# Criterion B:

## Designs:

### Decomposition/Top down diagram:

This diagram displays the flow of operation between pages and how the user to navigate to reach each page in the application. This diagram is used to determine which pages are required and how they will be linked to one another.

Wireframe designs:

The following diagrams are a draft layout of my GUI. The purpose of this diagram is to assist in the development and layout of my GUI and is used to determine the components present in each page. Additionally, due to the high volume of pages, it is beneficial to be able to understand how links are created between pages.

Graphical user interface, text, application

Description automatically generated Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedA picture containing text

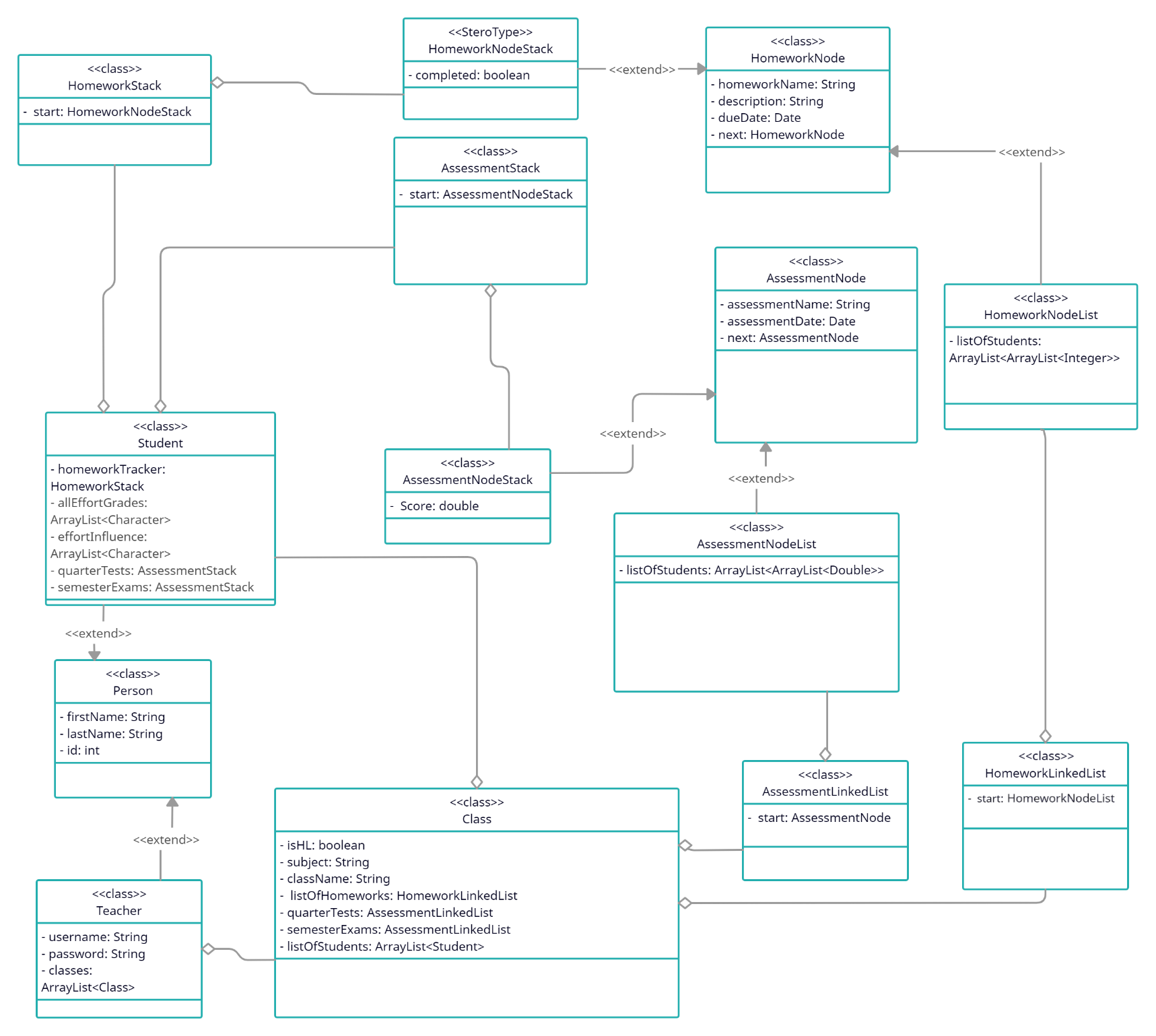
Description automatically generatedGraphical user interface, text, application, chat or text message

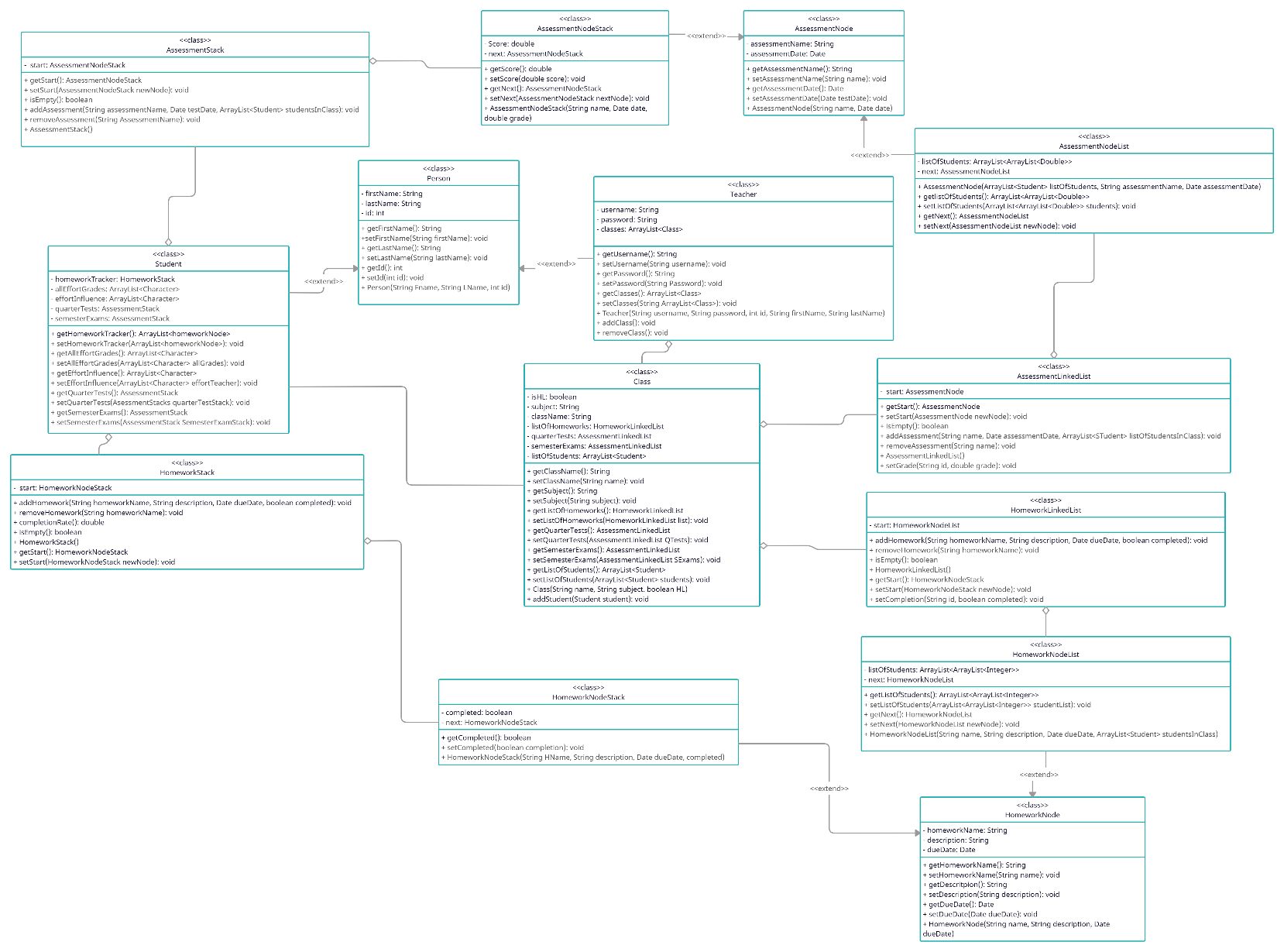
Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generated

UML Class diagrams:

The following UML class diagram is used to determine the different classes that will needed to develop the functionality of my application and to store various types of data. The following is a first brainstorm of the class diagrams I believe I will need, and I will change them as my IA develops.

The following is my second version of my UML diagrams, with all the methods I believe I will need to create for each class. Additionally, I have decided to move the next node within the HomeworkNode class to be specific to each HomeworkNodeList and HomeworkNodeStack. I have also done the same with the next reference for the AssessmentNodes. This was upon the realisation that the data type of these nodes should be specific to the stack or Linked List node as each stack and Linked List node have different attributes even though they inherit from the primary HomeworkNode/AssessmentNode. For a more clear view, please refer to UMLDiagramWithMethods.png in the documentation folder.

### Data Dictionary:

This is a list of the variables, their type and a short description of their intended use. This data dictionary is provided for increased comprehension of how the system functions and what the roles of each attribute are.

**Class: Person**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| firstName | String | private | This is the last name of the person |
| lastName | int | private | This is the first name of the person |
| id | int | private | This is the ID assigned to the person by our school |

**Class: Teacher**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| username | String | private | This is the username for the teacher |
| password | String | private | This is the password for the teacher |
| classes | ArrayList<Class> | private | This is the classes the teacher teaches |

**Class: Class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| className | String | private | This is the name of the class |
| subject | String | private | This is the subject of the class |
| isHL | boolean | private | Whether the class is higher level |
| listOfHomeworks | HomeworkLinkedList | private | This is the list of homeworks associated with this class |
| quarterTests | AssessmentLinkedList | private | This is the list of quarter tests associated with this class |
| semesterExams | AssessmentLinkedList | private | This is the list of semester exams associated with this class |
| listOfStudents | ArrayList<Student> | private | This is the list of students in this class |

**Class: Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| homeworkTracker | HomeworkStack | private | This is a stack that synchronizes with the homeworks added to the class and stores whether or not the student completed each homework. This is synchronized on a per student basis for the effort grade calculation. |
| quarterTests | AssessmentStack | private | This a stack that synchronizes with the quarter assessments added to the class and stores the grade achieved by the student on each one. This is synchronized on a per student basis for the effort grade calculation. |
| semesterExams | AssessmentStack | private | This a stack that synchronizes with the semester exams added to the class and stores the grade achieved by the student on each one. This is synchronized on a per student basis for the effort grade calculation. |
| effortInfluence | ArrayList<Character> | private | This is an arraylist of all the effort grade influences set by the teacher for each effort grade attained |
| allEffortGrades | ArrayList<Character> | private | This is an arraylist of all previous effort grades attained by the student |

**Class: AssessmentLinkedList**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| start | AssessmentNodeList | private | This is the start of the list of assessments |

**Class: AssessmentNodeList**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| listOfStudents | ArrayList<ArrayList<Double>> | private | This is a 2D arraylist of the ID of each student in the class and the score attained on the specific assessment. Column 1 is IDs and column 2 is the double representing the score of the student. This is stored on a per class basis to be displayed as such to the teacher in the GUI. This is synchronized with the respective assessment stack of each student. |
| next | AssessmentNodeList | private | This is a reference to the next node, a self-referential call. |

**Class: AssessmentNode**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| assessmentName | String | private | This is the name of the assessment |
| assessmentDate | Date | private | This is the date of the assessment |

**Class: HomeworkLinkedList**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| start | HomeworkNodeList | private | This is the start of the list of homeworks |

**Class: HomeworkNodeList**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| listOfStudents | ArrayList<ArrayList<Integer>> | private | This is a 2D arraylist of the ID of each student in the class and an integer value of 0 or 1 representing a boolean for whether the homework was not completed or completed respectively. Column 1 is IDs and column 2 is the integer representing the completion Boolean. This is stored on a per class basis to be displayed as such to the teacher in the GUI. This is synchronized with the homework stack of each student for a per student basis of storage for later effort grade calculation. |

**Class: HomeworkNode**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| homeworkName | String | private | This is the name of the homework |
| description | String | private | This is the description of the homework |
| dueDate | Date | private | This is the date the homework is due |

**Class: AssessmentStack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| start | HomeworkNodeStack | private | This is the start of the stack of assessments |

**Class: AssessmentNodeStack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| score | double | private | This is a double value indicating the score of the current student on the assessment the current node is |

**Class: HomeworkStack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| start | HomeworkNodeStack | private | This is the start of the stack of homeworks |

**Class: HomeworkNodeStack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Data Type** | **Modifier** | **Description** |
| completed | boolean | private | This is a boolean indicating whether the homework of the current node is completed by the student or not |

### Use case diagram:

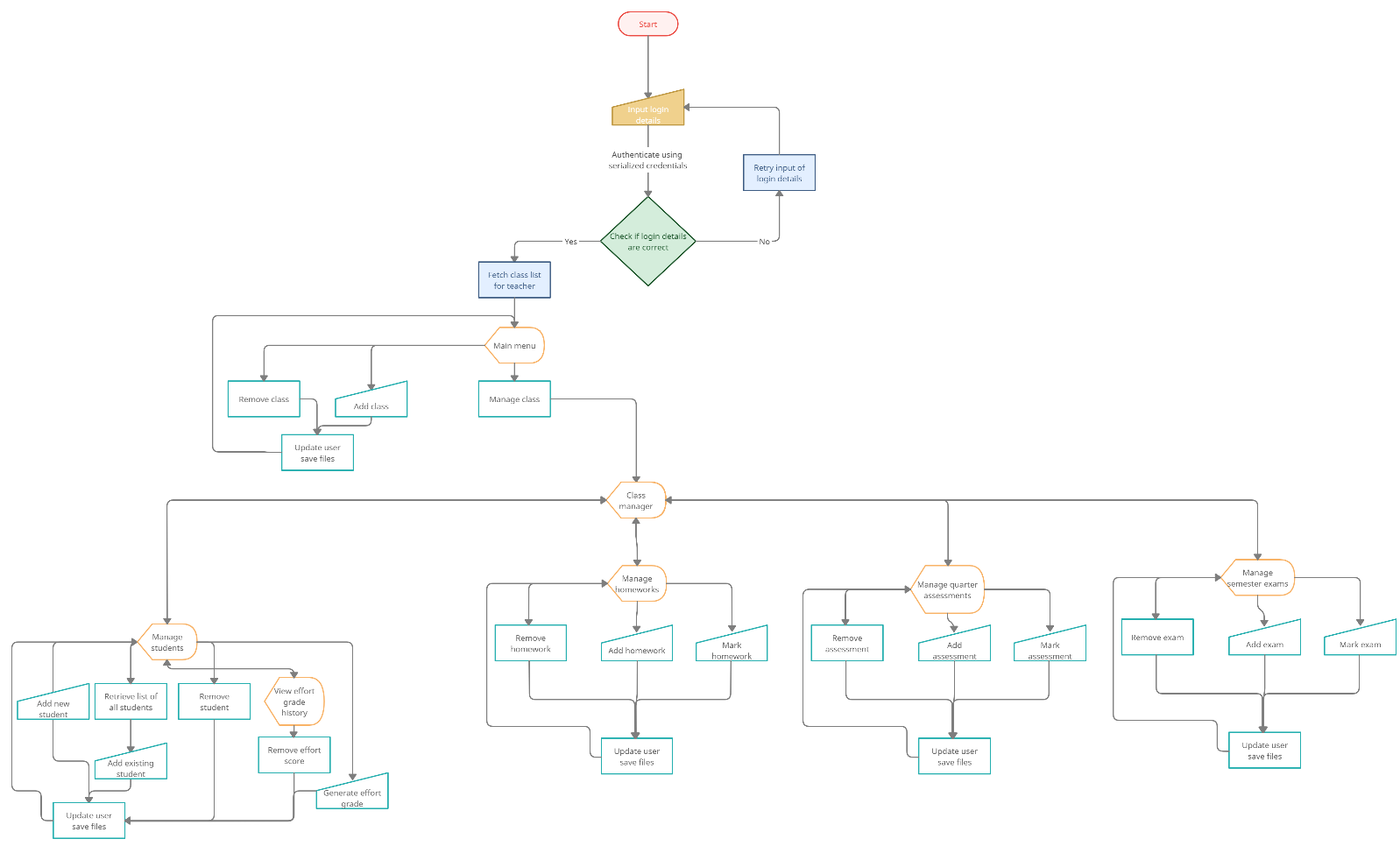
This diagram depicts the interactions between the user and the system and demonstrates the actions that they may opt to take.

Diagram, schematic

Description automatically generated

### Application flowchart:

This flowchart is used to compute how the various components of my application will function together in order to successfully generate effort grades and how the required data will be saved and loaded. For a more clear view, please refer to ApplicationFlowchart.png in the documentation folder.



Pseudocode:

This is the pseudocode for the search algorithm that will be used, to access elements within the 2D ArrayList within each HomeworkNodeList and AssessmentNodeList. This algorithm allows for students to be efficiently searched to be removed from each of the 2D ArrayLists within each node in each HomeworkLinkedList. Given that each time if a student is added to the system, their id is added to 3 linked lists each time, assuming each with x, y and z entries, then the resulting addition of a student generates 3z+3y+3z entries to the ArrayLists within the linked lists, notwithstanding the addition of a student object to the class. As a result, in order to make the process of removing all this information when a student is removed from the system more efficient, a binary search algorithm is used rather than a sequential search, with time complexity O(log(n)) compared to O(n).

A binary search functions by repeatedly dividing the portion of the given ArrayList that may contain the element being searched for in two, until the element is found. By first dividing the ArrayList in two and determining in which half the element will be present in (as the ArrayList would be sorted), the algorithm may find the element being searched for. This process of dividing in half is repeated until the element may be found.

|  |  |
| --- | --- |
| ARRAY<><> | 2D ArrayList to be sorted |
| LOW | Starting index to search from |
| HIGH | Last index to search to |
| LOCATION | Location of the element if found |
| FOUND | Boolean of whether the element was found |

**Binary Search algorithm**

1. set FOUND to false
2. set LOW to 0
3. set HIGH to LENGTH
4. set LOCATION to -1
5. loop while not FOUND and (LOW < HIGH)
6. set INDEX to (LOW + HIGH) / 2
7. if KEY = ARRAY<INDEX><0> then
8. set FOUND to true
9. set LOCATION to INDEX

exit loop

1. else
2. if KEY < ARRAY<INDEX><0> then
3. set HIGH to INDEX – 1
4. else
5. set LOW to INDEX + 1
6. end if
7. end if
8. end loop

If the element is not found, the location is set to be -1.

However, for the above algorithm to work, the 2D ArrayList has to be sorted. For this, I will utilise quicksort. The average time complexity to use quicksort is O(nlog(n)). Due to its performance being the best amongst all sorting algorithms (Wang), given the average case for most inputs, I chose this sorting method in order to ensure to use the fastest sorting method possible.

Quicksort functions by selecting a pivot element in the ArrayList and partitioning other elements into 2 arrays based on their value compared to the pivot. This process is repeated as these subarrays are further sorted recursively (Sehgal). In order to implement the sort function, a helper function, to partition the given ArrayList is created using the following pseudocode.

|  |  |
| --- | --- |
| ARRAY<><> | 2D ArrayList to be sorted |
| LOW | Starting index |
| HIGH | Last index of the ArrayList |
| PIVOT | Index of element below which all smaller elements are placed and all larger elements are placed in the indices above |

**Partition algorithm**

1. function Partition(ARRAY<><>, LOW, HIGH):
2. set PIVOT = ARRAY<HIGH><0>;
3. set I = (LOW-1)
4. loop for J for HIGH-1
5. if ARRAY<J><0> < PIVOT then
6. I++;
7. swap ARRAY<I> and ARRAY<J>
8. end if
9. end for
10. swap ARRAY<I + 1> and ARRAY<high>
11. return <I + 1>

In order to perform the quicksort, this algorithm will need to be called recursively upon the sub arrays fo the array created. This is achieved with the following pseudocode:

|  |  |
| --- | --- |
| ARRAY<><> | 2D ArrayList to be sorted |
| LOW | Starting index of ARRAY<><> |
| HIGH | Last index of the ArrayList |
| PIVOT | Index of element below which all smaller elements are placed and all larger elements are placed in the indices above, given from the partition algorithm |

**QuickSort algorithm**

1. function QuickSort(ARRAY<><>, LOW, HIGH):
2. if LOW < HIGH then
3. set PIVOT = Partition(ARRAY<><>, LOW, HIGH)
4. QuickSort(ARRAY<>, LOW, PIVOT - 1)
5. QuickSort(ARRAY<>,PIVOT + 1, HIGH)
6. end if

The quicksort algorithm calls upon the partition method to calculate the pivot index, and to sort all elements into subarrays of values smaller than the pivot being placed to the indices smaller than the pivot index and similarly for the values greater than the pivot. The quicksort is then called recursively on these subarrays to eventually sort the entire ArrayList.

## Test plan:

**Class: Person**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed**  **X** |
| firstName: String | Length check (length>2) | Normal | "Ben" | Accept value | **🗸** |
| Extreme | "P" | Reject value and input again | **🗸** |
| Format Check (Alphabet only) | Normal | "Ben" | Accept value | **🗸** |
| Abnormal | "Peter123512" | Reject value and input again | **🗸** |
| lastName: String | Length check (length>2) | Normal | "Parker" | Accept value | **🗸** |
| Extreme | "P" | Reject value and input again | **🗸** |
| Format Check (Alphabet only) | Normal | "Parker" | Accept value | **🗸** |
| Abnormal | "Peter123512" | Reject value and input again | **🗸** |
| Id: int | Range check (0<id<99999) | Normal | 128357 | Accept value | **🗸** |
| Extreme | -9 | Reject value and input again | **🗸** |
| Format Check (Integer only) | Normal | 128357 | Accept value | **🗸** |
| Abnormal | "Ninety Nine" | Reject value and input again | **🗸** |

**Class: Class**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| className: String | Length check (length>2) | Normal | "Mathematics 01" | Accept value | **🗸** |
| Extreme | "M" | Reject value and input again | **🗸** |
| Format Check (Alphabet and numbers) | Normal | "Mathematics 01" | Accept value | **🗸** |
| Abnormal | #$%^&!@# | Reject value and input again | **🗸** |
| Subject: String | Length check (length>2) | Normal | "Physics" | Accept value | **🗸** |
| Extreme | "P" | Reject value and input again | **🗸** |
| Format Check (Alphabets only) | Normal | "Physics" | Accept value | **🗸** |
| Abnormal | "Physics 09" | Reject value and input again | **🗸** |

**Class: HomeworkNode**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| homeworkName: String | Length check (length>2) | Normal | "Calculus" | Accept value | **🗸** |
| Extreme | "C" | Reject value and input again | **🗸** |
| Format Check (Alphabet with or without numbers) | Normal | "Calculus 01" | Accept value | **🗸** |
| Extreme | 125123 | Reject value and input again | **🗸** |
| Abnormal | \*&^&^&\* | Reject value and input again | **🗸** |
| description: String | Length check (length>10) | Normal | "Physics" + … (11 characters) | Accept value | **🗸** |
| Extreme | "P" | Reject value and input again | **🗸** |
| Format Check (Alphabets with or without numbers) | Normal | "Physics" + … (11 alphabets/numbers) | Accept value | **🗸** |
| Extreme | 125123 | Reject value and input again | **🗸** |
| Abnormal | !#$^@#$ | Reject value and input again | **🗸** |
| dueDate (day component): Date | Format Check | Normal | 14 | Accept value | **🗸** |
| Abnormal | "fourteen" | Reject value and input again | **🗸** |
| Range Check (1-30/31) | Extreme | 58 | Reject value and input again | **🗸** |
| dueDate (month component): Date | Format Check | Normal | 14 | Accept value | **🗸** |
| Abnormal | "fourteen" | Reject value and input again | **🗸** |
| Range Check (1-12) | Extreme | 58 | Reject value and input again | **🗸** |
| dueDate (year component): Date | Format Check | Normal | 2022 | Accept value | **🗸** |
| Abnormal | "twenty twenty two" | Reject value and input again | **🗸** |
| Range Check (2022-2050) | Extreme | 58 | Reject value and input again | **🗸** |

**Class: AssessmentNode**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| assessmentName: String | Length check (length>2) | Normal | "Calculus" | Accept value | **🗸** |
| Extreme | "C" | Reject value and input again | **🗸** |
| Format Check (Alphabet with or without numbers) | Normal | "Calculus 01" | Accept value | **🗸** |
| Extreme | 125123 | Reject value and input again | **🗸** |
| Abnormal | \*&^&^&\* | Reject value and input again | **🗸** |
| assessmentDate (day component): Date | Format Check | Normal | 14 | Accept value | **🗸** |
| Abnormal | "fourteen" | Reject value and input again | **🗸** |
| Range Check (1-30/31) | Extreme | 58 | Reject value and input again | **🗸** |
| assessmentDate (month component): Date | Format Check | Normal | 14 | Accept value | **🗸** |
| Abnormal | "fourteen" | Reject value and input again | **🗸** |
| Range Check (1-12) | Extreme | 58 | Reject value and input again | **🗸** |
| assessmentDate (year component): Date | Format Check | Normal | 2022 | Accept value | **🗸** |
| Abnormal | "twenty twenty two" | Reject value and input again | **🗸** |
| Range Check (2022-2050) | Extreme | 58 | Reject value and input again | **🗸** |

**Class: Student**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| effortInfluence: ArrayList<Character> | Length check (length=1) and Format Check (Alphabet) | Normal | "E" | Accept value | **🗸** |
| Extreme | "Excellent" | Reject value and input again | **🗸** |
| Abnormal | 1 | Reject value and input again | **🗸** |

**Class: Teacher**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| firstName: String | Presence check (check if the data has been found) | Normal | Found | Accept value | **🗸** |
| Abnormal | Not Found | Reject value and input again | **🗸** |
| lastName: String | Normal | Found | Accept value | **🗸** |
| Abnormal | Not Found | Reject value and input again | **🗸** |
| id: int | Normal | Found | Accept value | **🗸** |
| Abnormal | Not Found | Reject value and input again | **🗸** |
| username: String | Presence check (check if the data has been found) | Normal | Found | Accept value | **🗸** |
| Abnormal | Not Found | Reject value and input again | **🗸** |
| Consistency check (chack if username matches with data saved on disk) | Normal | Match | Accept value | **🗸** |
| Abnormal | Does not match | Reject value and input again | **🗸** |
| password: String | Presence check (check if the data has been found) | Normal | Found | Accept value | **🗸** |
| Abnormal | Not Found | Reject value and input again | **🗸** |
| Consistency check (chack if username matches with data saved on disk) | Normal | Match | Accept value | **🗸** |
| Abnormal | Does not match | Reject value and input again | **🗸** |

**Class: AssesmentNodeList**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variable or attribute** | **Type of validation** | **Example** | | **Expected Output** | **Test Passed 🗸 Test Failed X** |
| listOfStudents: ArrayList<ArrayList<Double>> | Range Check (1.0-7.9) | Normal | 7.8 | Accept value | **🗸** |
| Extreme | 89.1 | Reject value and input again | **🗸** |