DESIGN SPECIFICATION

CS2XB3 L02 G3

leej229, lij416, ningh4, liuy363, yoonj13

Jiwoo Lee, Jennie Li, Zoe Ning, Cynthia Liu, Jeehyun Toon

1) Revision

Team Members

Name	MacID	Student Number	Roles	Responsibilities
Jiwoo Lee	leej229	400182530	Leader	Overlook team progress, worked on UI for MajorSortActivity.java, retrieving csv data, and searching/sorting algorithms
Jennie Li	lij416	400204743	Programmer	Retrieve ranking data from csv, sorting/searching for ranking/coop info (MajorSortActivity.java)
Zoe Ning	ningh4	400183343	UI Designer	Designed and implemented UI for first and second pages (MainActivity.java, MajorActivity.java)
Cynthia Liu	liuy363	400172720	UI Designer	Designed and implemented UI for first and second pages (MainActivity.java, MajorActivity.java)
Jeehyun Yoon	yoonj13	400134040	Programmer	Implemented functionalities for first page (linking to university url), organizing csv data. MainActivity.java

Revision History:

- **Feb 6 –** Initial commit
- **Feb 7 –** Submit proposal
- **Feb 13 –** Submit power point
- **Mar 7 –** Submit requirement specification
- **Mar 23 –** Finished gathering csv data
- **April 3 –** Setup android project, wrote class to retrieve data from csv files, and implemented UI for first page
- **April 5 –** Changed csv files admission average area from 'mid 80' to '85' (integer)
- **April 8 –** Implemented searching/sorting algorithm, designed and implemented third page list view
- **April 9 –** Added coop/tuition/international/sorting options (ranking, admission average, tuition) functionalities in third page. Implemented UI in second page. Started documentation
- **April 10 –** Finish up search bar and logo functionalities
- **April 11** – wrap up project, export as eclipse project
- **April 12 –** submit

Consent:

By virtue of submitting this document we electronically sign and date that the work being submitted by all individuals in the group is their exclusive work as a group and we consent to make available the application developed through CS2XB3 project, the reports, presentation, and assignments (not including my name and student number) for future teaching purposes.

2) Contribution Page

Name	Role	Contribution
Jiwoo Lee	Leader	Data - mcmaster.csv - agoma.csv - carleton.csv - Lakehead.csv - qs_world_ranking.csv University.java – all MajorSortActivity.java - everything except coop part, booleanFilter(), and getUniversityRanking() activity_major_sort.xml – all list_group.xml – everything except uni logo list_item.xml – everything except uni logo Log, Design Specification – almost everything
Jennie Li	Programmer	Data - uoit.csv - ottawa.csv - uoft.csv - waterloo.csv Program.java – all MajorSortActivity.java - getUniversityRanking() - booleanFilter() - coop swtich activity_major_sort.xml list_item.xml - worked with Zoe on adding uni logo to list items - gathered images refactor code and write better comments
Zoe Ning	UI Designer	Data - ryerson.csv - trent.csv - guelph.csv

		- hearst.csv MainActivity.java - Created button “Search by major” activity_major_sort.xml list_item.xml MajorSortActivity.java - worked with Jennie on adding uni logo to list items - mostly worked on code MajorActivity.java - all except search bar activity_major.java - designed and implemented UI for buttons refactor code and write better comments
Cynthia Liu	UI Designer	Data - laurentian.csv - nipissing.csv - ocad.csv - queens.csv MainActivity.java - Added images with universities in alphabetical order activity_main.xml - All except button MajorActivity.java activity_major.java - Search bar refactor code and write better comments - Eclipse Junit test case
Jeehyun Yoon	Programmer	Data - windsor.csv - western.csv - wilfried_laurier.csv - york.csv - refactoring csv files MainActivity.java - code for functionalities in first page (link to official websites, etc)

3) Executive Summary / Description of Implementation

Name: UniFind

Short description: Android app that implements searching/sorting algorithms to find and rank universities and their programs in Ontario.

Abstract: The university application process is undoubtedly one of the most stressful times of highschool years. Most students spend hours trying to find related information and statistics on the university/program they desire and try to conceptualize their relative choices based on the information they find off the web. This process can be highly inefficient and time consuming, so UniFind aims to organize this process into a more effective and manageable one by providing algorithmic solutions in sorting Universities and their programs by categories such as ranking, admission average, tuitions, using related datasets.

Example of the dataset:

	A	B	C	D	E	F	G	H	I
1	Programs	Admission_Averag	Local_Tuition	International_Tuition	Requirements	Co-op	Target_Enrolment	Supplementary_Application	
2	Arts & Science	88	6042	32370	ENG4U,MHF4U/MCV4U	No	70	No	
3	Bachelor of Technology	78	7973	33690	ENG4U,MCV4U,SCH4U,SPH4U	Yes	240	Yes	
4	Business	86	9354	38160	ENG4U,Two of MFH4U/MCV4U/MDM4U	Yes	800	No	
5	Chemical and Physical Sciences Gateway	82	6030	32370	ENG4U,MHF4U,MCV4U,SCH4U,SPH4U	Yes	125	No	
6	Computer Science	92	7997	34800	ENG4U,MCV4U,Two of SBI4U/SPH4U/SES4U/ICS4U/TEJ4M	Yes	50	No	
7	Economics	80	9354	38160	ENG4U,Two of MHF4U/MCV4U/MDM4U	Yes	125	No	
8	Engineering	87	10080	41370	ENG4U,MCV4U,SCH4U,SPH4U	Yes	900	Yes	
9	Environmental & Earth Sciences Gateway	82	6030	32370	ENG4U,One of MHF4U/MCV4U/One of SBI4U/SCH4U, One of MHF4U/MCV4U/SBI4U/SPH4U	Yes	100	No	
10	Health and Society	80	6030	30774	ENG4U	Yes	60	No	
11	Health Sciences	90	6042	32340	ENG4U,One of MHF4U/MCV4U/MDM4U,SBI4U,SCH4U,One non-math/non-tech 4U/M	No	240	Yes	
12	Humanities	78	6030	30750	ENG4U	Yes	480	No	
13	Integrated Biomedical Engineering & Health Sciences	90	10080	40470	ENG4U,MCV4U,SBI4U,SCH4U,SPH4U	Yes	140	Yes	
14	Integrated Business & Humanities	88	10080	39750	ENG4U,MCV4U,MDM4U	Yes	60	Yes	
15	Integrated Science	87	10080	40470	ENG4U,MFH4U,MCV4U,Two of SBI4U/SCH4U/SPH4U	No	60	Yes	
16	Kinesiology	87	6030	32370	ENG4U,MCV4U,SBI4U	No	200	No	
17	Life Sciences	87	6030	32370	ENG4U,One of MHF4U/MCV4U,SBI4U,One of MHF4U/MCV4U/SCH4U/SPH4U	Yes	1000	No	
18	Mathematics & Statistics Gateway	85	6030	32370	ENG4U,MHF4U,MCV4U	Yes	300	No	
19	Medical Radiation Sciences	85	6030	32370	ENG4U,MFH4U,MCV4U,SBI4U,SCH4U	No	110	No	
20	Music	78	6030	30774	ENG4U	Yes	20	Yes	
21	Nursing	85	6030	35610	ENG4U,One of MHF4U/MCV4U/MDM4U	No	120	Yes	
22	Social Sciences	78	6030	30750	ENG4U	Yes	855	No	
23	Studio Art	78	6030	30774	ENG4U	Yes	25	Yes	
24									
25									

Description of classes:

Classes	Description
University	As you can see from the example of our dataset above, each university's program information is organized into csv files. This class serves to save all its program information of a university. In MajorSortActivity.java, a csv reader function reads all these data and organize the data into a field variable called universities of type University[]. Each item in that list will be, for example, mcmaster university object, waterloo university object, … Which contains information of all the program it has. **State Variables:** **String name** – name of university **ArrayList<Program>** program – list of all programs in that university **Int ranking** – qs world ranking of that university **Methods:** **Getters and setters** **Public Program getProgram(String name)** - return Program object given that program's name in string - return null if program doesn't exist in that university
Program	The csv columns are organized as: program name, admission average, local tuition, international tuition, requirements, coop, target enrolment, and supplementary application. The program saves all this information within that class as its state variables. **State Variables:** **String name** – name of program in a particular university **Int admission_average** – admission average in integer **Int local_tuition** – tuition for domestic students **Int international_tuition** – tuition for international students **Boolean coop** – whether that program has coop or not

	String target enrolment – number of students accepted into that program, made it string because we know for sure no numeric comparison is made with this variable **Boolean supplementary_applicaion** – whether you need to provide supplementary application for that program **<u>Methods:</u>** **Getters/Setters**
MainActivity	**[FIRST PAGE]** **Linked to:** activity_main.xml Displays information of all universities in Ontario in alphabetical order. Also has a button on the bottom to navigate to the next page (MajorActivity.java) **State variables:** **Button –** button object to navigate to next page **Methods:** **Public void openMajor()** - Change to next page on button click
MajorActivity	**[SECOND PAGE]** **Linked to:** activity_major.xml Has buttons that represent the program you wish to search for in the list of all universities in Ontario. For example, when the button “Computer Science” is pressed, the page navigates to the next page (MajorSortActivitiy.java) with the data “Computer Science” transferred from current page to next page. Using this information about the major user selected, the app generates the third page that displays information about all computer science programs in Ontario, where user can further choose options to filter/search/sort them in the category they want.
MajorSortActivity	**[THIRD PAGE]** **Linked to:** activity_major_sort.xml, list_group.xml, list_item.xml

	This is the class(page) where algorithms come in to play. In this class, using csv reader methods, we obtain the information from our ~20 csv files from raw directory and organize them into Program/University objects. We save this whole data into ArrayList<University> as a state variable. Upon creation, this page displays the major the user chose in the previous page according to "ranking" in decreasing order. To customize sorting/filtering options, there are the following functionalities: 1) **105 Switch** = turn on to focus on international tuition (when sorting based on tuition or apply max bound on tuition) **2) Coop Switch** = turn on to filter programs for only programs that have coop **3) Tuition Upperbound Text Field:** input the upper bound on tuition of that program (if input is 10000, the app will filter for programs whose tuition (domestic/international depending on 105 switch status) is less than or equal to the input. **4) Rank Based On Spinner:** select one of "ranking", "admission average", "tuition". For example, when ranking is chosen, the app will sort the programs depending on the university's world ranking, in increasing order. For admission average, it will sort in decreasing order, and for tuition, it will sort in increasing order **5) Refresh button –** the changes made by the above functionalities will hold effect only after the refresh button is pressed. When it's pressed, the app will filter/sort through the programs again and generate a new list view that corresponds to the filter/sorting result **Field Variables:**

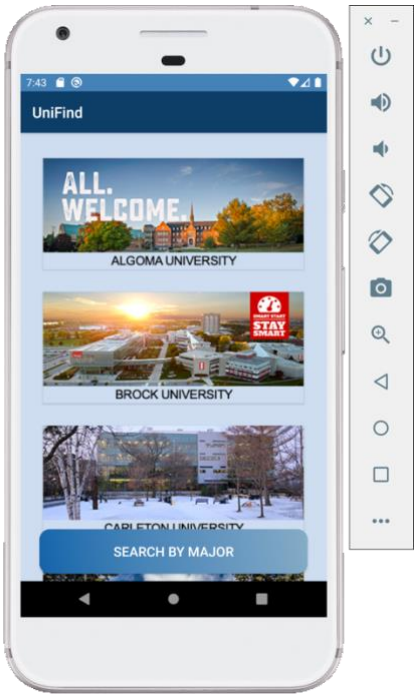
	String major – stores information about the selected major from previous page (MajorActivity.java) when respective button was pressed **Int tuitionUpperBound –** user input into tuition upper bound text field **Boolean coop –** status of coop switch **Boolean isInternational –** status of 105 switch **String sortCategory –** user choice from Rank Based On Spinner **ExpandableListView expandableListView –** expandable view object that'll save information about the sorted programs and display them **List<String> listGroup –** university names in that list **HashMap<String,List<String>> listItem –** program information for each university's program in listGroup **MainAdaptor adaptor –** object that'll sinc listView data to display **<u>Methods:</u>** **public void resetSetting(Boolean coop, Boolean internationalStatus, int textFieldInput, String sortCategory)** - Resets the field variable to the status of the switches/spinner/textfield when "refresh" button is pressed. Then, it calls getListViewData() to retrieve the new set of information of the list view with the new constraints. **Public void getListViewData()** - gets the new University[] in sorted order using the current states (isInternational,coop,tuitionUpperBound,category) and convert that to new list_group and list_item in order to display the new result as a list view **Public void getData()** - Runs getUniversityData() and getUniversityRanking()

	- Data is retrieved here and not model because you can't access raw directory in a non-android java files (need content) **Public void getUniversityData()** - Reads the csv files from raw directory and saves the data retrieved in the university field variable of the class **Public void getUniversityRanking()** - Read from qs_world_ranking.csv and update the ranking field of each university in the universities field variable
Model	**String[] fileNames** – names of csv files so that we can loop through and find the files in csv reader method **ArrayList<University> universities** – stores all data found by the csvs for later use **HashMap<String,String> universityNameConversion** – converts name like "mcmaster" into "McMaster University" **HashMap<String,String> rankingList –** list of universities and their world rankings **Public boolean isNumeric(String str)** - Returns true if string is a number **public String removeQuotations(String s)** - Return the result of removing the quotations from a string (used to remove the quotation from requirement section of Program object **Public Boolean yesNoConversion(String s)** - Converts "Yes" to true and "No" to false - Used for converting coop/supplementary_application columns of csv **Public String booleanToString(Boolean b)** - Converts "true" to "Yes" and "false" to "no" - Used to convert the field variables coop/supApp in Program object to display in the list view nicely **public University[] getProgramBasedOnCategory(String programName, String category)**

	- Returns the sorted array of universities of given program based on given category in desired ordering. **Public String[] objToString(Object[] obj)** - side function needed for getUniversityRanking() - converts Object[] to String[] **Public Integer[] objToInt(Object[] obj)** - side function needed for getUniversityRanking() - Converts Object[] to Integer[] **Public String[] sortDecreasingOrder(String[] universityNames, int[] values)** - Returns sorted result of universityNames based on values in decreasing order - Uses bubble sort **Public String[] sortIncreasingOrder(String[] universityNames, int[] values)** - Returns sorted result of universityNames based on values in increasing order - Use bubble sort **Public void exchString(String[] a, int I, int j)** - Exchanges element at indices I and j in array a - Needed for sortIncreadingOrder/sortDecreasingOrder **Public void exchInt(int[] a, int I, int j)** - Exchanges elements at indices I and j in array a - Needed for sortIncreasingOrder/sortDecreasingOrder **Public University getUniversity(String name)** - Return University object from universities field variable given its string name
MainAdaptor	Customized adaptor object that links list_group, list_item from MajorSortActivity.java and display as list view -

Pictures of the pages:

We also have a video demo for our app.

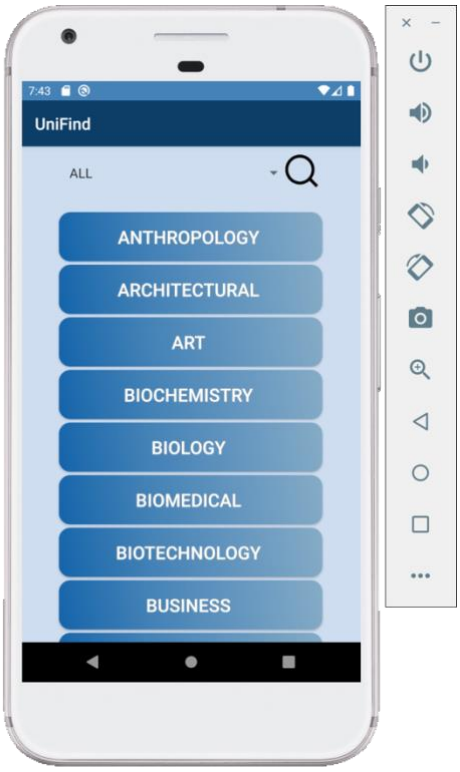


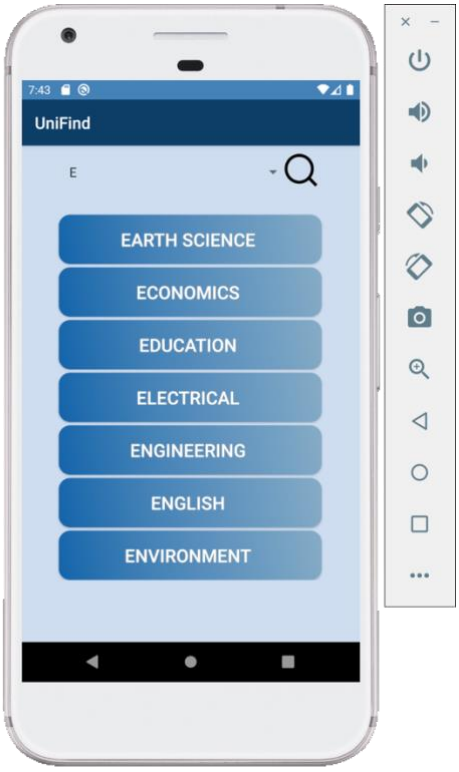
1) **MainActivity.java**

   This page is the main page, which is shown when starting the app. It shows the pictures and names of the universities in Ontario. After clicking the pictures, it should jump to the official website of that university. We can jump to the second page by clicking the "SEARCH BY MAJOR" button.

2) **MajorActivity.java**

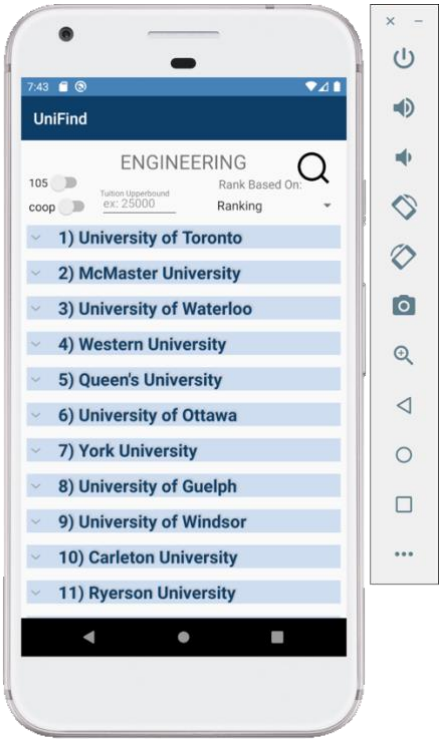
   This page shows all the popular majors, sorting by alphabetic. There is a spinner bar on the top, by selecting the letters in the spinner and clicking the search button, it shows the majors starting with that letter. After clicking the buttons, it jumps to university page.

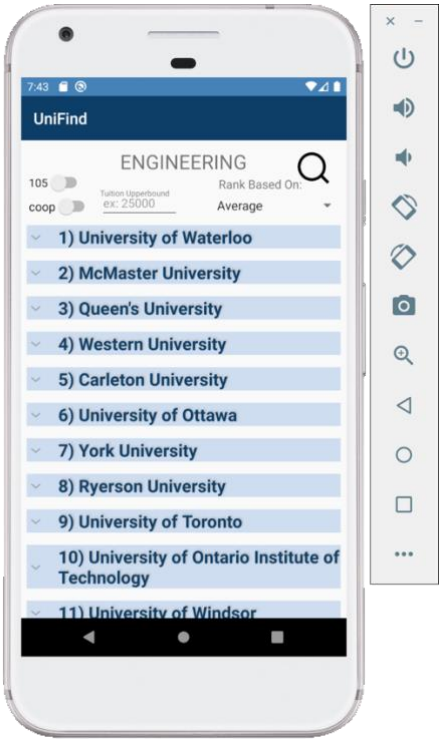


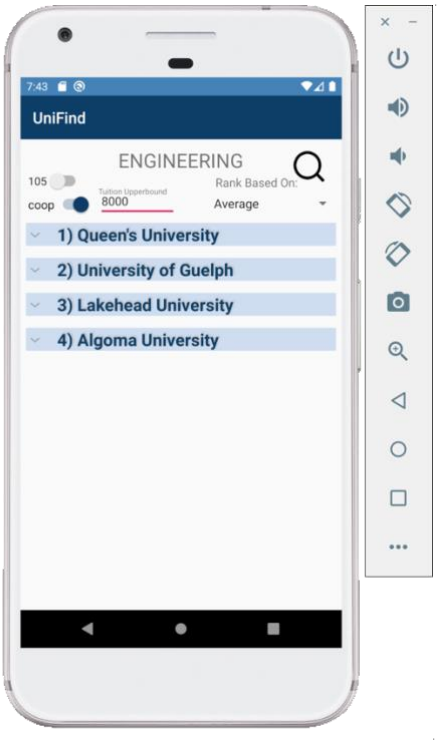


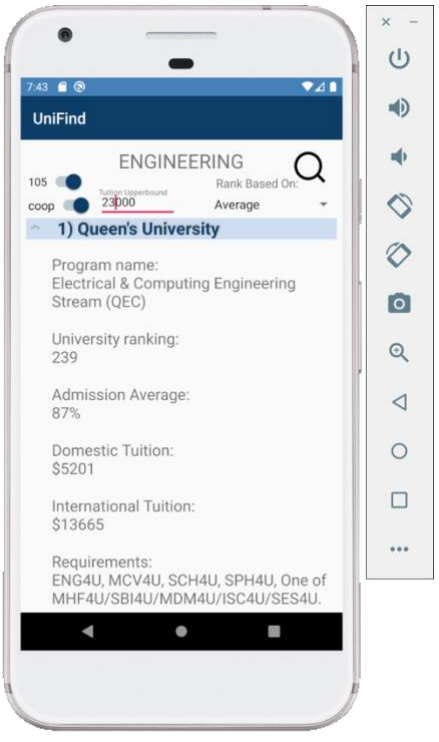
3) MajorSortActivity.java

This page shows the universities with the selected major, whose further information can be shown by clicking the name of universities. On the left side, there are two switches: 105 and coop. By turning on 105, the tuition would switch to the one for international student, by turning on 105, it only shows the universities that offer coop. There is an input text part for inputting the highest tuition that the user can bear. Also, they can choose to sort the universities by either ranking, admission average or tuition.



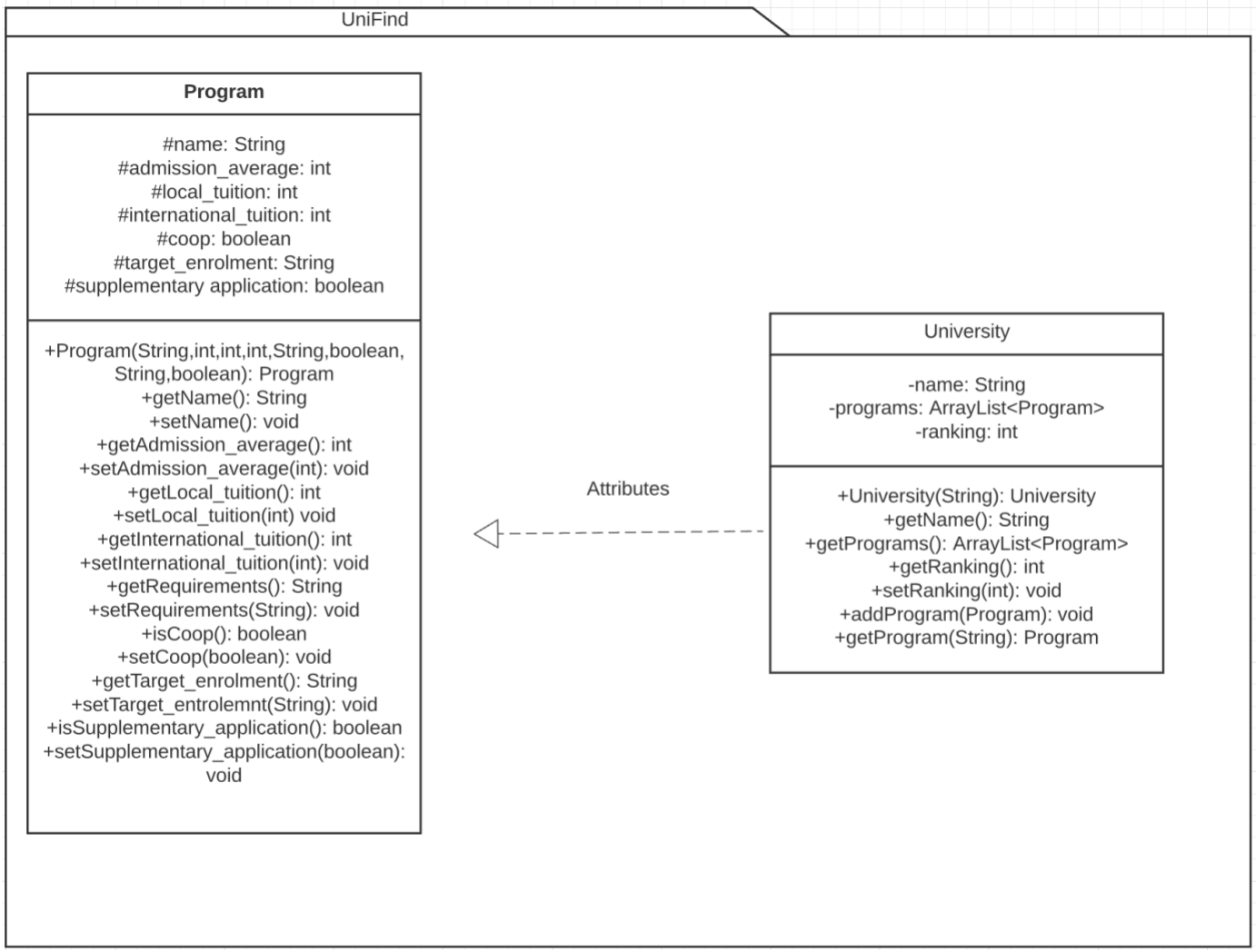




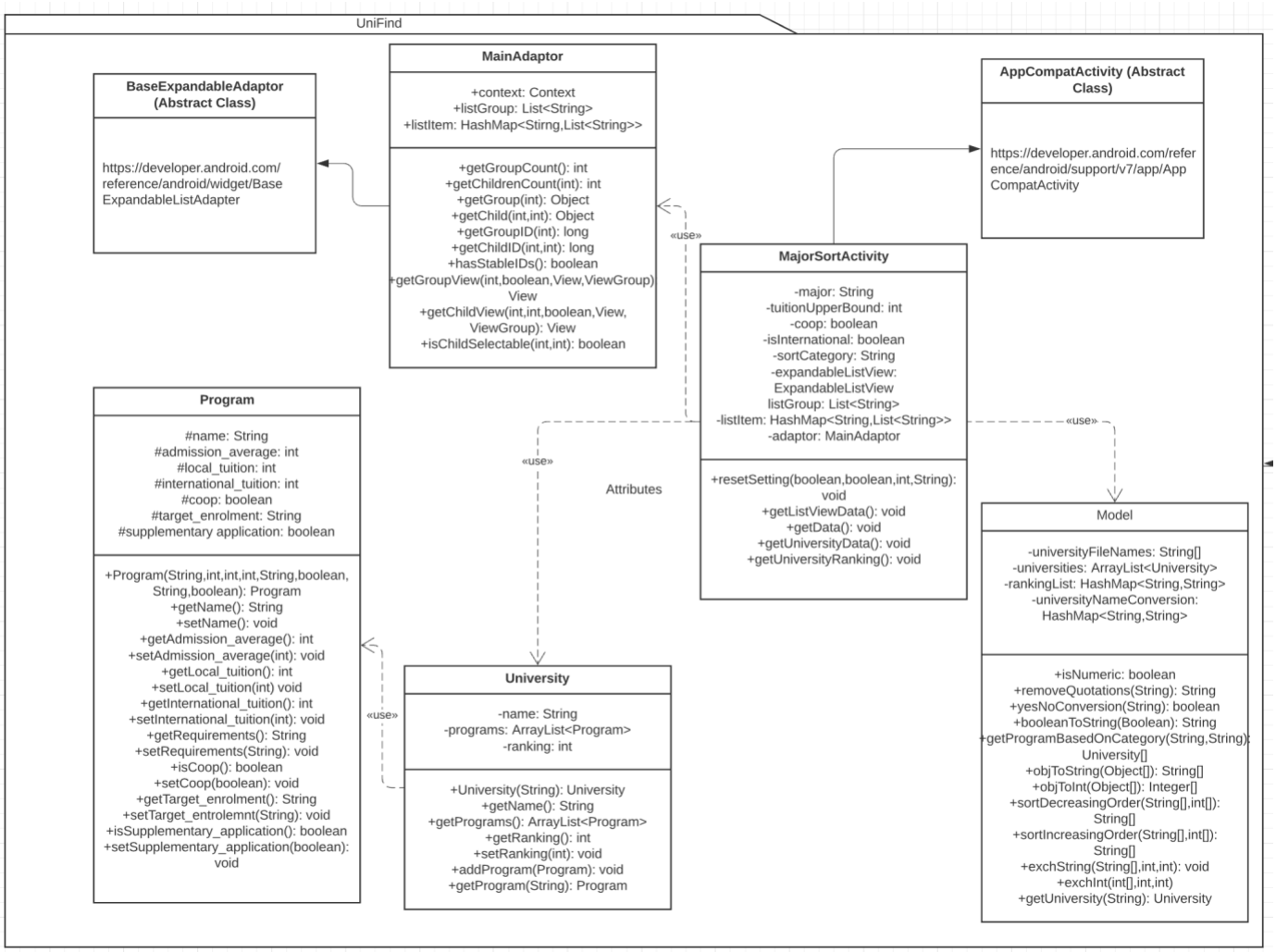


Two UML state machine diagrams for two most interesting classes in your implementation:

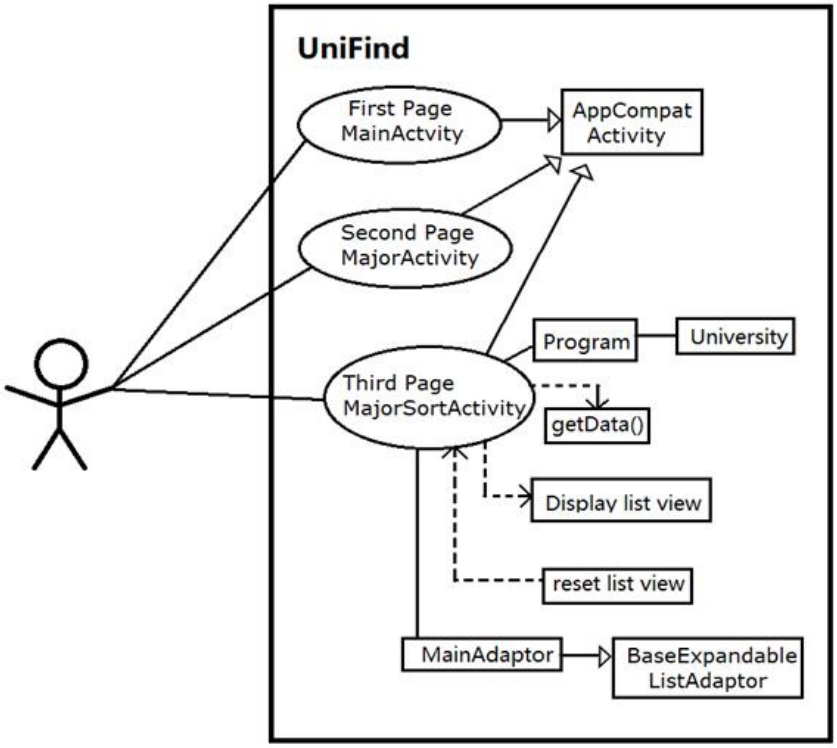
1) University



2) MajorSortActivity.java



4) A View of Uses Relationships



5) Traceback to Requirements in Each Class Interface

In our requirement specification, we said the domain was for highschool students mainly. We think that the end result of our projects suits its purpose well, it's a very easy to use and concise application that lets students search through programs in Ontario very efficiently at once glance. Important informations like world ranking, highschool course requirements, admission average, coop, target enrolment, supplementary application are included in the expandable list view, which makes it easy for students to explore through programs with ease.

For functional requirements, on our requirement specification document, we said the first page displays list of universities in Ontario by QS ranking, and when a user selects one university, it'll give them general information about that university. Also, there would be a button to navigate to "rank by subjects" page. On our final product, we list the universities in alphabetical order, and when we click the university it directs user to the official website, but everything else is the same. We decided to make this change because we only wanted to make the app so that data is accessed only in MajorSortActivity.java for code cleanliness and efficiency.

On second page, we said it will categorize university majors by their faculty. Each major will be a button to navigate to the third page, with input keyword major = "selectedButton.text". The end product of our app does the exact same job. We didn't organize the buttons by faculty, but we have multiple buttons where if selected, it navigates to third page with the passed data that corresponds to the button's text. For example, if "computer science" button is pressed, the app will go to the next page with the data "computer science" transferred from MajorActivity.java to MajorSortActivity.java.

On third page, we said we will display university rankings by the major that the user inputs initially in second page. Then the user will be able to select a bunch of filter options (tuition, employment rate, etc…) to customize the rankings. When user selects one of the university in the list ,it shows related information about that program such as admission average and target enrolment. Again, our final app meets this requirement. It displays a list of universities that has the selected program in meaningful order, which can be customized by the user. We have 3 categories this list can be sorted – (1) ranking, (2) admission average, (3) tuition.

For non-functional requirements, here are the following results:

(1) **Reliability:** the database collected for our program is exclusively from the official university websites (in the current year), so it's guaranteed that the information in this app is 100% reliable for students to use

(2) **Availability:** on requirement specification, we said we must use relative path instead of absolute path to make sure the csv files are available. We met this requirement by saving the csv files in raw directory within the android project and address it by using relative pathing

(3) **Security:** we don't have any backend system that collects user data so the users from this app is safe from all potential security problems

(4) **Safety:** the data in the software is never changed by user. In requirement specification, we said since this app is an information application, credibility of the resources is the key factor that will be evaluated. If data is easily modified, the users will doubt the safety of the application. So we said we should monitor the dataset continuously and prevent access to data. In our app, there is no functionality where user can directly modify the data, so our dataset is safe from whatever changes/modifications/maluses from users

(5) **Performance:** we previously said efficiency and accessibility are the most vital parts of this software. To be specific, we said we should provide accurate information in a short time to customers. The internal qualities are determined by which algorithm we choose in order to achieve best efficiency and accessibility. We chose bubble sort to implement our sorting algorithm because it has a complexity of O(n), while most of other algorithms are more complex because they perform the entire sorting process on the set. For filter functions, we tried to minimize the uses of nested for loops to avoid the increase of time complexities.

(6) **Human-computer Interface Issues:** for most part, we believe that our app is easy to navigate and use. In the second page (MajorActivity.java), not only are there individual buttons, but we also implemented a search bar to make the user interface experience easier and more efficient. Also we think the UI design for the third page (MajorSortActivity.java) is straight forward to use. So for now, we believe there's no notable human-computer interface issues

(7) **Operating constraints:** Since our application is a simple search/sort/filter app, it doesn't require a complicated operation method. However, we said there will be an operation constraint in the searching system. For example,

in the tuition upper bound text field, we made it so that user can only input numbers, but if for some reason the user inputs something like “28,332.11” the operation will terminate with error because this value cannot be converted directly into an integer. Of course, we tried to minimize this error by making methods and adding exception clauses to check for possible errors

(8) **Physical constrains:** in requirement specification, we said there will be a barrier constraint. For example, when user search in the search bar with incorrect spelling, the program might not give any output as search result or suggest alternative option to customers. This is true for our final product, and its assumed that user will input a valid search keyword in the search bar in second page (MajorActivity.java)

(9) Portability Issues: Since our software is mobile application, specifically for android, it will not work on computers that doesn’t have android studio installed, or on apple devices/tablets. Portability is kind of an issue for our case because the requirement for this project says we need to submit an eclipse project that can compile and run. However, we can’t compile and run android project in eclipse…

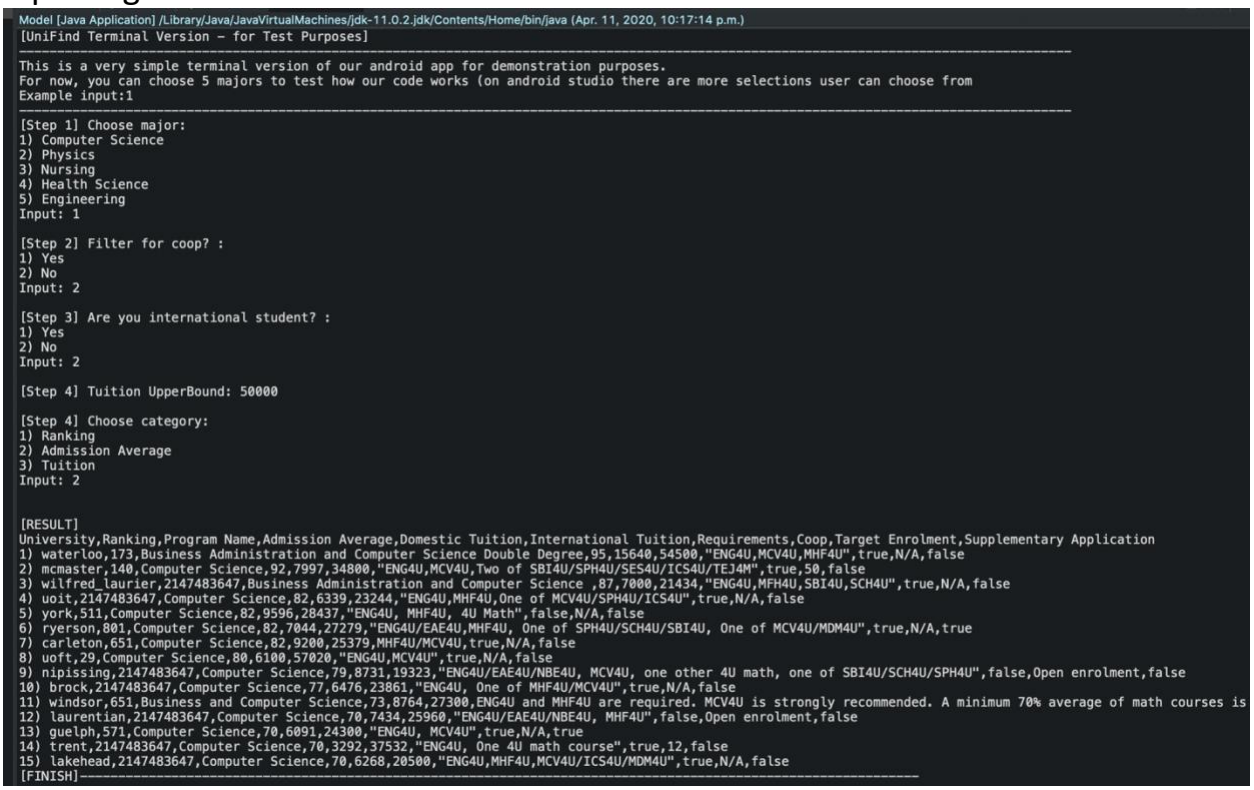
For requirements on development and maintenance process, we tried to be with consistent with private/protected, getter/setters, and method names for ease of debugging later. We also tried to modularize everything to individual methods so that if there’s any error we could quickly detect where the source of problem was. In terms of testing, we use unit testing by adding the main functions to an eclipse project and write Junit test for that. Also, our code ca be run by running it on virtual device (simulator) and using the Log functionality to print values on console.

6) Internal Review/Evaluation of Design

It feels like there's too much going on in MajorSortActivity.java at once. I feel like it could've been nicer if we could modularize the local functions maybe into another class like Utilities.java and kept only what's necessary for android activity manipulation in that class alone.

Another problem worth mentioning is that when we are searching for specific programs in the universities, what we did was we checked if, for example, the program name contains the word "Computer Science". However, this has a problem because, for example in waterloo, it filters out "Business Administration and Computer Science Double Degree" instead of the intended "Computer Science" program.

Other than that, I think our team did a pretty good job in modularizing the data into Programs and Universities with meaningful state variables so that it made the rest of the functions to
work very logically and efficiently.

7) Testing

For testing purpose, we create a terminal app and a Junit test. A screen shoot of the terminal app running version is shown below, and a video demo of the android studio emulator is included in Readme.md. By running it on terminal, it shows the buttons in app as different choice, and user can interact with it by inputting the choice.