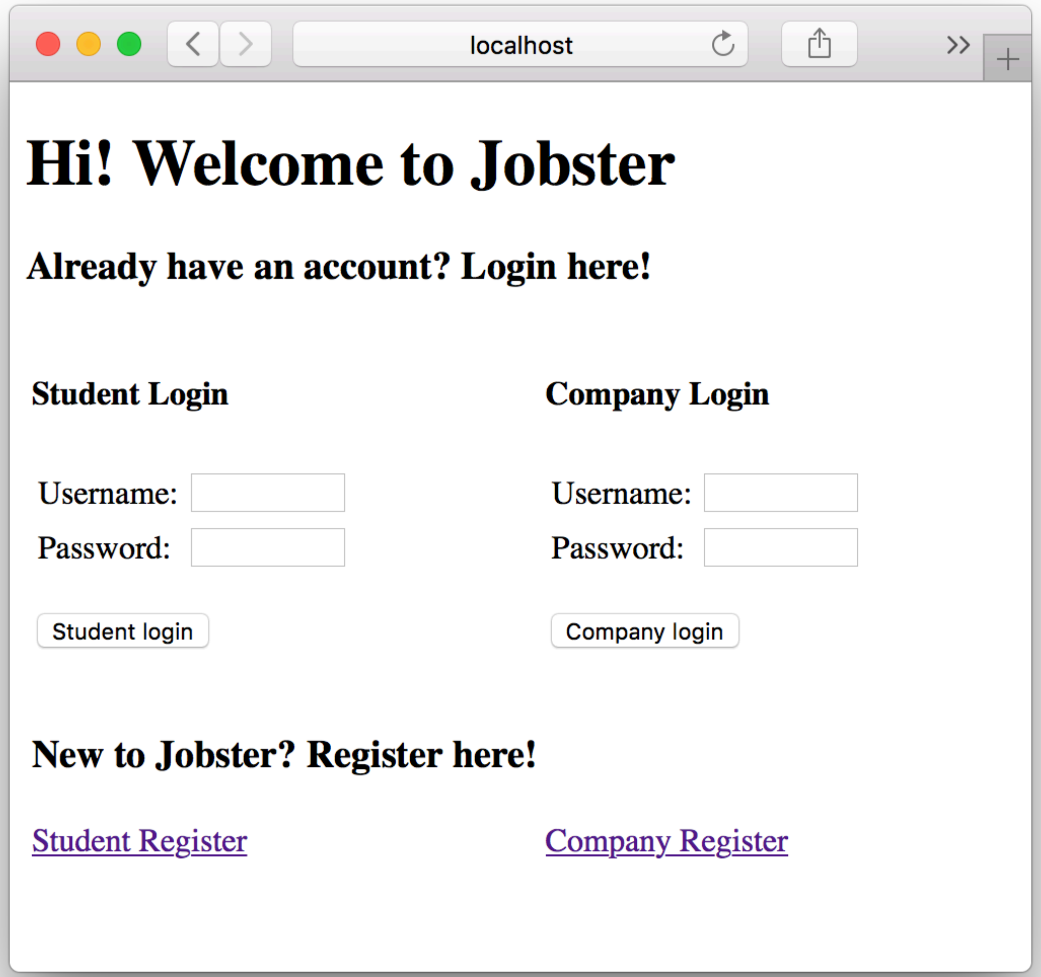
The website will provide service for both companies and students, each with different services. Thus, some pages are only available for students, some are only for companies, some are for both. In my frontend site structure design,

(1) Public user login and registration pages: login.html, logincheck.php, StuReg.html, StuReg.php, ComReg.html, ComReg.php, logout.php.
(2) Students-only pages: StuHome.php, friend.php, message.php, MesHis.php, JobNot.php, JobFeed.php.
(3) Company-only pages: ComHome.php, newjob.php.
(4) Common pages: people.php, job.php, company.php.
(5) Test and backend function script: mysql.sql, function.inc.

With different user type, the pages that user can access are different. In common pages that both type of user can access, the page is different when providing services.

(b) Page Design and Query Optimization

Login.html
This static page is used for login and registration guide.



Hi! Welcome to Jobster

Already have an account? Login here!

Student Login

Username:
Password:

Student login

Company Login

Username:
Password:

Company login

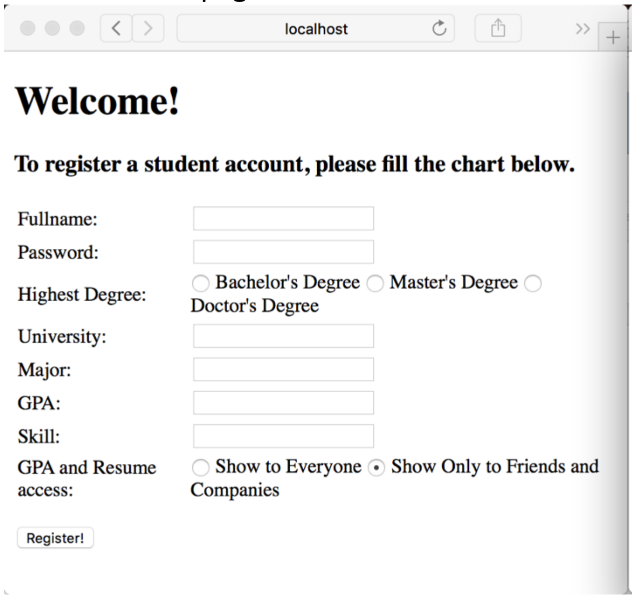
New to Jobster? Register here!

Student Register

Company Register

StuReg.html and ComReg.html

Both are static pages used for user information collection.



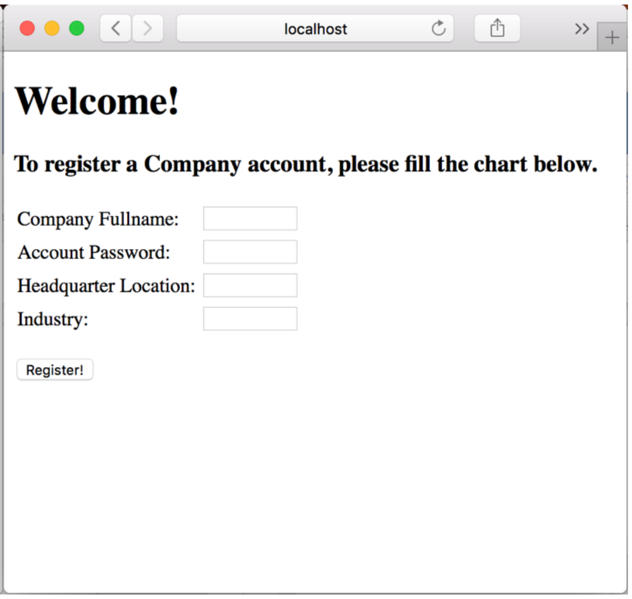
localhost

Welcome!

To register a student account, please fill the chart below.

Fullname:	
Password:	
Highest Degree:	○ Bachelor's Degree ○ Master's Degree ○ Doctor's Degree
University:	
Major:	
GPA:	
Skill:	
GPA and Resume access:	○ Show to Everyone [✓ Show Only to Friends and Companies]

Register!



localhost

Welcome!

To register a Company account, please fill the chart below.

Company Fullname:	
Account Password:	
Headquarter Location:	
Industry:	

Register!

StuReg.php and ComReg.php

Both scripts are for backend processing and will not generate any pages. Once a user successfully creates an account, it will redirect new user to home page, which is generated by StuHome.php and ComHome.php respectively.

The SQL queries used are:

INSERT INTO `STUDENT` (`sname`,`password`,`degree`,`university`,`major`,`gpa`,`skill`,`access`) VALUES (…);

SELECT * from student order by rgtime DESC limit 1;

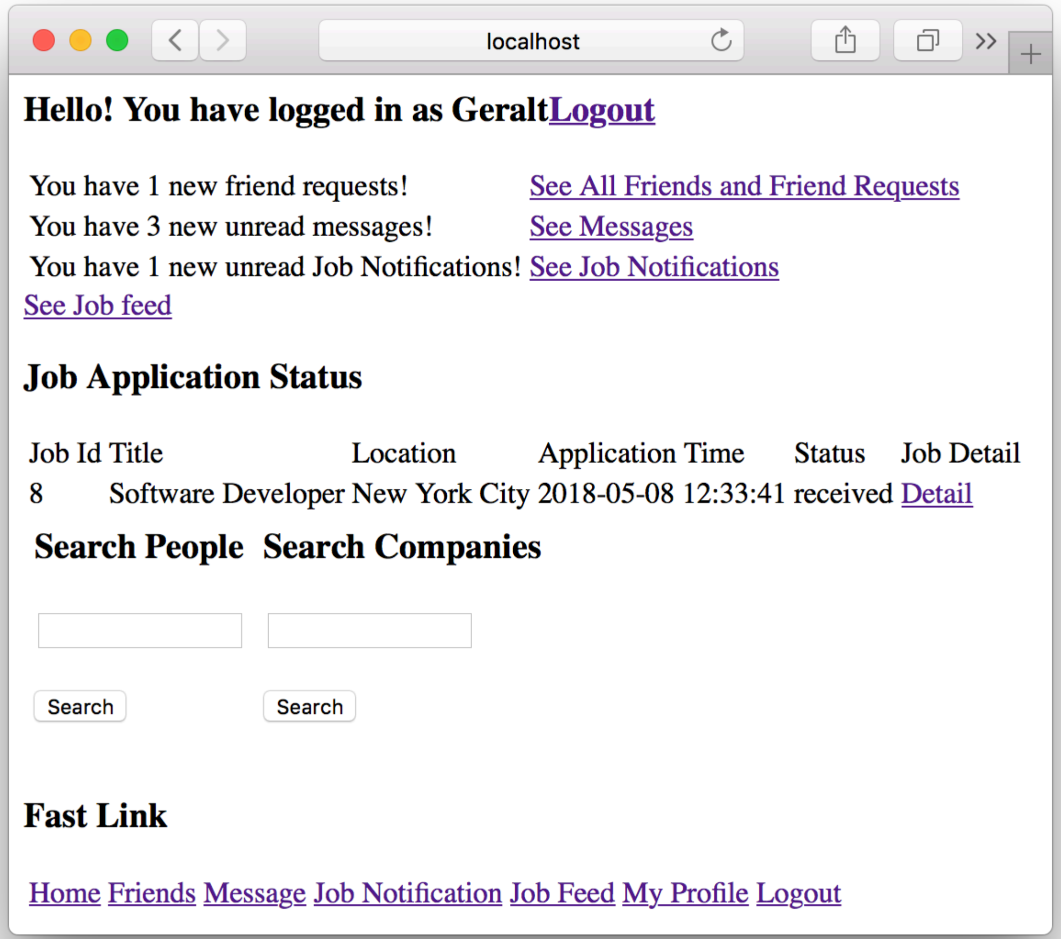
INSERT INTO `Company`(`cname`,`password`,`location`,`industry`) VALUES (…);

SELECT * from company order by rgtime DESC limit 1;

The second and forth queries are used to retrieve newly registered user's id.

StuHome.php

This is the student's home page to access services. First 3 lines in main content provide notifications after last login or StuHome.php page reload. Then the 'See Job feed' button will guide user to JobFeed.php for job feed. Below that is the 'Job Application Status' area which shows recent job application status. I think this functionality is important so I put it in the first page. After that is search area where students can type in search keyword to search either people or companies. The search button will guide them to people.php and company.php respectively. The last part is 'Fast Link' for users to quickly jump between pages.



Hello! You have logged in as GeraltLogout

You have 1 new friend requests!	See All Friends and Friend Requests
You have 3 new unread messages!	See Messages
You have 1 new unread Job Notifications!	See Job Notifications

See Job feed

Job Application Status

Job Id	Title	Location	Application Time	Status	Job Detail
8	Software Developer	New York City	2018-05-08 12:33:41	received	Detail

Search People Search Companies

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout

The SQL queries used in this page are:

SELECT sname,sid from Invitation natural join student where rcv =(…) and Invitation.sd = student.sid and status='Awaiting';

SELECT * from Message where rcv=(…) and mstatus = 'Sent';

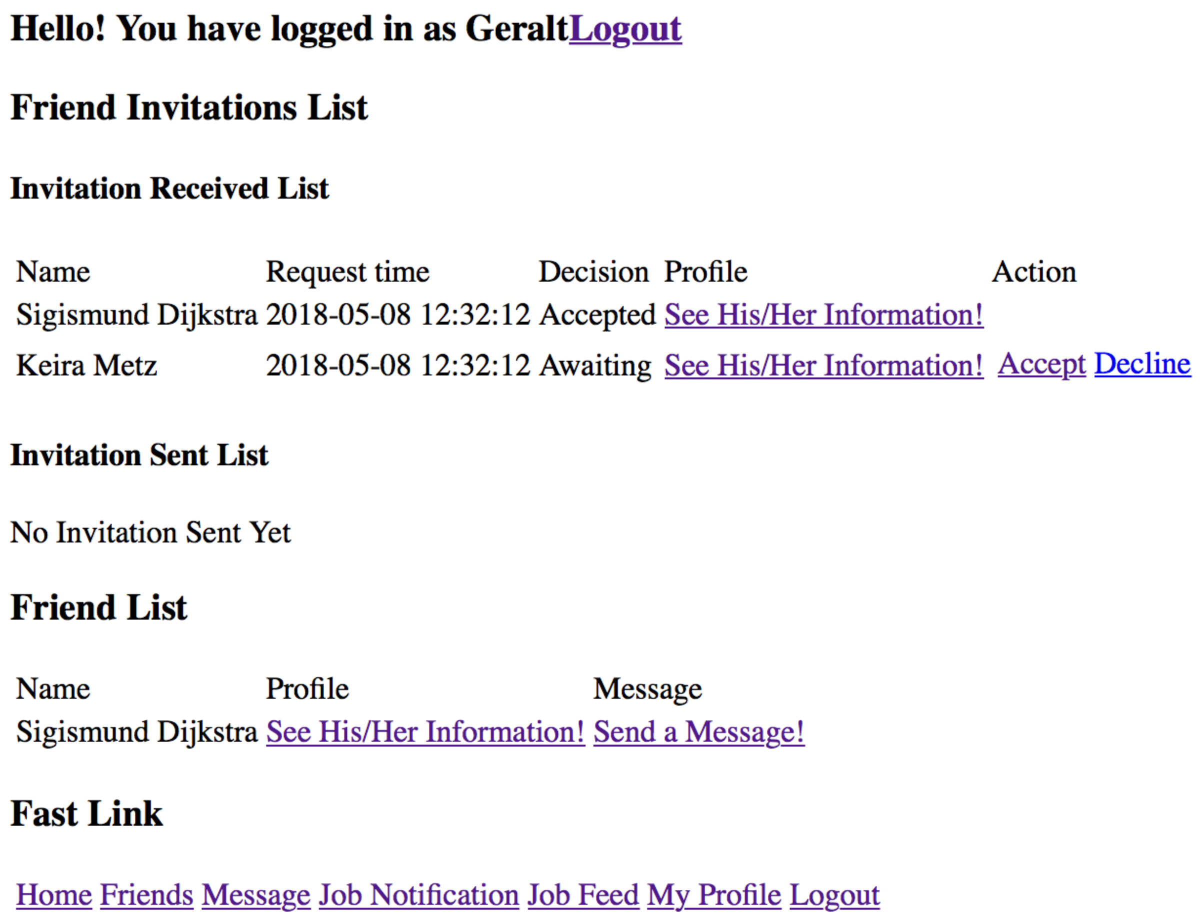
SELECT * from JobNotification where sid = (…) and status = 'unread';

SELECT * from Application natural join job where sid=(…) order by apptime DESC;

These queries are used for collecting general notifications. I put the user-id-check clause infront of other check clause because ideally, the number of tuples that are uniquely identified by sid is less than tuples identified by status attribute in these tables. Thus, the query will be processed faster.

friend.php

This page is where students manage friend invitation and check friend list.



Hello! You have logged in as GeraltLogout

Friend Invitations List

Invitation Received List

Name	Request time	Decision	Profile	Action
Sigismund Dijkstra	2018-05-08 12:32:12	Accepted	See His/Her Information!	
Keira Metz	2018-05-08 12:32:12	Awaiting	See His/Her Information!	Accept Decline

Invitation Sent List

No Invitation Sent Yet

Friend List

Name	Profile	Message
Sigismund Dijkstra	See His/Her Information!	Send a Message!

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout

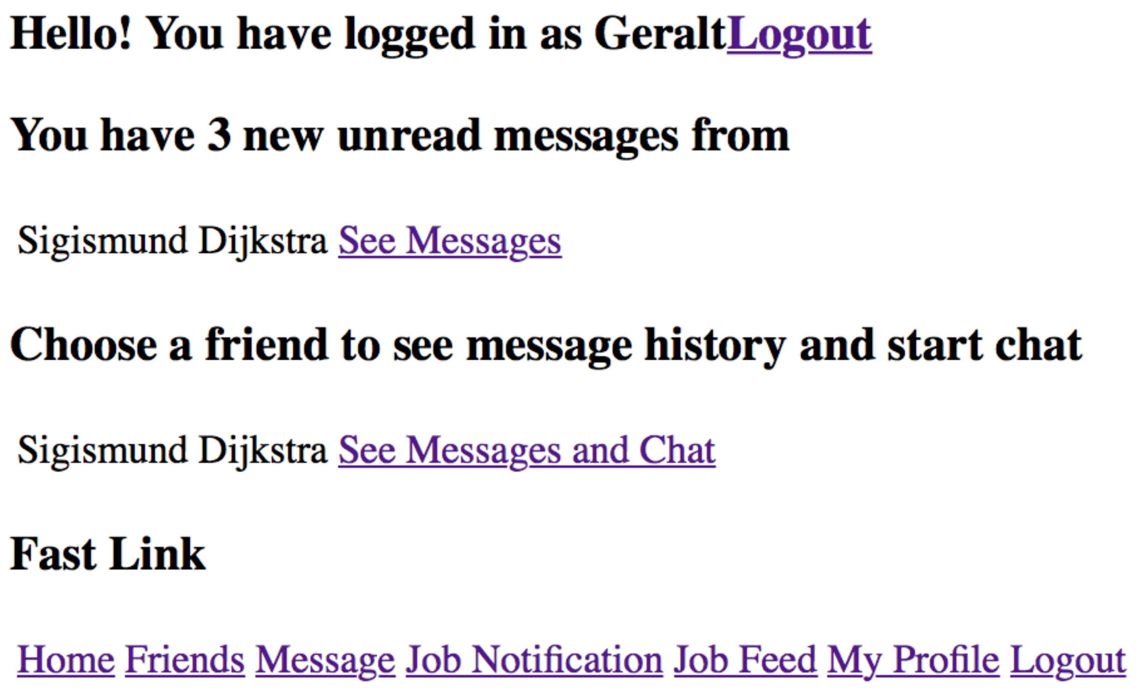
The SQL queries used in this page are:

select * from Invitation where sd = (…) and rcv = (…) and status = 'Awaiting';
update Invitation set status = 'Accepted' where sd = (…) and rcv = (…) and status = 'Awaiting';
update Invitation set status = 'Declined' where sd = (…) and rcv = (…) and status = 'Awaiting';
insert into friend values (…,….);
select sname,sid,itime,status from Invitation natural join student where rcv =(…) and Invitation.sd = student.sid order by itime DESC;
select sname,sid,itime,status from Invitation natural join student where sd =(….) and Invitation.rcv = student.sid order by itime DESC;
select sname,sid from friend natural join student where fid = (…) and friend.sid = student.sid order by sname;

In these queries, the first three can be optimized because sd,rcv attributes are much more sparse than status attribute. Thus, I put sd=(…) and rcv=(…) first in where clause. The rest queries do not have much space for optimization because they are combinations of one join and one selection operation.

Message.php

This page is where student choose a friend to see message history with him/her.



Hello! You have logged in as GeraltLogout

You have 3 new unread messages from

Sigismund Dijkstra See Messages

Choose a friend to see message history and start chat

Sigismund Dijkstra See Messages and Chat

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout

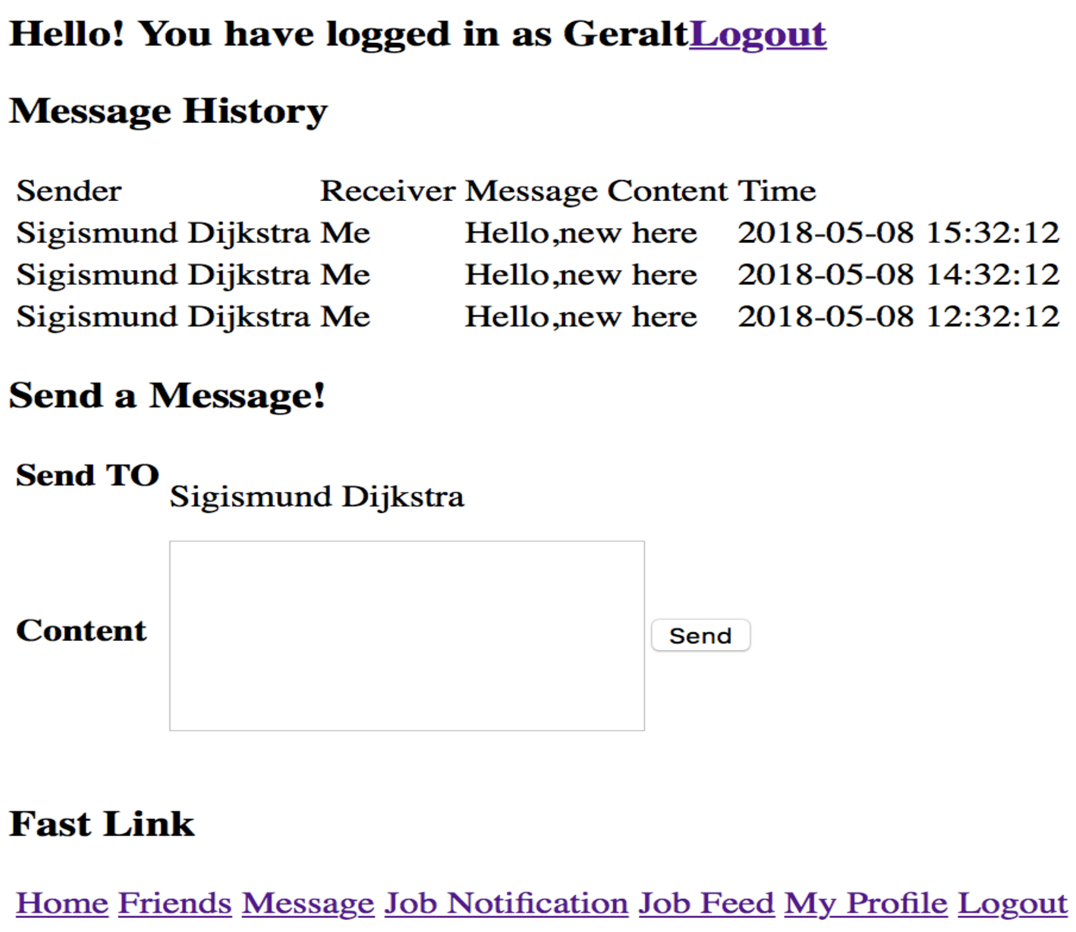
The queries used are:

select * from Message where rcv = (…) and mstatus = 'Sent';
select distinct sname,sid from Message natural join Student where rcv = (…) and mstatus = 'Sent' and Message.sd = Student.sid;
select distinct sname,sid from friend natural join Student where fid = (…)

These queries are optimized due to similar reasons like before.

MesHis.php

This page is to show message history with another user. I assume that when a student reach this page, all message will be set to ‘read’ instead of ‘unread’, even though in reality the user may not see the messages on this page.



Hello! You have logged in as Geralt Logout

Message History

Sender	Receiver	Message Content	Time
Sigismund Dijkstra	Me	Hello,new here	2018-05-08 15:32:12
Sigismund Dijkstra	Me	Hello,new here	2018-05-08 14:32:12
Sigismund Dijkstra	Me	Hello,new here	2018-05-08 12:32:12

Send a Message!

Send TO Sigismund Dijkstra

Content

Fast Link

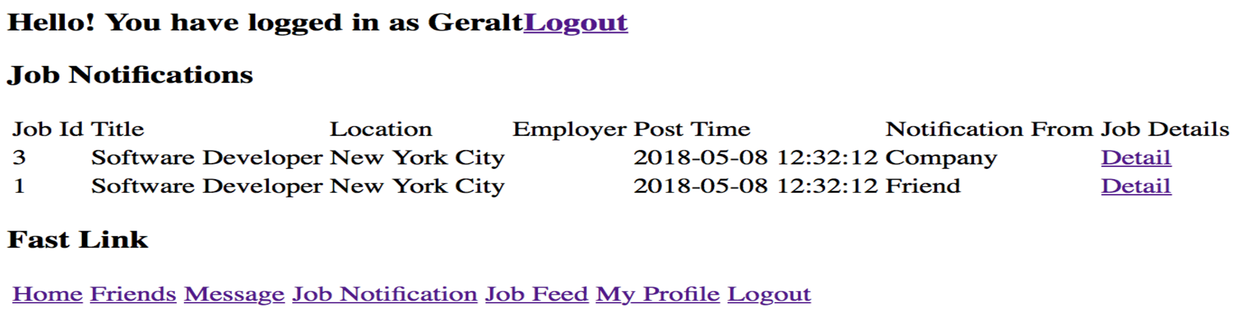
Home Friends Message Job Notification Job Feed My Profile Logout

The SQL queries are:

INSERT INTO `Message` (`sd`,`rcv`,`content`,`mstatus`) VALUES(…,…,…,'Sent');
select * from Message where (sd = … and rcv = …) or (sd = … and rcv = …) order by mtime DESC";
update message set mstatus = 'Received' where (sd = … and rcv = …);

JobNot.php

This page to represent the job notification received from companies and friends.



Hello! You have logged in as Geralt Logout

Job Notifications

Job Id	Title	Location	Employer	Post Time	Notification From	Job Details
3	Software Developer	New York City		2018-05-08 12:32:12	Company	Detail
1	Software Developer	New York City		2018-05-08 12:32:12	Friend	Detail

Fast Link

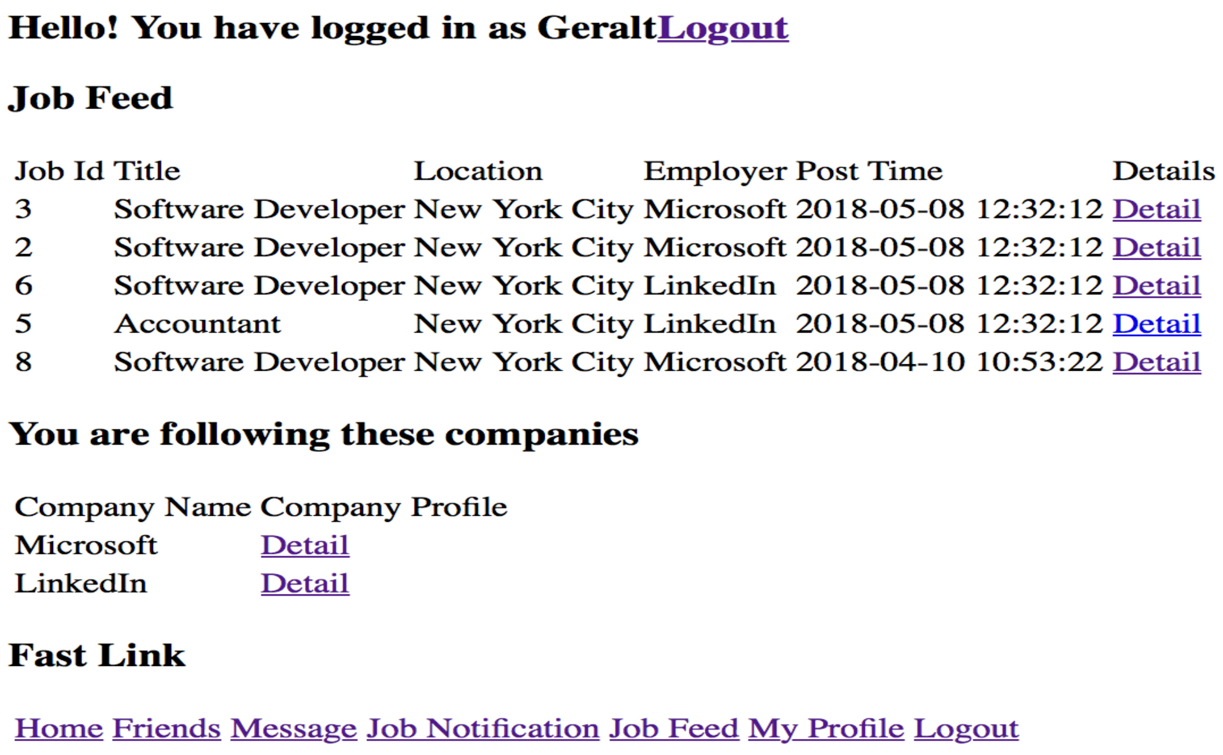
Home Friends Message Job Notification Job Feed My Profile Logout

The queries are:

update JobNotification set status='read' where sid =(…);
select * from JobNotification natural join job where sid= (…) order by mtime DESC;

They are common queries and do not require optimization.

JobFeed.php
This page is to represent job feed for student.



Hello! You have logged in as GeraltLogout

Job Feed

Job Id	Title	Location	Employer	Post Time	Details
3	Software Developer	New York City	Microsoft	2018-05-08 12:32:12	Detail
2	Software Developer	New York City	Microsoft	2018-05-08 12:32:12	Detail
6	Software Developer	New York City	LinkedIn	2018-05-08 12:32:12	Detail
5	Accountant	New York City	LinkedIn	2018-05-08 12:32:12	Detail
8	Software Developer	New York City	Microsoft	2018-04-10 10:53:22	Detail

You are following these companies

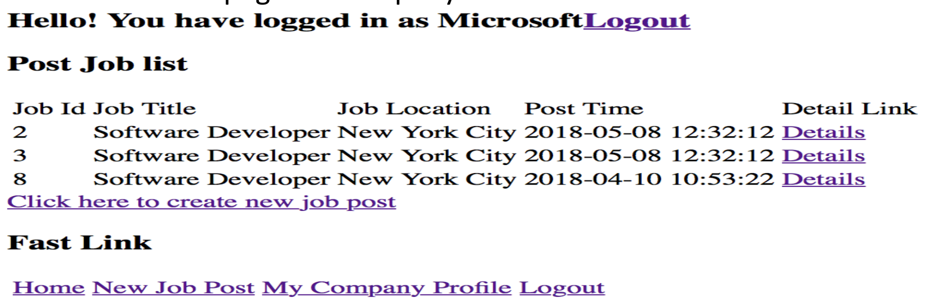
Company Name	Company Profile
Microsoft	Detail
LinkedIn	Detail

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout

The queries are:
select * from following natural join job natural join company where sid = (…) and posttype='Public' order by posttime DESC;
select cid,cname from following natural join company where sid=(…);
The first query is optimized due to similar reason.

ComHome.php
This is the home page for company user.



Hello! You have logged in as MicrosoftLogout

Post Job list

Job Id	Job Title	Job Location	Post Time	Detail Link
2	Software Developer	New York City	2018-05-08 12:32:12	Details
3	Software Developer	New York City	2018-05-08 12:32:12	Details
8	Software Developer	New York City	2018-04-10 10:53:22	Details

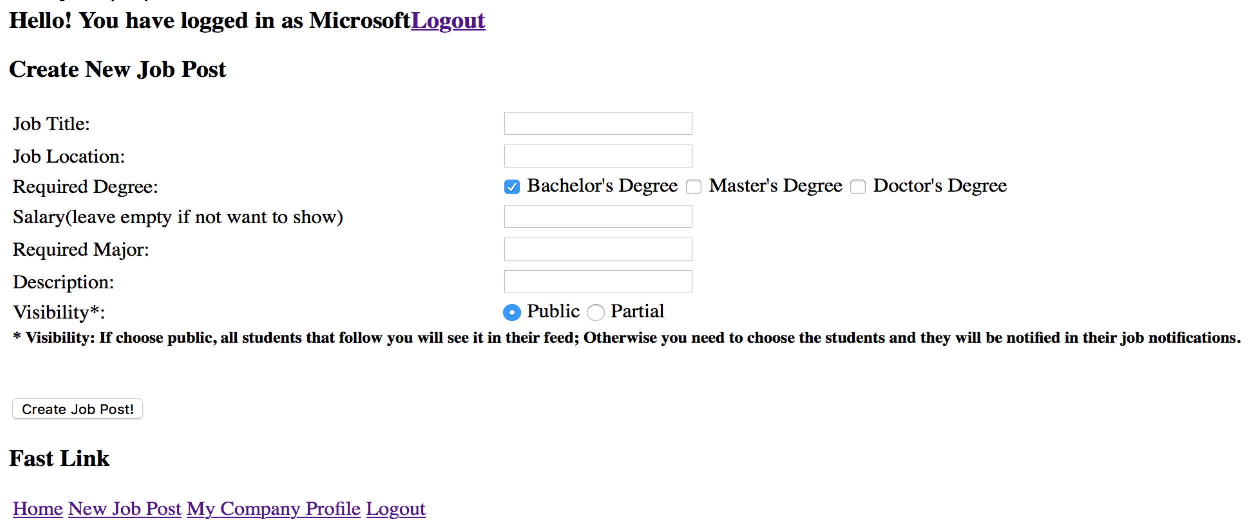
Click here to create new job post

Fast Link

Home New Job Post My Company Profile Logout

The queries used are:
select * from job where cid = (…) order by posttime DESC;
It is a simple query.

Newjob.php



Hello! You have logged in as Microsoft Logout

Create New Job Post

Job Title:	
Job Location:	
Required Degree:	[✓ Bachelor's Degree] Master's Degree Doctor's Degree
Salary(leave empty if not want to show)	
Required Major:	
Description:	
Visibility*:	[✓ Public] Partial

*** Visibility: If choose public, all students that follow you will see it in their feed; Otherwise you need to choose the students and they will be notified in their job notifications.**

Create Job Post!

Fast Link

Home New Job Post My Company Profile Logout

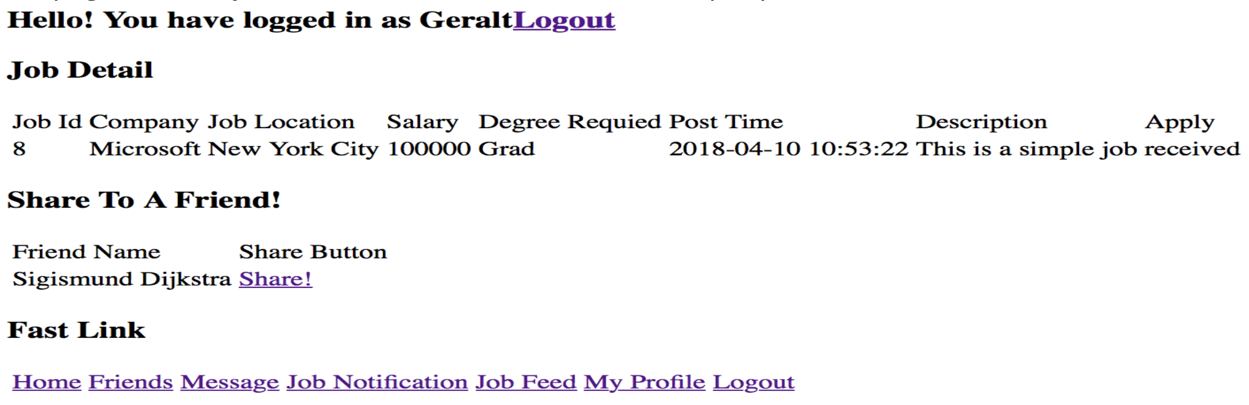
The query used is:

insert into job (\`cid\`,\`title\`,\`jlocation\`,\`salary\`,\`major\`,\`academicbar\`,\`descrp\`,\`posttype\`) values(…);

It is an insertion query. The input value type check is done at frontend to make sure the data type is correct.

Job.php

It a page to show job details for both student and company user.



Hello! You have logged in as Geralt Logout

Job Detail

Job Id	Company	Job Location	Salary	Degree Requied	Post Time	Description	Apply
8	Microsoft	New York City	100000	Grad	2018-04-10 10:53:22	This is a simple job	received

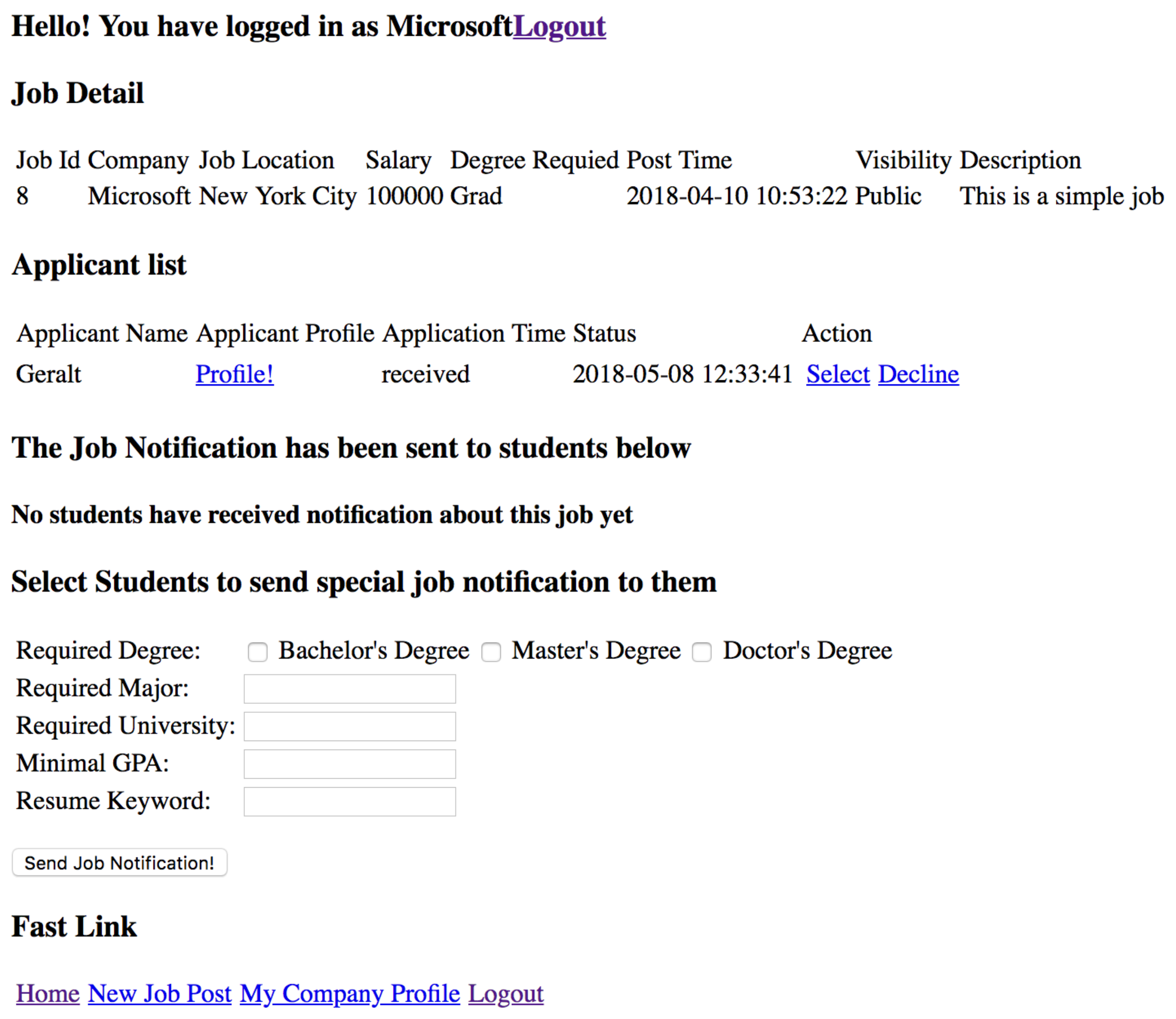
Share To A Friend!

Friend Name	Share Button
Sigismund Dijkstra	Share!

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout

Student Page Style



Hello! You have logged in as Microsoft Logout

Job Detail

Job Id	Company	Job Location	Salary	Degree Requied	Post Time	Visibility	Description
8	Microsoft	New York City	100000	Grad	2018-04-10 10:53:22	Public	This is a simple job

Applicant list

Applicant Name	Applicant Profile	Application Time	Status	Action
Geralt	Profile!	received	2018-05-08 12:33:41	Select Decline

The Job Notification has been sent to students below

No students have received notification about this job yet

Select Students to send special job notification to them

Required Degree:	☐ Bachelor's Degree ☐ Master's Degree ☐ Doctor's Degree
Required Major:	
Required University:	
Minimal GPA:	
Resume Keyword:	

Fast Link

Home New Job Post My Company Profile Logout

Company Page Style

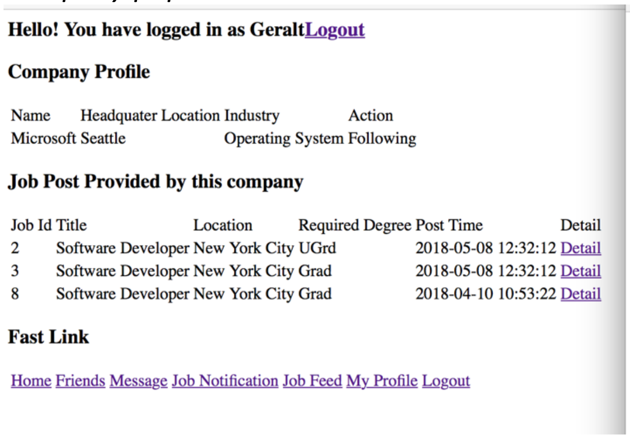
The queries used are:

select * from job where jid =(…);
insert into JobNotification (`jid`,`sid`,`source`) values(…,…,'Friend');
select * from application where jid = (…)and sid =(…);
insert into application (`jid`,`sid`) values(…,…)
select * from application where jid = (…) and sid = (…);
select * from job natural join Company where jid=(…)
select * from friend natural join student where friend.fid = (…) and friend.sid=student.sid
update application set status='selected' where jid= (…) and sid= (…);
update application set status='rejected' where jid= (…) and sid= (…);
select sname,sid,status,apptime from application natural join student where jid = (…) order by apptime DESC;
insert into JobNotification (`jid`,`sid`,`source`)values(…,…,'Company');

select * from jobnotification natural join student where jid = (…) and source='Company' order by mtime DESC;

Most of them are common query. The last one is optimized to filter by 'jid' first because the tuples that match the source check are much more than those that match the jid check.

Company.php



Hello! You have logged in as Geralt Logout

Company Profile

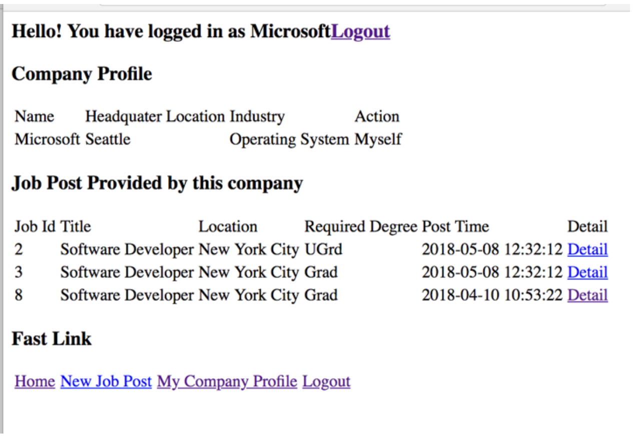
Name	Headquater	Location	Industry	Action
Microsoft	Seattle		Operating System	Following

Job Post Provided by this company

Job Id	Title	Location	Required Degree	Post Time	Detail
2	Software Developer	New York City	UGrd	2018-05-08 12:32:12	Detail
3	Software Developer	New York City	Grad	2018-05-08 12:32:12	Detail
8	Software Developer	New York City	Grad	2018-04-10 10:53:22	Detail

Fast Link

Home Friends Message Job Notification Job Feed My Profile Logout



Hello! You have logged in as Microsoft Logout

Company Profile

Name	Headquater	Location	Industry	Action
Microsoft	Seattle		Operating System	Myself

Job Post Provided by this company

Job Id	Title	Location	Required Degree	Post Time	Detail
2	Software Developer	New York City	UGrd	2018-05-08 12:32:12	Detail
3	Software Developer	New York City	Grad	2018-05-08 12:32:12	Detail
8	Software Developer	New York City	Grad	2018-04-10 10:53:22	Detail

Fast Link

Home New Job Post My Company Profile Logout

The queries used are:

SELECT * FROM company where cid =(…);
SELECT * FROM following where cid = (…) and sid =(…);
insert into following (`cid`,`sid`) values (…,…);
select * from job where cid=(…) and posttype='Public' order by posttime DESC;

These are all common queries that do not need optimization.

(c) Test and common function

Mysql.sql is the script that create tables and test data.

Function.inc is the script for functions that used across the sites, including user login authentication, page header and footer output, etc. The idea of using this file is to apply object-oriented programming prospective into frontend design to make program more manageable and unitize the mods and functionalities.

5. Optimization of Part 1

In part 2, I optimize student table by adding 'access' attribute to provide private information access restriction which is missing in part 1.

I also add 'rgtime' attribute to both student and company info to help get new user id to provide better experience for users.

For comments provided for part 1 by TAs: friend table PK are clear in this report, it was a typo in part 1 report; I used enum type in both part 1 and part 2 report, which means job notification source is either student or company; Functionality of user can limit access to their profile added in this part of design.