COMP3111: Software Engineering

Project Report

Decision-Making System for University Selection

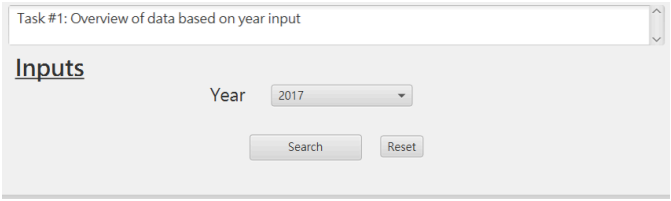
I. Brief Overview

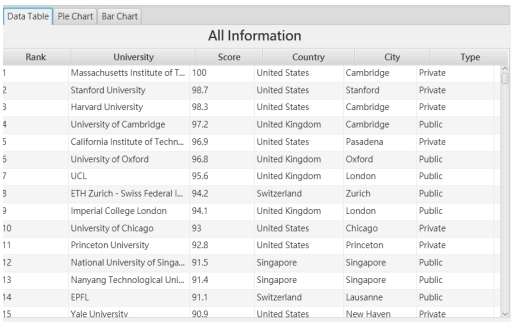
This university comparison system aims to help aspiring students to choose universities that may best fit to their liking. Our system is divided into 3 functions, so that there are multiple ways of intuitively visualizing how one university is better than the other.

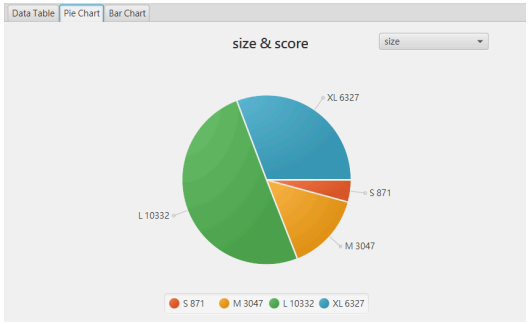
II. Sample Inputs and Outputs

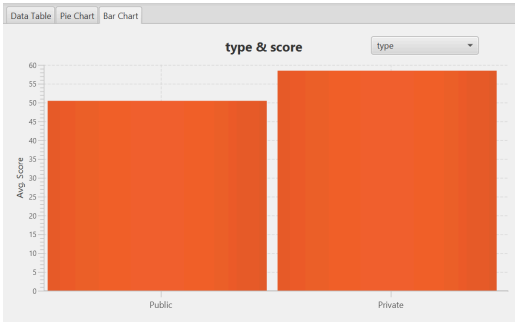
A. Task 1

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).



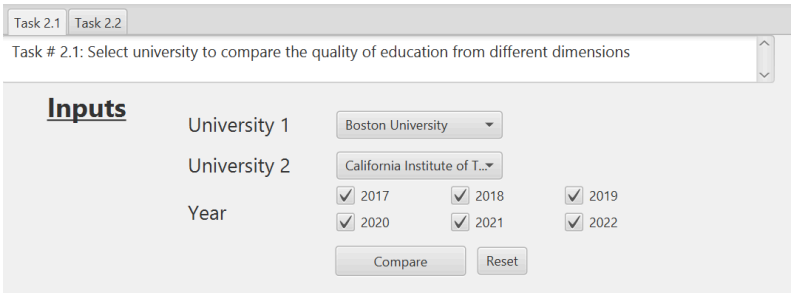


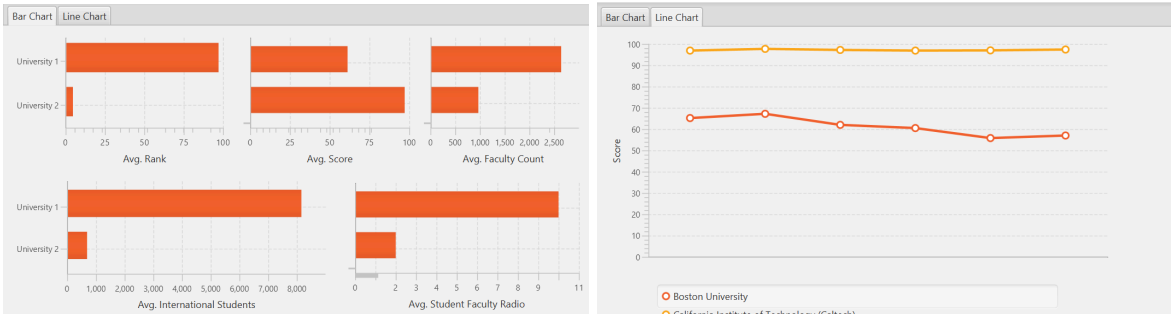




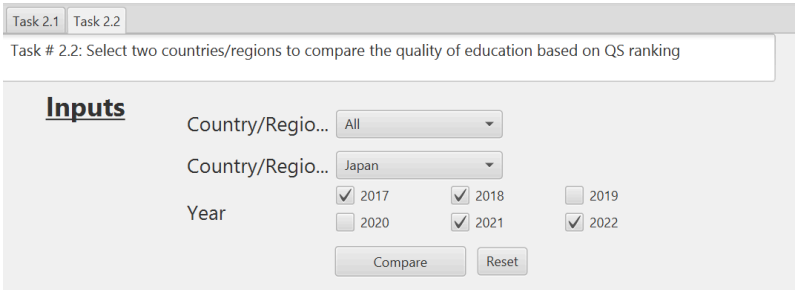
B. Task 2

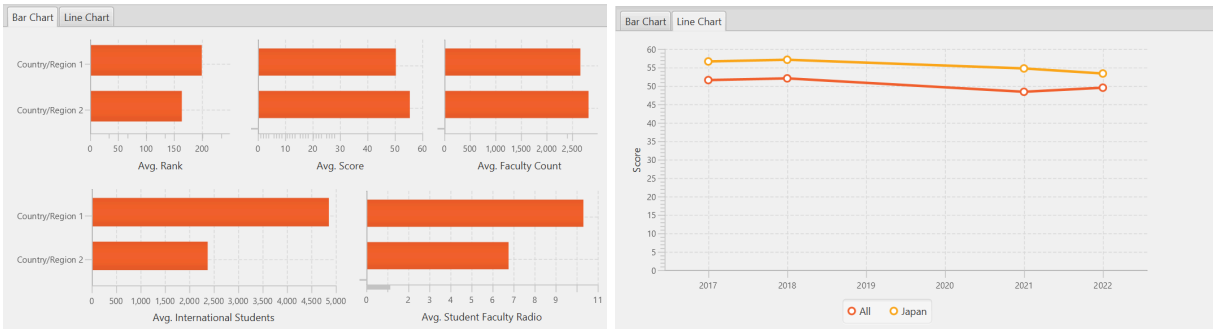
The second functionality provides the option to compare two universities. By inputting two universities of their choice and a selection of years, it compares their data based on rank, score, faculty count, international students, and student faculty ratio. It displays the data in two different ways: (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.





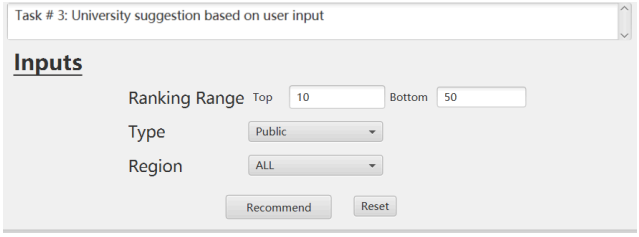
The second functionality also provides the option to compare two countries/regions. By inputting two countries/regions of their choice and a selection of years, it compares their data based on rank, score, faculty count, international students, and student faculty ratio. It displays the data in two different 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).

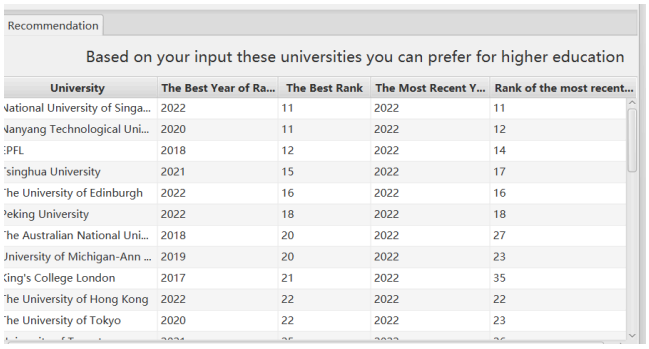




C. Task 3

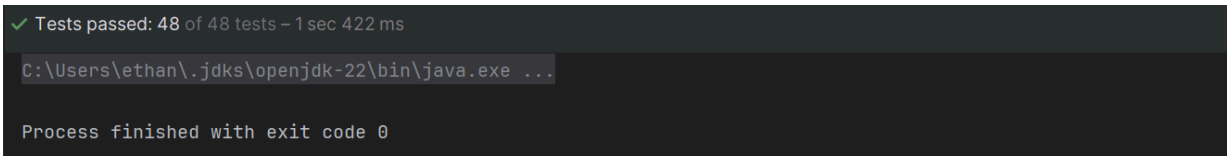
The third function requires the user to input a ranking range, university regio,n and type to generate a list of recommended universities that meets the user's input criteria.





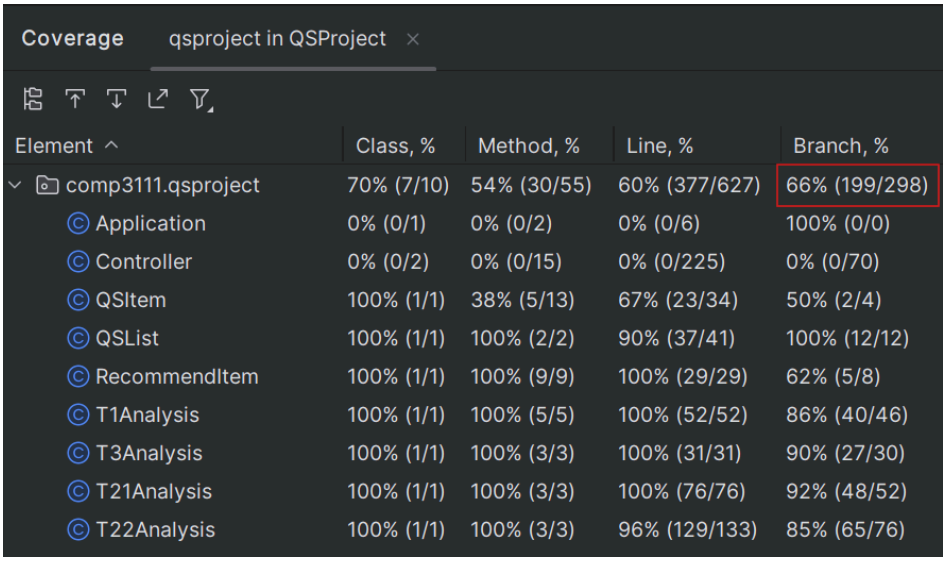
III. Unit Testing Report

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



IV. Coverage Report

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



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)
QSList	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 linster 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.