

COMP3111: Software Engineering

Project Report

Decision-Making System for University Selection

I. Brief Overview

This university comparison system aims to help aspiring students to choose universities that may best fit to their liking. Our system is divided into 3 functions, so that there are multiple ways of intuitively visualizing how one university is better than the other.

II. Sample Inputs and Outputs

A. Task 1

The first functionality requires the user to input a year to generate the overview of the university scores in table view, pie chart view and bar chart view (from left to right).

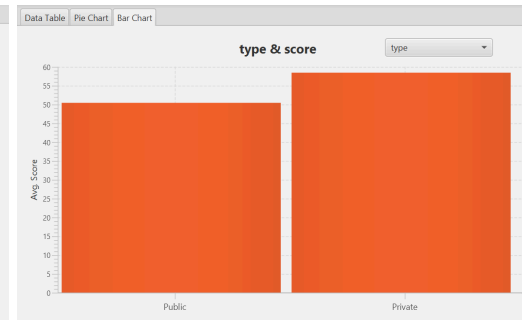
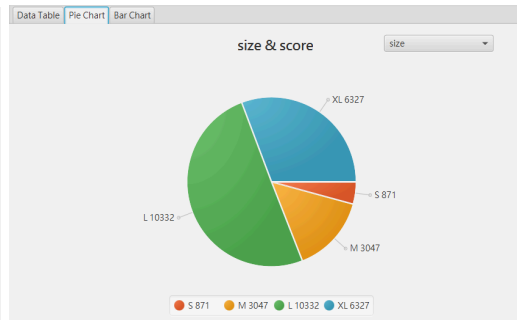
Task #1: Overview of data based on year input

Inputs

Year: 2017

Search Reset

All Information					
Rank	University	Score	Country	City	Type
1	Massachusetts Institute of T...	100	United States	Cambridge	Private
2	Stanford University	98.7	United States	Stanford	Private
3	Harvard University	98.3	United States	Cambridge	Private
4	University of Cambridge	97.2	United Kingdom	Cambridge	Public
5	California Institute of Techn...	96.9	United States	Pasadena	Private
5	University of Oxford	96.8	United Kingdom	Oxford	Public
7	UCL	95.6	United Kingdom	London	Public
8	ETH Zurich - Swiss Federal L...	94.2	Switzerland	Zurich	Public
9	Imperial College London	94.1	United Kingdom	London	Public
10	University of Chicago	93	United States	Chicago	Private
11	Princeton University	92.8	United States	Princeton	Private
12	National University of Singa...	91.5	Singapore	Singapore	Public
13	Nanyang Technological Uni...	91.4	Singapore	Singapore	Public
14	EPFL	91.1	Switzerland	Lausanne	Public
15	Yale University	90.9	United States	New Haven	Private



B. Task 2

The second functionality provides the option to compare two universities. By inputting two universities of their choice and a selection of years, it compares their data based on rank, score, faculty count, international students, and student faculty ratio. It displays the data in two different ways: (1) the bar chart, which calculates the averages of each property related to the selected years, (2) the line chart, which displays the scores of each university across the selected years.

Task 2.1 Task 2.2

Task # 2.1: Select university to compare the quality of education from different dimensions

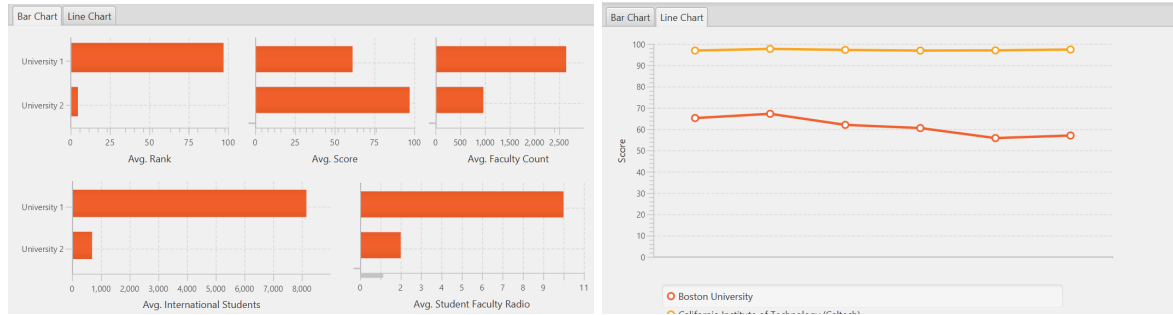
Inputs

University 1: Boston University

University 2: California Institute of T...

Year: ☒ 2017 ☒ 2018 ☒ 2019 ☒ 2020 ☒ 2021 ☒ 2022

Compare Reset



The second functionality also provides the option to compare two countries/regions. By inputting two countries/regions of their choice and a selection of years, it compares their data based on rank, score, faculty count, international students, and student faculty ratio. It displays the data in two different ways: (1) the bar chart, which calculates the averages of each property related to the selected years, (2) the line chart, which displays the average scores of each country/region across the selected years. (Note that the “All” option is also provided to allow the user to visualize how a country compares to the global averages).

Task 2.1 Task 2.2

Task # 2.2: Select two countries/regions to compare the quality of education based on QS ranking

Inputs

Country/Region... All

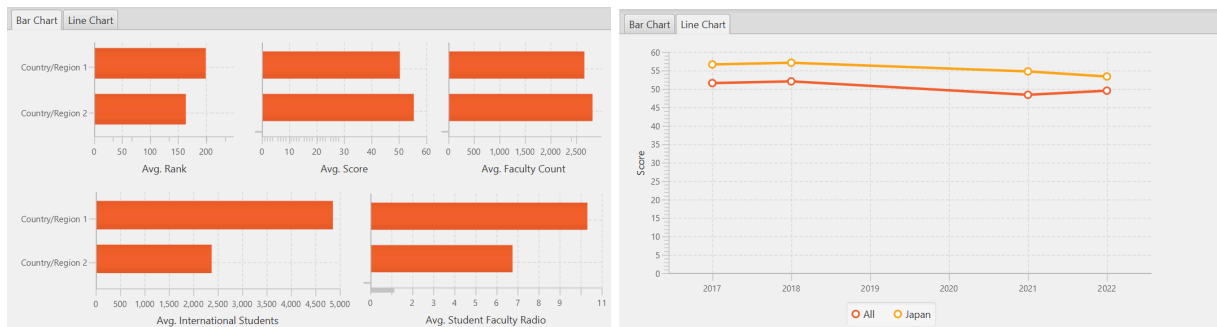
Country/Region... Japan

Year

☒ 2017 ☒ 2018 ☐ 2019

☐ 2020 ☒ 2021 ☒ 2022

Compare Reset



C. Task 3

The third function requires the user to input a ranking range, university region, and type to generate a list of recommended universities that meets the user's input criteria.

Task # 3: University suggestion based on user input

Inputs

Ranking Range Top 10 Bottom 50

Type Public

Region ALL

Recommend Reset

Recommendation				
Based on your input these universities you can prefer for higher education				
University	The Best Year of Ra...	The Best Rank	The Most Recent Y...	Rank of the most recent...
National University of Singa...	2022	11	2022	11
Nanyang Technological Uni...	2020	11	2022	12
UPFL	2018	12	2022	14
Peking University	2021	15	2022	17
The University of Edinburgh	2022	16	2022	16
Peking University	2022	18	2022	18
The Australian National Uni...	2018	20	2022	27
University of Michigan-Ann ...	2019	20	2022	23
King's College London	2017	21	2022	35
The University of Hong Kong	2022	22	2022	22
The University of Tokyo	2020	22	2022	23

III. Unit Testing Report

Our project successfully passed all the cases provided during our unit testing:

✓ Tests passed: 48 of 48 tests – 1 sec 422 ms

C:\Users\ethan\.jdk\openjdk-22\bin\java.exe ...

Process finished with exit code 0

✓ qspjproject (comp3111)	1sec 422 ms	✓ T21AnalysisTest	242 ms
✓ T1AnalysisTest	721 ms	✓ Test1_getBarChartData()	6 ms
✓ test1_getBarChartData()	7 ms	✓ Test1_getLineChartData()	139 ms
✓ test1_getKeyList()	5 ms	✓ Test2_getBarChartData()	9 ms
✓ test1_getPieChartData()	619 ms	✓ Test2_getLineChartData()	8 ms
✓ test1_getTableList()	8 ms	✓ Test3_getBarChartData()	7 ms
✓ test2_getBarChartData()	8 ms	✓ Test3_getLineChartData()	7 ms
✓ test2_getKeyList()	6 ms	✓ Test4_getBarChartData()	7 ms
✓ test2_getPieChartData()	12 ms	✓ Test4_getLineChartData()	8 ms
✓ test2_getTableList()	6 ms	✓ Test5_getBarChartData()	11 ms
✓ test3_getBarChartData()	7 ms	✓ Test6_getBarChartData()	7 ms
✓ test3_getKeyList()	6 ms	✓ Test7_getBarChartData()	6 ms
✓ test3_getPieChartData()	13 ms	✓ Test8_getBarChartData()	13 ms
✓ test3_getTableList()	7 ms	✓ Test9_getBarChartData()	7 ms
✓ test4_getKeyList()	7 ms	✓ Test10_getBarChartData()	7 ms
✓ test5_getKeyList()	5 ms		
✓ test6_getKeyList()	5 ms	✓ T22AnalysisTest	87 ms
✓ T3AnalysisTest	372 ms	✓ Test1_getBarChartData()	8 ms
✓ Test1_getRecommendData()	42 ms	✓ Test1_getLineChartData()	11 ms
✓ Test2_getRecommendData()	11 ms	✓ Test2_getBarChartData()	9 ms
✓ Test3_getRecommendData()	13 ms	✓ Test2_getLineChartData()	8 ms
✓ Test4_getRecommendData()	17 ms	✓ Test3_getBarChartData()	15 ms
✓ Test5_getRecommendData()	10 ms	✓ Test3_getLineChartData()	10 ms
✓ Test6_getRecommendData()	39 ms	✓ Test4_getBarChartData()	8 ms
✓ Test7_getRecommendData()	12 ms	✓ Test4_getLineChartData()	9 ms
✓ Test8_getRecommendData()	22 ms	✓ Test5_getBarChartData()	9 ms
✓ Test9_getRecommendData()	191 ms		
✓ Test10_getRecommendData()	15 ms		

IV. Coverage Report

Overall, our branch coverage reached 66% for the entire project:

Coverage qsproject in QSPProject x				
🔍 ⬆ ⬇ ↗ 🔍				
Element ^	Class, %	Method, %	Line, %	Branch, %
▼ comp3111.qsproject	70% (7/10)	54% (30/55)	60% (377/627)	66% (199/298)
🔵 Application	0% (0/1)	0% (0/2)	0% (0/6)	100% (0/0)
🔵 Controller	0% (0/2)	0% (0/15)	0% (0/225)	0% (0/70)
🔵 QSItem	100% (1/1)	38% (5/13)	67% (23/34)	50% (2/4)
🔵 QList	100% (1/1)	100% (2/2)	90% (37/41)	100% (12/12)
🔵 RecommendItem	100% (1/1)	100% (9/9)	100% (29/29)	62% (5/8)
🔵 T1Analysis	100% (1/1)	100% (5/5)	100% (52/52)	86% (40/46)
🔵 T3Analysis	100% (1/1)	100% (3/3)	100% (31/31)	90% (27/30)
🔵 T21Analysis	100% (1/1)	100% (3/3)	100% (76/76)	92% (48/52)
🔵 T22Analysis	100% (1/1)	100% (3/3)	96% (129/133)	85% (65/76)

V. Supplementary Notes

A. Implementation and Design

Task 1	
T1Analysis()	Implemented by extracting data of universities that are from the year specified by the user. The data are loaded into the table as columns for different aspects, namely “rank”, “name”, “score”, “country”, “city” and “type”. A numerical comparator was supplied so that numerical fields can be sorted in the correct order. These data are also used to generate a pie chart and a bar chart. The charts can dynamically change the classification aspects according to the user input from a drop-down list. This is achieved by a listener which calls the update function when necessary.
getPieChartData()	Used to generate a pie chart which shows the sum of scores of each category, according to the aspect specified by the user. If a university does not have a score, it contributes 0 to the sum.
getBarChartData()	Used to generate a bar chart which shows the average of scores of each category, according to the aspect specified by the user. The average is computed by calculating the sum of the scores, which is then divided by the total count of universities. If a university does not have a score, it contributes 0 to the sum, and is excluded from the university count.

Task 2.1	
getBarChartData()	Handled by averaging all the relevant data in the given selected years. Since there are different possible searchName inputs, we handled each property in different cases—replacing commas with periods for “score”, and removing all commas and periods for integral values like “internationalStudents” and “facultyCount”. For empty data, we simply omitted it from the calculation to ensure the data returned is as accurate as possible.
getLineChartData()	Implemented by going through every statistic related to the selected universities recorded during the selected years. If a university has no data in that particular year, it will simply be omitted in the line chart. The handling of dirty data only considered “score” as the searchName, since the interface only displays score as a statistic. For empty data, we simply omitted it from the calculation to ensure the data returned is as accurate as possible. During the implementation of the line chart, we noticed that the year ordering is sometimes not sorted in the user interface due to missing data in a particular year. We fixed this issue by forcibly setting the line chart’s x-coordinates after processing.

Task 2.2	
getBarChartData()	Handled similarly to Task 2.1’s version, except we consider the data on a country/region level instead of a particular university.
getLineChartData()	Handled much differently compared to Task 2.1, since we are considering multiple university statistics recorded in the same year. Since the data in CountryRegion1List and CountryRegion2List have already been sorted by year, all we have to do is continuously add up the data until we see a statistic with a different year, in which we can compute the average and store the data into the series before adding it to the line chart. The handling of dirty data only considered “score” as the searchName, since the interface only displays score as a statistic. For empty data, we simply omitted it from the calculation to ensure the data returned is as accurate as possible. During the implementation of the line chart, we noticed that the year ordering is sometimes not sorted in the user interface due to missing data in a particular year. We fixed this issue by forcibly setting the line chart’s x-coordinates after processing.

Task 3	
T3Analysis()	Achieved by extracting data from universities in the statistics that match the ranking range, type and region entered by the user. If the university has no data in type, the system will include the university in the consideration list for any input type. When matching universities that fit within the input rankings, the system will be based on the university's best rankings.
RecommendItem.update()	Updates the properties of the RecommendItem. RecommendItem.bestYear will store the most recent year in which university received its best rank.

Other Methods Used	
QSList.initialize()	The initialize function was simply handled by a built-in csvReader, and scans through each individual row of the csv file until all the entries have been seen.
Controller search/compare/recommend functions	It is worth noting that the empty inputs were handled by returning an error message to the user to prevent invalid values to be processed by the analyzers. *Task 3 explicitly returns all universities if no ranking range is given.

B. Assumptions and Limitations

- Our implemented system “cleans” the data under the assumption of all possible anomalies seen in the provided csv file. Hence, our system will likely not work if the format for the csv file is changed (there could be other special characters that are not handled in the data filtering sections of our code).
- The system only supports data that ranges from 2017-2022, so our system may be inaccurate in the sense that it does not provide insight on statistics recorded after the supported time. However, it still provides an insightful visualization of university data recorded in the given years.
- A university’s properties are only limited by those provided in the QS World Rankings system, so there are no additional properties/metrics that can be used to compare these universities from one another.

C. Recommendations

- One possible improvement could be to add a function for Task 2 to switch between properties (and not just score), but for the current project, it would be increasingly more complex to implement.