Progress Report 2 for ConGrad

Group members: Jieyu Ren, Qi Cheng, Zijun Liu, Denglin Zhang, Hao Yang, Tao Huang

1.Completed Work

1.1. Progress for Information Collection

An essential part of our project is the dataset that we can use to generate acceptance patterns and to be displayed on the search page. In addition, only after we have collected some information can our team test and debug our programs. Therefore, our team continued to enlarge the dataset and to collect more cases that our algorithm can use.

Since 60 schools are sufficient for our project, we enlarge our dataset by entering more programs to be displayed and more admission cases to be used for our searching and suggesting algorithms. This process is important because our team realizes that the admission cases with a result of "accepted" can help improve our algorithm's precision.

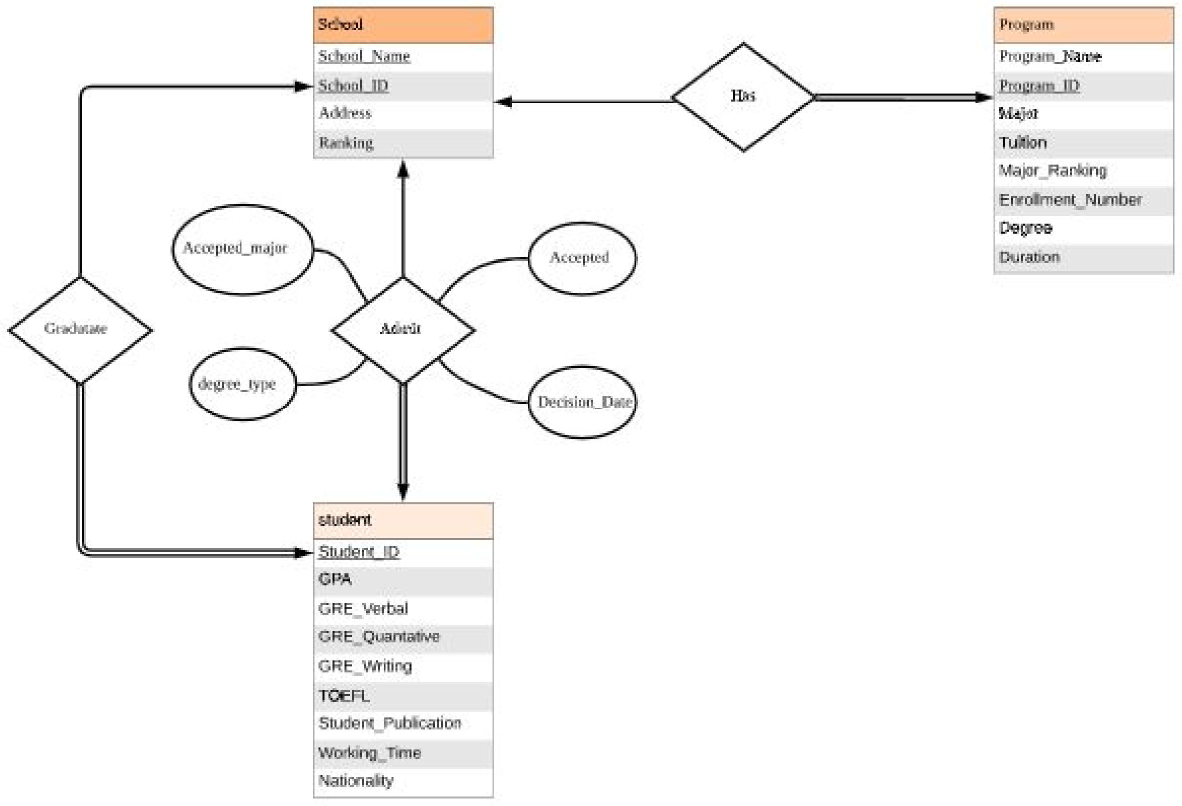
1.2. Database Design and Initialization

Our application requires a large amount of data about various schools, programs, and students, so we needed to design our database carefully so that all the information can be properly stored when we are doing data collection and easily retrieved in the later phase of the program when we need to analyze the data.

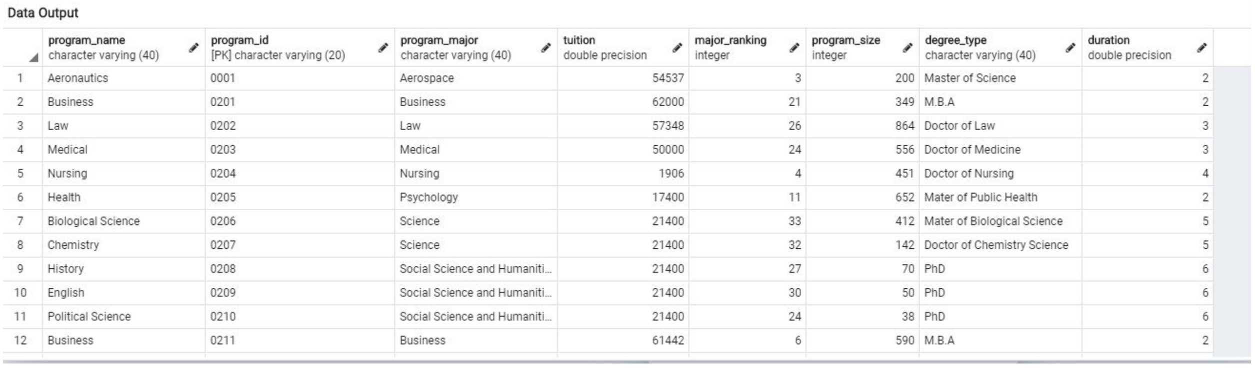
As a result, we designed six entities for our database. School entity to store school name, made-up school ID, address of schools, and U.S. News ranking of schools, with school ID being the primary key of the entity school. Entity program to store program name, made-up program ID, major of programs, tuitions, major ranking according to U.S. News, the total number of enrollment of programs, degree type provided by the program, with program ID being the primary key. To link each school with its programs, we designed a relational entity since each school has multiple programs. The entity Has got only two keys, school ID and program ID which are all foreign keys and it is a many-to-one relationship since each school can have many programs but each program can only belong to one school.

There is an entity Student which stores all the applicants' information we found in the data collection process. It stores made-up student ID, GPA, GRE scores for different sections, TOEFL score, a boolean called student publication to indicate whether the student had published any articles, work experience, and nationality of the student. To link students with the schools they applied and with the schools they graduated from, we designed two relational entities called Admit and Graduate. For relational entity Admit, it has two foreign keys which are student ID and School ID from entity Student and School. It also has a boolean called Accepted to indicate the decision made by schools, decision data to indicate when the student received the result, Accepted major to indicate which major the student applied and degree type to indicate which degree the student applied. The relational entity Admit has a many to many relationships since each student can apply for many schools and each school can have many applicants. Then there is a relational entity Graduate to indicate where the student graduated. It has only two foreign keys, student ID and school ID and many to one relationship since a student can only graduate from one school and each school can have many students graduate.

ER Diagram for ConGrad

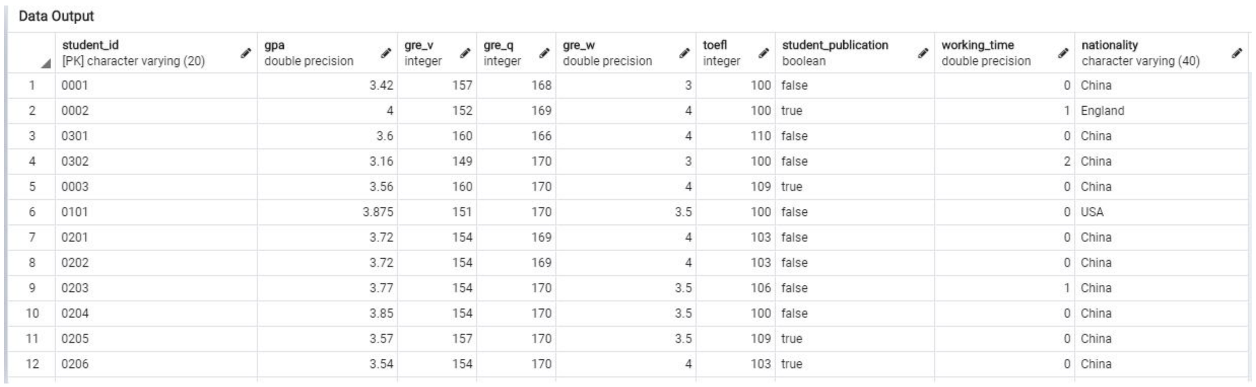


We have successfully initialized our database as descripted above. In addition, we have put all the data we collected online into the database so that the database is ready to be connected to the back end as well as the front end applications. Below is a screenshot of a scratch of our database in PostgreSQL.



Data Output

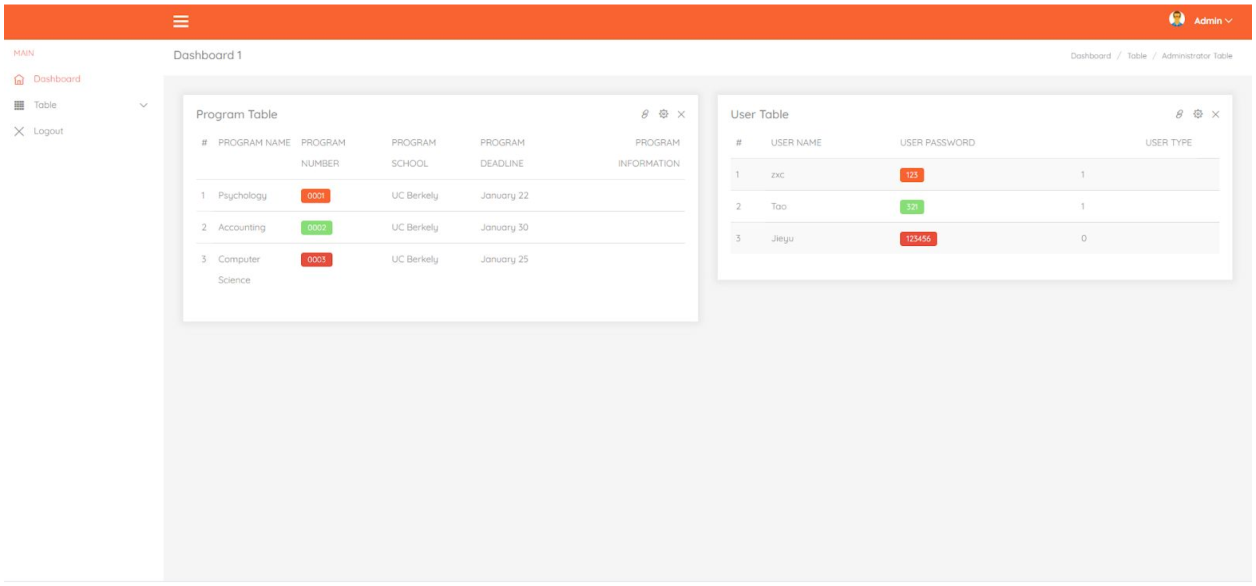
	program_name character varying (40)	program_id [PK] character varying (20)	program_major character varying (40)	tuition double precision	major_ranking integer	program_size integer	degree_type character varying (40)	duration double precision
1	Aeronautics	0001	Aerospace	54537	3	200	Master of Science	2
2	Business	0201	Business	62000	21	349	M.B.A	2
3	Law	0202	Law	57348	26	864	Doctor of Law	3
4	Medical	0203	Medical	50000	24	556	Doctor of Medicine	3
5	Nursing	0204	Nursing	1906	4	451	Doctor of Nursing	4
6	Health	0205	Psychology	17400	11	652	Mater of Public Health	2
7	Biological Science	0206	Science	21400	33	412	Mater of Biological Science	5
8	Chemistry	0207	Science	21400	32	142	Doctor of Chemistry Science	5
9	History	0208	Social Science and Humaniti...	21400	27	70	PhD	6
10	English	0209	Social Science and Humaniti...	21400	30	50	PhD	6
11	Political Science	0210	Social Science and Humaniti...	21400	24	38	PhD	6
12	Business	0211	Business	61442	6	590	M.B.A	2



Data Output

	student_id [PK] character varying (20)	gpa double precision	gre_v integer	gre_q integer	gre_w double precision	toefl integer	student_publication boolean	working_time double precision	nationality character varying (40)
1	0001	3.42	157	168	3	100	false	0	China
2	0002	4	152	169	4	100	true	1	England
3	0301	3.6	160	166	4	110	false	0	China
4	0302	3.16	149	170	3	100	false	2	China
5	0003	3.56	160	170	4	109	true	0	China
6	0101	3.875	151	170	3.5	100	false	0	USA
7	0201	3.72	154	169	4	103	false	0	China
8	0202	3.72	154	169	4	103	false	0	China
9	0203	3.77	154	170	3.5	106	false	1	China
10	0204	3.85	154	170	3.5	100	false	0	China
11	0205	3.57	157	170	3.5	109	true	0	China
12	0206	3.54	154	170	4	103	true	0	China

1.3. Progress for UI Design (Tao)



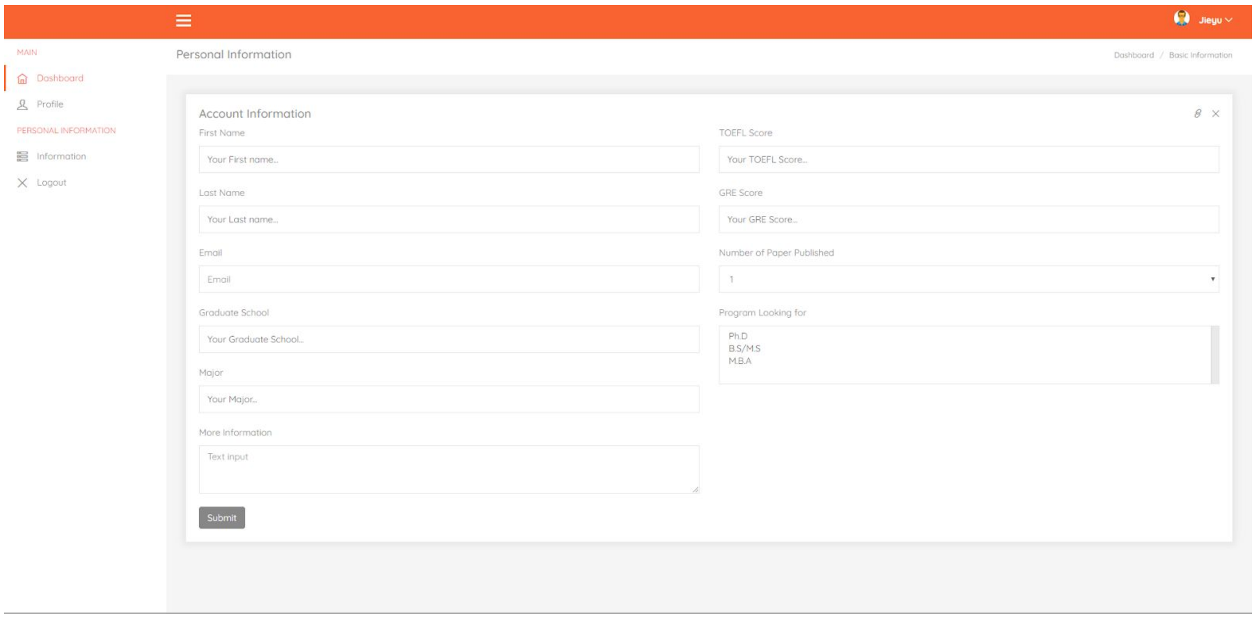
Program Table

#	PROGRAM NAME	PROGRAM NUMBER	PROGRAM SCHOOL	PROGRAM DEADLINE	PROGRAM INFORMATION
1	Psychology	0001	UC Berkely	January 22	
2	Accounting	0002	UC Berkely	January 30	
3	Computer Science	0003	UC Berkely	January 25	

User Table

#	USER NAME	USER PASSWORD	USER TYPE
1	zxc	123	1
2	Tao	321	1
3	Jieyu	123456	0

The administrator page will allow the administrator who uses specific username and password to log into this page and create, update, delete and read the programs' information. It also allows the administrator to change the information of registered users.



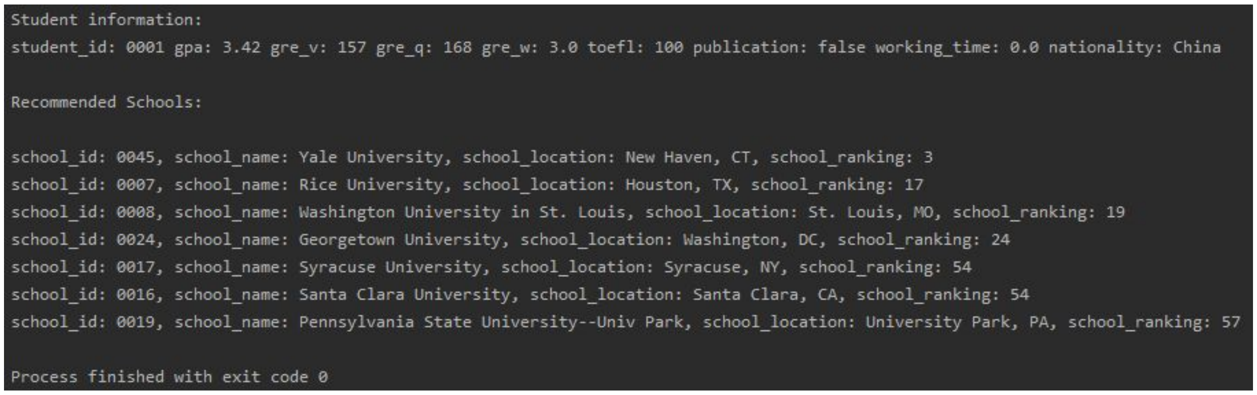
After the users register and log into the front page, the personal information page will allow them to fill out their personal information(such as name, email, gender and age)as well as their education experience(such as GRE, TOEFL score, number of papers published, and major information). The users can change their information in other pages at any time when clicking the top right button and choosing the profile button to get into this page. The backend algorithm will recommend the programs based on the information which users enter.

1.4. Progress for Database Connectors

In order to utilize our database in Java, we developed a package called DBConnect to execute SQL statements. There are three main classes under DBConnect, which are called ProgramDBConnect, SchoolDBConnect and StudentDBConnect. Some typical methods include getAcceptedStudentGPA(school_id), getAllSchool(), and getAllProgramNames(). These connectors are completed, tested and optimized since progress report 1 was submitted.

1.5. Progress for Recommendation Algorithm

The main recommendation method called Generate() is in the Generator class and under the Generator package. It takes a student object as input and returns an arraylist of schools. At first, it calls the getAllSchool method to get all school information ordered by ranking and then calculate the average GPA, GRE score, Toefl score, working time and publication status of the previously admitted students of each school respectively. Then, it compares the input student's information with these average data and checks whether the input student has satisfied the requirements of each school. Of course, the student does not need to satisfy them all, but to get the school recommended by our algorithm, the student's data should not be much lower than average. Furthermore, If the student graduates from Top 50 colleges according to US News ranking, the requirements for getting recommendation will be less strict. Since progress report 1 was submitted, we optimized the recommendation algorithm so that now the generated schools are ordered by ranking. We also added more admission data to the database so that the recommendation results are more diverse. We are now trying to make the algorithm faster by shortening the database connection time. An example output for a normal student is shown below.

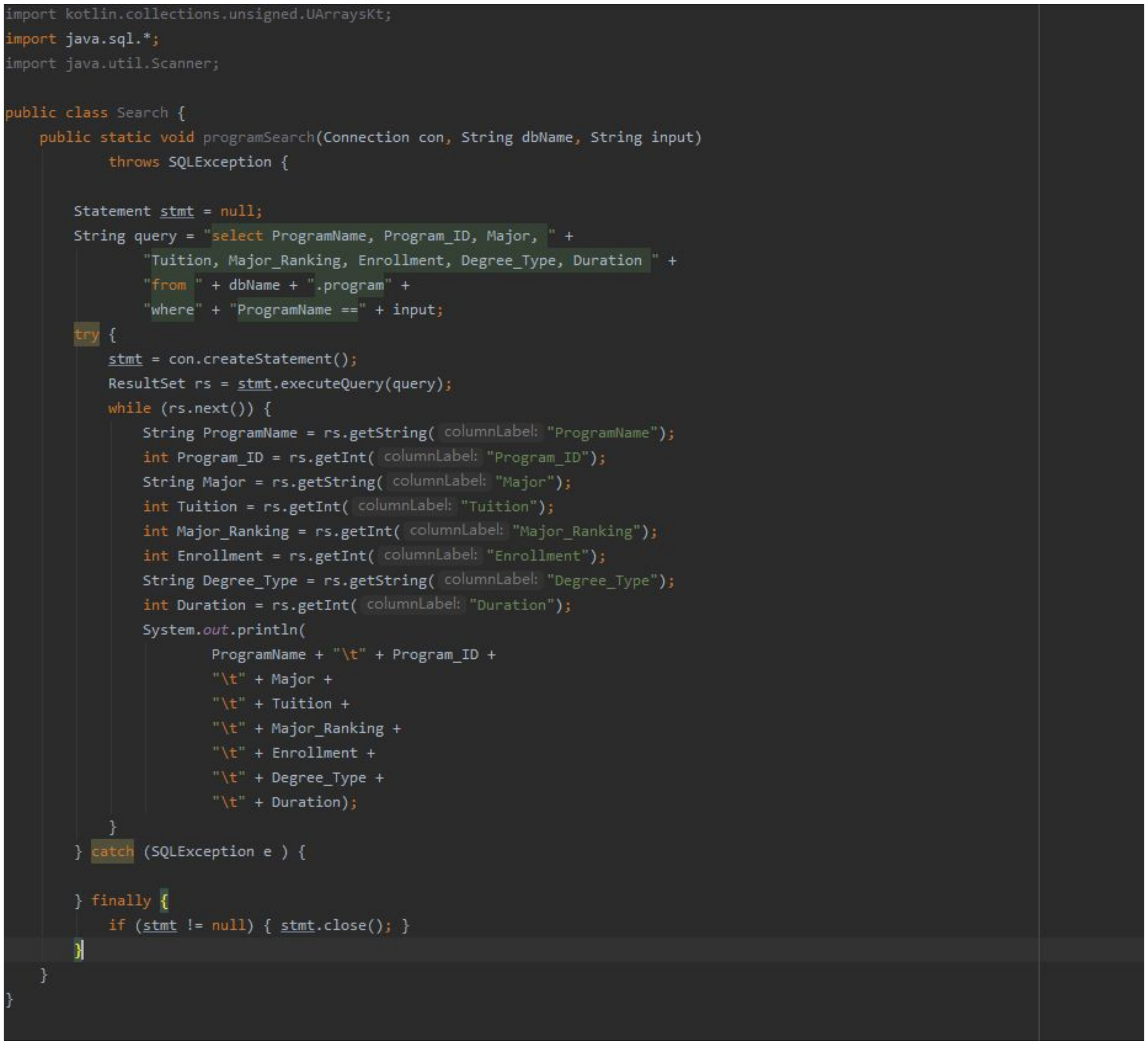


```
Student information:
student_id: 0001 gpa: 3.42 gre_v: 157 gre_q: 168 gre_w: 3.0 toefl: 100 publication: false working_time: 0.0 nationality: China

Recommended Schools:

school_id: 0045, school_name: Yale University, school_location: New Haven, CT, school_ranking: 3
school_id: 0007, school_name: Rice University, school_location: Houston, TX, school_ranking: 17
school_id: 0008, school_name: Washington University in St. Louis, school_location: St. Louis, MO, school_ranking: 19
school_id: 0024, school_name: Georgetown University, school_location: Washington, DC, school_ranking: 24
school_id: 0017, school_name: Syracuse University, school_location: Syracuse, NY, school_ranking: 54
school_id: 0016, school_name: Santa Clara University, school_location: Santa Clara, CA, school_ranking: 54
school_id: 0019, school_name: Pennsylvania State University--Univ Park, school_location: University Park, PA, school_ranking: 57

Process finished with exit code 0
```

1.6. Progress for Search Algorithm

We needed to implement a search algorithm in order for the search box to work. As described in the proposal, our search box should let users input either school name or program name so that a separate web page can pop up to show the result of the search.

For now we have implemented the program name search part of the search algorithm like in below screenshot. The program still needs revisement and tests before it can be connected to the front end application.



```java
import kotlin.collections.unsigned.UArraysKt;
import java.sql.*;
import java.util.Scanner;

public class Search {
    public static void programSearch(Connection con, String dbName, String input)
            throws SQLException {

        Statement stmt = null;
        String query = "select ProgramName, Program_ID, Major, " +
                "Tuition, Major_Ranking, Enrollment, Degree_Type, Duration " +
                "from " + dbName + ".program" +
                "where" + "ProgramName ==" + input;
        try {
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                String ProgramName = rs.getString( columnLabel: "ProgramName");
                int Program_ID = rs.getInt( columnLabel: "Program_ID");
                String Major = rs.getString( columnLabel: "Major");
                int Tuition = rs.getInt( columnLabel: "Tuition");
                int Major_Ranking = rs.getInt( columnLabel: "Major_Ranking");
                int Enrollment = rs.getInt( columnLabel: "Enrollment");
                String Degree_Type = rs.getString( columnLabel: "Degree_Type");
                int Duration = rs.getInt( columnLabel: "Duration");
                System.out.println(
                        ProgramName + "\t" + Program_ID +
                        "\t" + Major +
                        "\t" + Tuition +
                        "\t" + Major_Ranking +
                        "\t" + Enrollment +
                        "\t" + Degree_Type +
                        "\t" + Duration);
            }
        } catch (SQLException e ) {

        } finally {
            if (stmt != null) { stmt.close(); }
        }
    }
}
```

1.7. Progress for Test

Code testing is an essential part of creating software and we used JUnit testing to accelerate our testing process and make sure we would have sufficient time to complete or revise the rest of the work. We used JUnit testing to test both front-end and back-end like our search algorithm and recommendation algorithm. We finally achieved almost 55% test coverage. And we still need to try our best to improve at this point in order to make the coverage higher.

2. Project Completion Plan

Task	**Members in charge**	**Deadline**
Enlarge the dataset by collecting more information about past applications	Everyone	March 12nd (done)
User page and information display page's frameworks ready	Tao, Hao, Zijun	March 19th (done)
Account management page's framework ready	Tao, Hao, Zijun	March 21th (done)
Code for the search information ready	Qi, Denglin	March 21st (done)
Database and backend connection	Qi, Jieyu	March 23rd (done)
Backend and frontend connection	Tao, Qi, Jieyu, Zijun, Hao, Denglin	March 23rd (done)
Webpage for the users to	Tao, Hao, Zijun	March 26th (done)

update application progress ready		
Progress report 2 submission	Everyone	April 4th (done)
Functionality test	Jieyu, Zijun, Qi	April 5th (in progress)
Web view beautification	Everyone	April 10th (in progress)
Final report draft	Everyone	April 18th
Final report ready	Everyone	April 25th
Final project submission	Everyone	April 27th

3. Member Contributions

Jieyu Ren:

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Wrote Abstract, Proposal, Progress Report 1 and Progress Report 2
- Constructed the database
- Inserted data into the database
- Wrote the database connectors
- Wrote and optimized the backend recommendation algorithm

Zijun Liu:

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Wrote Abstract, Proposal and Progress Report 1 and Progress Report 2
- Entered information into the database

- Web page UI mockup
- Code review and Testing

Hao Yang:

- Co-designed system architecture
- Collected information about admission information from previous years
- Code Review and Testing
- Designed and implemented the search page for the web system
- Inserted data into database
- Wrote Abstract, Proposal and Progress Report 1 and Progress Report 2

Tao Huang

- Co-designed system architecture
- Collected information about admission information from previous years
- Web page UI mockup
- Designed and implemented the administrator page for the web system
- Designed and implemented the user profile page for the web system
- Wrote Abstract, Proposal, Progress Report 1 and Progress Report 2

Qi Cheng

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Designed database schema
- Inserted information into the database
- Wrote Abstract, Proposal, Progress Report 1 and Progress Report 2
- Implemented the search algorithm

Denglin Zhang

- Wrote proposal

- Collected information about graduate schools, programs, and past applications