Team Members:
Julian Tan
Connor Alvin
Michelle Jakab

Initial decisions

We chose to use Java 11 for programming since we have all used the language before and because of the availability of a vast number of libraries. Furthermore, separating each line of the students file into separate objects made intuitive sense, and Java, being an OO language, fit that intuition. Julian used atom, Connor worked on the unix servers, and I (Michelle) used IntelliJ IDEA.

Notes on selected internal architecture

As mentioned above, we decided to take an Object Oriented approach to the assignment. We parsed the provided file and created a list of objects (specifically Student objects) and used stream filtering to find the data for which the user was searching.

Task Log:

Task	Student(s)	Start Time	End Time	Total Hours
Strategising	Julian Tan Connor Alvin Michelle Jakab	4/3/2019	4/5/2019	1 hour
Parsing the given file	Julian Tan	4/3/2019	4/5/2019	30 min
Command line and Input	Connor Alvin	4/3/2019	4/5/2019	30 min
Filtering and Output	Michelle Jakab	4/5/2019	4/9/2019	2 hours
	Julian Tan	4/5/2019	4/9/2019	1 hour
Testing	Connor Alvin	4/5/2019	4/5/2019	2 hours
Writeup	Michelle Jakab Julian Tan	4/5/2019	4/10/2019	1 hour

Notes on Testing

As shown in the task log above, Connor did the testing. It took about 1 hour to write the tests, around 15 bugs were found, and it took about 1 hour to clean it all up and get it working properly.

Final Notes

An input command that has the start of a command but too many or too few arguments will print to the screen, “Invalid input.” Otherwise, if no command is input, if there is a typo in the command, or if the search yields no results, nothing of note prints. Commands are case sensitive, and, as stated in the README file, we named our main java file schoolSearch.java.