

Final Project Report for Congrad

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

1. Background and Objectives

1.1 Background

ConGrad is a web application dedicated to tracking and providing advice surrounding the graduate school application process. It has two major functions: 1. It allows users to search admission information from previous years and use our sort and time filters to refine the results. 2. It generates a recommended program list based on users' application information. In the normal process of applying for graduate schools, there are too many programs/schools that are listed on various websites. Students would need a lot of time for the information gathering process. With our web app, students can find all the information they need in our system and get recommendations for the school/programs that they are probably interested in and have a larger chance of getting accepted given their academic status.

1.2 Objectives, Success Criteria, Measures and Results

No	Component	Success Criteria	Measure	Priority	Results
1	UI Design	The UI design is able to anticipate what users might need to do and ensure that the interface has	Effective input controls (buttons, text fields, check boxes, etc.)	1	Completed

		elements that are easy to access, understand, and use to facilitate those actions.	Effective navigational components (search field, breadcrumbs, etc.)	1	Completed
			Purposeful page layout: design the spatial relationships between items based on importance.	1	Completed
2	Algorithm	The algorithms should be able to return the most optimal solutions using a reasonable amount of time.	The recommendation algorithm should return schools/programs with the highest ranking that are likely to accept our users.	1	Completed
			The recommendation algorithm should return the results within 10 seconds	2	Completed
			The search algorithm should return the correct information of the schools/programs that our users are interested in within 3 seconds	1	Completed
3	Database Management	Our web application should provide	Accessible entry with security concerns addressed to future	1	Completed

		access for administrators and users to update information about schools, programs, and admission cases.	administrators to edit the database.		
			Access to users to upload their application's admission results.	1	Completed
			Access to users to communicate with each other by leaving comments on our web application	2	Incomplete

1.3 Features

1.3.1 Major Features

These are the features that we think are the most important and can likely be completed within a semester.

MF-1	Recommendation algorithm: Our web application is able to return programs for users according to their GPA, the rank of their current college, GRE score, current major in college, desired major for graduate school and desired location of graduate school to recommend master programs. The level of recommendation for a program is based on the possibility of acceptance, the major ranking of a program and user preference.
MF-2	Search result display algorithm: Our web application provides a search box for users to search for specific programs or schools manually.

MF-3	Database update: Our web application provides access to future administrators to update the information about schools and school programs. We also enable users to share their admission results to enlarge our dataset.
------	--

1.3.2 Stretch Features

These are the features that are less important and we would like to complete them if we have time left.

SF-1	Recommendation algorithm: We should optimize our algorithm so that it takes no more than 5 seconds to give back results.
SF-2	Forum development: We want to create a forum on our web application to enable users to share opinions and experiences with each other.

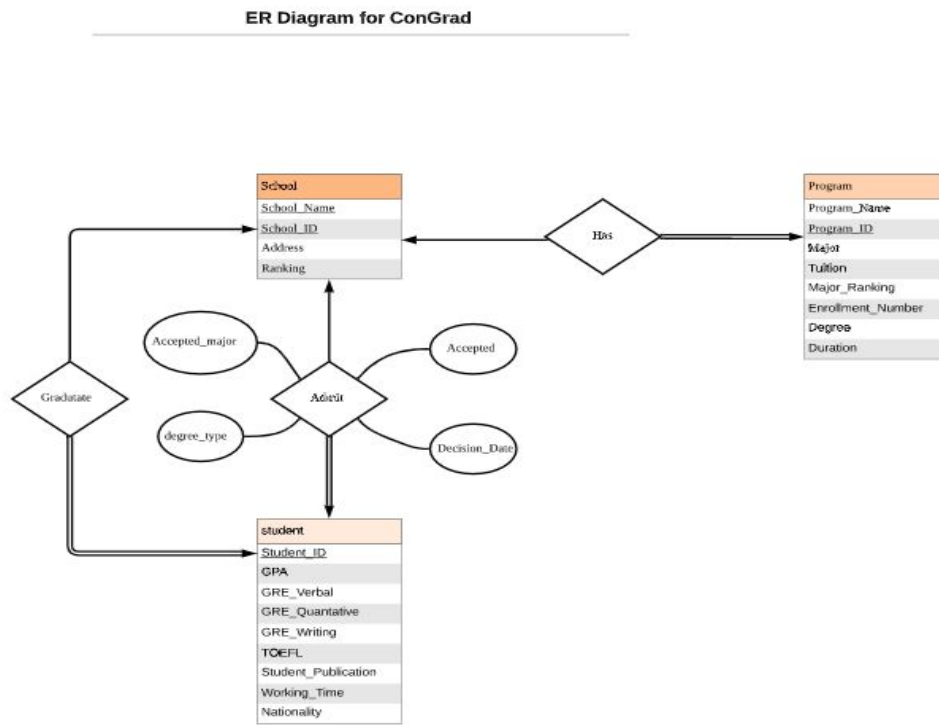
2. Deliverable Report

2.1 Information Collection

For the dataset, we basically collected information about schools and students. We used the national ranking of colleges from US News & Report as our primary source. In the table School, we included the name, location, and overall ranking of each top 50 school. Then we took advantage of the official websites of each school to collect information on 10 popular graduate programs, including the major, program, tuition, program size, degree, and duration. On the other hand, we also collected applicants' information, which includes GPA, GRE score, TOEFL score, publication, working time, nationality, decision, decision date, major, and degree. We basically used online forums and other websites in which applicants share their admission results with others such as The GradCafe as our primary sources. Due to the relatively large data size we need for our algorithm, we were trying to collect as much data as we could. Therefore, part of our applicant data was from college application agencies.

2.2 Database

We created a database containing six tables and three of those tables are the relational entity tables. The tables are school, Has, Program, Admit, Student, and Graduate. The relationship between tables and their entries are in the following ER diagram.



2.3 Recommendation

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 be close to average at least. Furthermore, If the student graduates from Top 50 colleges according to the US News Ranking, the requirements for getting recommendations will be less strict.

2.4 Search

Users can use our search feature to search for programs that they are interested in. This feature was achieved by connecting the front end search box on the main web page to the back end search algorithm and database. When users input the program names they want to search, a method called search with loop-through our database and find the programs with the matched names. After inputting the program name, users will see a new web page with the program information that the search method returned. The information returned includes program names, major, tuition, ranking, the total number of enrollment, and degree type the program provides. The search result page will show “No Result Found” if there is no program that matched the user input.

2.5 UI

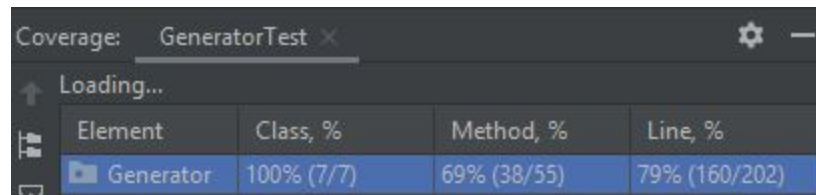
UIs have been connected to corresponding pages and the backend database has been connected to different pages to show the output and input. Also the recommendation algorithm is finished and it is now added into the main page(recommendation page).

school	program	information1	information2	operation
A1	program1	information1	information1	<button>Archive</button> <button>Not Intersted</button>
A1	program1	information1	information1	<button>Archive</button> <button>Not Intersted</button>
A1	program1	information1	information1	<button>Archive</button> <button>Not Intersted</button>
A1	program1	information1	information1	<button>Archive</button> <button>Not Intersted</button>

Copyright © 2020 Company name All rights reserved.

2.6 Code Test

Code testing is an essential part of creating a software application. We used JUnit to test both our front-end and back-end codes, like the search algorithm and the recommendation algorithm. Finally, we achieved more than 70% coverage (getters and setters are not included), which is shown below:



Coverage: GeneratorTest				
Loading...				
Element	Class, %	Method, %	Line, %	
Generator	100% (7/7)	69% (38/55)	79% (160/202)	

3. Future Work

We have developed all the functions we initially expected. Our next step is to refine our code test, we would probably write more cases to test in order to detect the bugs of some extreme cases. In the case of time permitting, we would either beautify our User Interface or refine our search algorithm. By beautifying our User Interface, we would find and add more pictures to the interface. By refining our search algorithm, we would probably apply the spell checker algorithm into the search algorithm to make our application more convenient to use when a user misspells the word. If the user misspells a word, the spell checker algorithm will provide some possible correct outcomes to the user. For example, If the user wants to search New York University but the user types “Nwe York University”, the application would still provide the outcome of New York University.

4. Member Contribution

Jieyu Ren:

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Wrote Abstract, Proposal, Progress Report 1 & 2, and Final Report

- Constructed the database
- Inserted data into the database
- Wrote the database connectors
- Wrote the backend recommendation algorithm
- Wrote the Junit tests for the Generator and the Search class

Qi Cheng

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Designed database schema
- Implemented Search algorithms and tests for it
- Inserted information into the database
- Wrote Abstract, Proposal, Progress Report 1 & 2, and Final Report

Zijun Liu:

- Co-designed system architecture
- Collected information about graduate schools, programs, and past applications
- Co-designed database schema
- Co-designed UI
- Wrote Abstract, Proposal, Progress Report 1 & 2, and Final Report
- Wrote functional tests for database connection parts

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, Progress Report 1 & 2, and Final Report

Denglin Zhang:

- Collected information about graduate schools, programs, and past applications
- Wrote proposal and Final Report

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 & 2, and Final Report