TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BACS2063 Data Structures and Algorithms

ASSIGNMENT 202401

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University of Management and Technology's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Student Name	Student ID	Prog / Tut.Grp	Signature
Tan Lock Kwan	2213638	RDS2S2G3	Tan
Loh Jia Shou	2213596	RDS2S2G3	LOH
Goh Boon Xiang	2214263	RDS2S2G3	goh
Lim Hoi Yau	2214081	RDS2S2G3	LHY
Tan Hoong Guan	2213632	RDS2S2G3	THG

Note: The submission date and time will be according to the timestamp recorded in Google Classroom for Assignment Submission.

Leader Name:

Tan Lock Kwan

Loh Jia Shou

Goh Boon Xiang

Lim Hoi Yau

Tan Hoong Guan

Table of Contents

A. TEAM REPORT

1. Abstract Data Type (ADT)

1.1 ADT Specification

ADT ArrayList

An array list is a collection of distinct elements of a type T.

boolean add(T newEntry);

Description	:	Adds a **newEntry** to a collection.
Precondition	:	The **newEntry** is not null.
Postcondition	:	The **newEntry** is added to the end of the list if there is enough capacity, otherwise, the capacity of the list is increased and then the newEntry is added.
Returns	:	**True** if the addition is successful.

boolean add(int newPosition, T newEntry);

Description	:	Inserts a specified element at a specified position in the list, shifting subsequent elements to the right.
Precondition	:	The **newPosition** is within the bounds of the list and the **newEntry** is not null.
Postcondition	:	The **newEntry** is inserted at newPosition, and subsequent elements are shifted accordingly.
Returns	:	**true** if the insertions successful, **false** otherwise.

T remove(int givenPosition);

Description	:	Removes the element at the specified position in this list.
Precondition	:	The **givenPosition** is within the bounds of the list.
Postcondition	:	The element at the specified position is removed from the list. Subsequent elements are shifted to the left to fill the gap created by the removal.
Returns	:	The element that was removed from the list.

void clear();

Description : Removes all of the elements from this list.

Postcondition : The list is left empty after this operation.

boolean replace(int givenPosition, T newEntry);

Description : Replace the element at the specified position with the specified element.

Precondition : The **givenPosition** is within the bounds of the list, and the **newEntry** is not null.

Postcondition : The element at the specified position is replaced with the **newEntry**.

Returns : **true** if the replacement is successful, **false** otherwise.

T getEntry(int givenPosition);

Description : Returns the element at the specified position in this list.

Precondition : The **givenPosition** is within the bounds of the list

Returns : The element at the specified position in the list.

boolean contains(T anEntry);

Description : Checks whether the specified element is present in this list.

Returns : **True** if the list contains the specified element, **false** otherwise.

int indexOf(T element);

Description : Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

Returns : The index of the first occurrence of the specified element in the list, or -1 if the element is not found.

int getNumberOfEntries();

Description : Returns the number of elements currently stored in this list.

Returns : The number of elements currently stored in the list.

boolean isEmpty();

Description : Check whether the list is empty, or contains no elements.

Returns : **true** if the list contains no elements, **false** otherwise.

boolean isFull();

Description : Check whether this list is full, indicating it has reached its maximum capacity.

Returns : **true** if the list has reached its maximum capacity, **false** otherwise.

Iterator<T> iterator();

Description : Returns an iterator over the elements in this list.

Postcondition : An iterator over the elements in the list.

void sortByAscending(Function<T, Comparable<?>> keyExtractor);

Description : Sorts the elements of this list in ascending order based on the provided **keyExtracto**r.

Postcondition : The elements of the list are reordered in ascending order according to the specified **keyExtractor**.

void sortByDescending(Function<T, Comparable<?>> keyExtractor);

Description : Sorts the elements of this list in descending order based on the provided **keyExtracto**r.

Postcondition : The elements of the list are reordered in descending order according to the specified **keyExtractor**.

1.2 ADT Implementation

Java Interface

```
package adt;
import java.util.Iterator;
import java.util.function.Function;

public interface ListInterface<T> {

  /**
   * Task: Adds a new entry to the end of the list. Entries currently in the
   * list are unaffected. The list's size is increased by 1.
   *
   * @param newEntry the object to be added as a new entry
   * @return true if the addition is successful, or false if the list is full
   */
  public boolean add(T newEntry);

  /**
   * Task: Adds a new entry at a specified position within the list. Entries
   * originally at and above the specified position are at the next higher
   * position within the list. The list's size is increased by 1.
   *
   * @param newPosition an integer that specifies the desired position of the
   * new entry
   * @param newEntry the object to be added as a new entry
   * @return true if the addition is successful, or false if either the list
   * is full, newPosition < 1, or
   *         newPosition > getNumberOfEntries()+1
   */
  public boolean add(int newPosition, T newEntry);

  /**
```

```
 * Task: Removes the entry at a given position from the list. Entries
 * originally at positions higher than the given position are at the next
 * lower position within the list, and the list's size is decreased by 1.
 *
 * @param givenPosition an integer that indicates the position of the entry
 * to be removed
 * @return a reference to the removed entry or null, if either the list was
 * empty, givenPosition < 1, or
 *          givenPosition > getNumberOfEntries()
 */
public T remove(int givenPosition);

/**
 * Task: Removes all entries from the list.
 */
public void clear();

/**
 * Task: Replaces the entry at the specified position in the list with the
 * given entry.
 *
 * @param givenPosition an integer that indicates the position of the entry
 * to be replaced
 * @param newEntry the object that will replace the entry at the position
 * givenPosition
 * @return true if the replacement occurs, or false if either the list is
 * empty, givenPosition < 1, or givenPosition > getNumberOfEntries()
 */
public boolean replace(int givenPosition, T newEntry);

/**
 * Task: Retrieves the entry at a given position in the list.
 *
```

```
 * @param givenPosition an integer that indicates the position of the
 * desired entry
 * @return a reference to the indicated entry or null, if either the list is
 * empty, givenPosition < 1, or givenPosition > getNumberOfEntries()
 */
public T getEntry(int givenPosition);

/**
 * Task: Checks if the list contains the specified element.
 *
 * @param anEntry the object that is the desired entry
 * @return true if the list contains anEntry, or false if not
 */
public boolean contains(T anEntry);

/**
 * Task: Retrieves the index of the first occurrence of the specified
 * element in the list.
 *
 * @param element the element whose index is to be found
 * @return the index of the first occurrence of the specified element in the
 * list,
 */
public int indexOf(T element);

/**
 * Task: Retrieves the number of entries in the list.
 *
 * @return the integer number of entries currently in the list
 */
public int getNumberOfEntries();

/**
```

```
 * Task: Checks if the list is empty.
 *
 * @return true if the list is empty, or false if not
 */
public boolean isEmpty();

/**
 * Task: Checks if the list is full.
 *
 * @return true if the list is full, or false if not
 */
public boolean isFull();

/**
 *
 * Task: Returns an iterator over the elements in this list in proper
 * sequence.
 *
 * @return an iterator over the elements in this list
 */
Iterator<T> iterator();

/**
 *
 * Task: Sorts the elements of the list in ascending order based on the
 * values extracted using the provided keyExtractor function.
 *
 * @param keyExtractor a function that extracts a comparable key from each
 * element
 * @param <T> the type of elements in the list
 */
public void sortByAscending(Function<T, Comparable<?>> keyExtractor);
```

```
    /**
     *
     * Task: Sorts the elements of the list in descending order based on the
     * values extracted using the provided keyExtractor function.
     *
     * @param keyExtractor a function that extracts a comparable key from each
     * element
     * @param <T> the type of elements in the list
     */
    public void sortByDescending(Function<T, Comparable<?>> keyExtractor);

}
```

Java Implementation Class

```
package adt;
import java.io.Serializable;
import java.util.Iterator;
import java.util.function.Function;

public class ArrayList<T> implements ListInterface<T>, Serializable {

    private T[] array;
    private int numberOfEntries;
    private static final int DEFAULT_CAPACITY_SIZE = 10;

    public ArrayList() {
        this(DEFAULT_CAPACITY_SIZE);
    }

    public ArrayList(int initialCapacity) {
        numberOfEntries = 0;
        array = (T[]) new Object[initialCapacity];
```

```
}

@Override
public boolean add(T newEntry) {
   if (isArrayFull()) {
      doubleArray();
   }

   array[numberOfEntries] = newEntry;
   numberOfEntries++;
   return true;
}

@Override
public boolean add(int newPosition, T newEntry) {
   boolean isSuccessful = true;

   if ((newPosition >= 1) && (newPosition <= numberOfEntries + 1)) {
      if (isArrayFull()) {
         doubleArray();
      }
      makeRoom(newPosition);
      array[newPosition - 1] = newEntry;
      numberOfEntries++;
   } else {
      isSuccessful = false;
   }

   return isSuccessful;
}

@Override
public T remove(int givenPosition) {
```

```
    T result = null;

    if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {
        result = array[givenPosition - 1];

        if (givenPosition < numberOfEntries) {
            removeGap(givenPosition);
        }

        numberOfEntries--;
    }

    return result;
}

@Override
public void clear() {
    numberOfEntries = 0;
}

@Override
public boolean replace(int givenPosition, T newEntry) {
    boolean isSuccessful = true;

    if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {
        array[givenPosition - 1] = newEntry;
    } else {
        isSuccessful = false;
    }

    return isSuccessful;
}
```

```
@Override
public T getEntry(int givenPosition) {
   T result = null;

   if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {
      result = array[givenPosition - 1];
   }

   return result;
}

@Override
public boolean contains(T anEntry) {
   boolean found = false;
   for (int index = 0; !found && (index < numberOfEntries); index++) {
      if (anEntry.equals(array[index])) {
         found = true;
      }
   }
   return found;
}

@Override
public int indexOf(T element) {
   for (int index = 0; index < numberOfEntries; index++) {
      if (element.equals(array[index])) {
         return index; // Return the index of the found element
      }
   }
   return -1; // Return -1 if the element is not found
}

@Override
```

```
public int getNumberOfEntries() {
   return numberOfEntries;
}

@Override
public boolean isEmpty() {
   return numberOfEntries == 0;
}

@Override
public boolean isFull() {
   return false;
}

private void doubleArray() {
   T[] oldArray = array;
   array = (T[]) new Object[oldArray.length * 2];
   for (int i = 0; i < oldArray.length; i++) {
      array[i] = oldArray[i];
   }
}

private boolean isArrayFull() {
   return numberOfEntries == array.length;
}

@Override
public String toString() {
   String outputStr = "";
   for (int index = 0; index < numberOfEntries; ++index) {
      outputStr += array[index] + "\n";
   }
```

```
    return outputStr;
}

/**
 * Task: Makes room for a new entry at newPosition. Precondition: 1 <=
 * newPosition <= numberOfEntries + 1; numberOfEntries is array's
 * numberOfEntries before addition.
 */
private void makeRoom(int newPosition) {
    int newIndex = newPosition - 1;
    int lastIndex = numberOfEntries - 1;

    // move each entry to next higher index, starting at end of
    // array and continuing until the entry at newIndex is moved
    for (int index = lastIndex; index >= newIndex; index--) {
        array[index + 1] = array[index];
    }
}

/**
 * Task: Shifts entries that are beyond the entry to be removed to the next
 * lower position. Precondition: array is not empty; 1 <= givenPosition <
 * numberOfEntries; numberOfEntries is array's numberOfEntries before
 * removal.
 */
private void removeGap(int givenPosition) {
    // move each entry to next lower position starting at entry after the
    // one removed and continuing until end of array
    int removedIndex = givenPosition - 1;
    int lastIndex = numberOfEntries - 1;

    for (int index = removedIndex; index < lastIndex; index++) {
        array[index] = array[index + 1];
```

```
    }
}

@Override
public Iterator<T> iterator() {
    return new ArrayListIterator();
}

// Inner class implementing Iterator interface
private class ArrayListIterator implements Iterator<T> {

    private int currentIndex = 0;

    @Override
    public boolean hasNext() {
        return currentIndex < numberOfEntries;
    }

    @Override
    public T next() {
        if (!hasNext()) {
            return null;
        }
        T element = array[currentIndex];
        currentIndex++;
        return element;
    }
}

@Override
public void sortByAscending(Function<T, Comparable<?>> keyExtractor) {
    quickSort(0, numberOfEntries - 1, keyExtractor);
}
```

```
    // Sort in descending order based on a particular variable
    @Override
    public void sortByDescending(Function<T, Comparable<?>> keyExtractor) {
        quickSortDescending(0, numberOfEntries - 1, keyExtractor);
    }

    // Quicksort algorithm for ascending order
    private void quickSort(int low, int high, Function<T, Comparable<?>> keyExtractor) {
        if (low < high) {
            int pi = partition(low, high, keyExtractor);
            quickSort(low, pi - 1, keyExtractor);
            quickSort(pi + 1, high, keyExtractor);
        }
    }

    // Quicksort algorithm for descending order
    private void quickSortDescending(int low, int high, Function<T, Comparable<?>>
keyExtractor) {
        if (low < high) {
            int pi = partitionDescending(low, high, keyExtractor);
            quickSortDescending(low, pi - 1, keyExtractor);
            quickSortDescending(pi + 1, high, keyExtractor);
        }
    }

    // Partition for ascending order
    private int partition(int low, int high, Function<T, Comparable<?>> keyExtractor) {
        T pivot = array[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            Comparable<Object> keyJ = (Comparable<Object>) keyExtractor.apply(array[j]);
            Comparable<Object> pivotKey = (Comparable<Object>) keyExtractor.apply(pivot);
```

```
            if (keyJ.compareTo(pivotKey) <= 0) {
                i++;
                swap(i, j);
            }
        }
        swap(i + 1, high);
        return i + 1;
    }

    // Partition for descending order
    private int partitionDescending(int low, int high, Function<T, Comparable<?>> keyExtractor)
{
        T pivot = array[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            Comparable<Object> keyJ = (Comparable<Object>) keyExtractor.apply(array[j]);
            Comparable<Object> pivotKey = (Comparable<Object>) keyExtractor.apply(pivot);
            if (keyJ.compareTo(pivotKey) > 0) {
                i++;
                swap(i, j);
            }
        }
        swap(i + 1, high);
        return i + 1;
    }

    // Utility method to swap elements in the array
    private void swap(int i, int j) {
        T temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}
```

B. INDIVIDUAL REPORTS

2. Use of ADTs

Name	Student ID	Prog / Tut.Grp	Signature
Tan Lock Kwan	2213638	RDS2S2G3	lockkwan

Subsystem : Student Registration Management Subsystem

Source codes for Control classes.

```
package control;

import adt.*;
import boundary.StudentManagementUI;
import boundary.CourseManagementUI;
import dao.*;
import entity.Student;
import entity.Enrollment;
import entity.Course;
import entity.courseProgramme;
import entity.Programme;
import utility.MessageUI;
import java.util.Scanner;
import java.util.function.Function;

/**
 *
 * @author Tan Lock Kwan
 */
public class StudentManagement {

    private ListInterface<Student> studentList = new ArrayList<>();
    private ListInterface<Enrollment> enrollmentList = new ArrayList<>();
    private ListInterface<Course> courseList = new ArrayList<>();
    private ListInterface<Student> sortedList = new ArrayList();
    private ListInterface<courseProgramme> courseProgrammeList = new ArrayList<>();
    private ListInterface<Programme> programmeList = new ArrayList<>();
    private StudentDAO StudentDAO = new StudentDAO();
    private EnrollmentDAO EnrollmentDAO = new EnrollmentDAO();
    private StudentInitializer StudentInitializer = new StudentInitializer();
    private EnrollmentInitializer EnrollmentInitializer = new EnrollmentInitializer();
    private StudentManagementUI studentUI = new StudentManagementUI();
```

```
    private CourseManagementUI courseUI = new CourseManagementUI();
    private CourseDAO CourseDAO = new CourseDAO();
    private CourseInitializer CourseInitializer = new CourseInitializer();
    private courseProgrammeInitializer courseProgrammeInitializer = new
courseProgrammeInitializer();
    private courseProgrammeDAO courseProgrammeDAO = new courseProgrammeDAO();
    private ProgrammeInitializer ProgrammeInitializer = new ProgrammeInitializer();
    private ProgrammeDAO ProgrammeDAO = new ProgrammeDAO();
    private ListInterface<Student> studentFilteredList = new ArrayList<>();
    private ListInterface<Student> studentSortedList = new ArrayList<>();
    static final double PRICEPERCREDITHOUR = 200.00;

    Scanner scanner = new Scanner(System.in);

    public StudentManagement() {
        studentList = StudentDAO.retrieveFromFile();
        enrollmentList = EnrollmentDAO.retrieveFromFile();
        courseList = CourseDAO.retrieveFromFile();
        courseProgrammeList = courseProgrammeDAO.retrieveFromFile();
        programmeList = ProgrammeDAO.retrieveFromFile();
        /*
        studentList = StudentInitializer.initializeStudents();
        StudentDAO.saveToFile(studentList);
        enrollmentList = EnrollmentInitializer.initializeEnrollment();
        EnrollmentDAO.saveToFile(enrollmentList);
        courseList = CourseInitializer.initializeCourses();
        CourseDAO.saveToFile(courseList);
        courseProgrammeList = courseProgrammeInitializer.initializeCourseProgramme();
        courseProgrammeDAO.saveToFile(courseProgrammeList);
        ProgrammeDAO.saveToFile(programmeList);
        */
    }


    public void runStudentManagement() {
        int choice = 0;
        do {
            choice = studentUI.getMenuChoice();
            switch (choice) {
                case 0:
                    MessageUI.displayExitMessage();
                    break;
                case 1:
                    displayStudentList();
                    break;
                case 2:
```

```
                addNewStudent();
                break;
            case 3:
                removeStudent();
                break;
            case 4:
                editStudent();
                break;
            case 5:
                searchStudent();
                break;
            case 6:
                addStudentToCourse();
                break;
            case 7:
                removeStudentFromCourse();
                break;
            case 8:
                calculateCourseFee();
                break;
            case 9:
                filterStudents();
                break;
            case 10:
                sortStudents();
                break;
            case 11:
                displayReport();
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
    } while (choice != 0);
}

//Display Student List
public void displayStudentList() {
    displayStudents();
    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

//Add new student
public void addNewStudent() {
    String newStudentAgain = "Yes";
    do {
```

```
            boolean check;
            int nextStudentIDNum = getStudentIDNum() + 1;
            String newStduentID = "S" + nextStudentIDNum ;
            String newStudentName = studentUI.inputNewStudentName();
            int newStudentAge = studentUI.inputNewStudentAge();
            String newStudentGender = studentUI.inputNewStudentGender();
            String newStudentPhoneNo = studentUI.inputNewStudentPhoneNo();
            String newStudentEmail = studentUI.inputNewStudentEmail();
            displayProgramme();
            String newStudentProgrammeCode;
            do {
               newStudentProgrammeCode = studentUI.inputNewStudentProgrammeCode();
               if (!findProgramme(newStudentProgrammeCode)) {
                  System.out.println("Invalid Programme Code! Please input a valid Programme
Code.\n");
                  check = false;
               } else {
                  check = true;
               }
            } while (!check);
            Student newStudent = (Student) studentUI.inputNewStudentDetails(newStudentName,
newStudentAge, newStudentGender, newStudentPhoneNo, newStudentEmail,
newStudentProgrammeCode);
            studentList.add(newStudent);
            StudentDAO.saveToFile(studentList);
            newStudentAgain = studentUI.inputNewStudentAgain();
         } while ("Yes".equals(newStudentAgain));
      }

      //Remove Student
      public void removeStudent() {
         displayStudents();
         if (studentList.isEmpty()) {
            System.out.println("Student list is empty.");
         } else {
            String removeStudentID = studentUI.inputRemoveStudentID();
            int indexToRemove = -1;
            for (int i = 1; i < studentList.getNumberOfEntries() + 1; i++) {
               Student student = studentList.getEntry(i); // Retrieve Student object at index i
               if (student.getStudentID().equals(removeStudentID)) {
                  indexToRemove = i;
                  break; // Once found, exit the loop
               }
            }
            if (indexToRemove != -1) {
               studentList.remove(indexToRemove);
```

```
            StudentDAO.saveToFile(studentList);
            System.out.println("\nStudent with Student ID: " + removeStudentID + " removed
successfully.");
        } else {
            System.out.println("\nStudent with Student ID: " + removeStudentID + " not found in
the list.");
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }
  }

  //Edit Student Details
  public void editStudent() {
    displayStudents();
    boolean studentCheck = true;
    do {
      String editStudentID = studentUI.inputEditStudentID();

      int indexToEdit = -1;
      for (int i = 1; i < studentList.getNumberOfEntries() + 1; i++) {
        Student student = studentList.getEntry(i);
        if (student.getStudentID().equals(editStudentID)) {
          indexToEdit = i;
          studentCheck = true;
          break;
        }
      }
      if (indexToEdit != -1) {
        int typeEditChoice = 0;
        Student editStudent = studentList.getEntry(indexToEdit);
        do {
          typeEditChoice = studentUI.getTypeEditChoice();
          switch (typeEditChoice) {
            case 0:
              MessageUI.displayExitMessage();
              break;
            case 1:
              String oldStudentName = editStudent.getName();
              System.out.println("Old Name: " + oldStudentName);
              editStudent.setName(studentUI.inputNewName());
              System.out.print("\nStudent's name modified successfully!\n");
              break;
            case 2:
              int oldStudentAge = editStudent.getAge();
              System.out.println("Old Age: " + oldStudentAge);
```

```
                editStudent.setAge(studentUI.inputNewAge());
                System.out.print("\nStudent's age modified successfully!\n");
                break;

            case 3:
                String oldStudentGender = editStudent.getGender();
                System.out.println("Old Gender: " + oldStudentGender);
                editStudent.setGender(studentUI.inputNewGender());
                System.out.print("\nStudent's gender modified successfully!\n");
                break;
            case 4:
                String oldStudentPhoneNo = editStudent.getPhoneNo();
                System.out.println("Old Phone Number: " + oldStudentPhoneNo);
                editStudent.setPhoneNo(studentUI.inputNewPhoneNo());
                System.out.print("\nStudent's hone number modified successfully!\n");
                break;

            case 5:
                String oldStudentEmail = editStudent.getEmail();
                System.out.println("Old Email: " + oldStudentEmail);
                editStudent.setEmail(studentUI.inputNewEmail());
                System.out.print("\nStudent's email modified successfully!\n");
                break;

            case 6:
                displayProgramme();
                String StudentProgrammeCode;
                boolean check;
                do {
                    String oldStudentProgrammeCode = editStudent.getProgrammeCode();
                    System.out.println("Old ProgrammeCode: " + oldStudentProgrammeCode);
                    StudentProgrammeCode = studentUI.inputNewProgrammeCode();
                    if (!findProgramme(StudentProgrammeCode)) {
                        System.out.println("Invalid Programme Code! Please input a valid
Programme Code.\n");
                        check = false;
                    } else {
                        check = true;
                    }
                } while (!check);
                editStudent.setProgrammeCode(StudentProgrammeCode);
                System.out.print("\nStudent's programe ID modified successfully!\n");
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
```

```
            } while (typeEditChoice != 0);
            studentList.replace(indexToEdit, editStudent);
            StudentDAO.saveToFile(studentList);
        } else {
            studentCheck = false;
            System.out.print("Student with Student ID: " + editStudentID + " is not found in the
list. Please type a valid Student ID.\n");
        }
    } while (!studentCheck);
  }

  //Search Student for registered course
  public void searchStudent() {

    String searchStudentAgain = "Yes";

    do {
      String studentID;
      String courseCode;
      boolean check;
      displayStudents();
      do {
        studentID = studentUI.inputStudentID();
        if (!findStudent(studentID)) {
          System.out.println("Invalid Student ID! Please input a valid Student ID.\n");
          check = false;
        } else {
          check = true;
        }
      } while (!check);
      displayCourses();
      do {
        courseCode = studentUI.inputCourseCode();
        if (!findCourse(courseCode)) {
          System.out.println("Invalid Course Code! Please input a valid Course Code.\n");
          check = false;
        } else {
          check = true;
        }
      } while (!check);

      boolean enrollmentFound = false;
      String enrollmentStatus = null;

      for (int i = 1; i <= enrollmentList.getNumberOfEntries(); i++) {
        Enrollment enrollment = enrollmentList.getEntry(i); // Retrieve Enrollment object at
```

```
index i
            if (enrollment.getStudentID().equals(studentID) &&
enrollment.getCourseCode().equals(courseCode)) {
                enrollmentFound = true;
                enrollmentStatus = enrollment.getEnrollmentStatus();
                break;
            }
        }

        if (!enrollmentFound) {
            System.out.println("\nStudent with Student ID: " + studentID + " is not enrolled in " +
courseCode);
        } else {
            System.out.println("\nStudent with Student ID: " + studentID + " is enrolled in " +
courseCode
                    + " as " + enrollmentStatus);
        }
        searchStudentAgain = studentUI.inputSearchStudentAgain();
    } while ("Yes".equalsIgnoreCase(searchStudentAgain));

  }

  //Add student to the course
  public void addStudentToCourse() {
    String newAddStudentToCourseAgain = "Yes"; //if course added the programme id is not
same as student's programmeID, then error. And check got same or not
    do {
        boolean check = false;
        String studentID;
        String courseCode;
        Student student = null;
        courseProgramme courseProgramme;
        boolean programmeCheck = true;
        displayStudents();
        do {
            studentID = studentUI.inputStudentID();
            if (!findStudent(studentID)) {
                System.out.println("Invalid Student ID! Please input a valid Student ID.\n");
                check = false;
            } else {
                check = true;
            }
        } while (!check);
        displayCourses();
        do {
            courseCode = studentUI.inputCourseCode();
```

```
            if (!findCourse(courseCode)) {
                System.out.println("Invalid Course Code! Please input a valid Course Code.\n");
                check = false;
            } else {
                for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
                    student = studentList.getEntry(i); // Retrieve Student object at index i
                    if (student.getStudentID().equals(studentID)) {
                        break; // Once found, exit the loop
                    }
                }
                for (int i = 1; i <= courseProgrammeList.getNumberOfEntries(); i++) {
                    courseProgramme = courseProgrammeList.getEntry(i); // Retrieve Student object
at index i
                    if (courseProgramme.getCourseCode().equals(courseCode)) {
                        if
(courseProgramme.getProgrammeCode().equals(student.getProgrammeCode())) {
                            programmeCheck = true;
                            check = true;
                            break; // Once found, exit the loop
                        } else {
                            programmeCheck = false;
                            check = false;
                        }
                    }
                }
                if (!programmeCheck) {
                    System.out.println("This student is ineligible for course enrollment due to a
program code mismatch. "
                            + "Please provide a valid Course Code.");
                }
            }
        } while (!check);

        String enrollmentStatus = studentUI.inputEnrollmentStatus();
        for (int i = 1; i <= enrollmentList.getNumberOfEntries(); i++) {
            Enrollment enrollment = enrollmentList.getEntry(i); // Retrieve Enrollment object at
index i
            if (enrollment.getStudentID().equals(studentID) &&
enrollment.getCourseCode().equals(courseCode)) {
                System.out.println("\nStudent with Student ID: " + studentID + " ALREADY
enrolled in " + courseCode
                        + " as " + enrollmentStatus);
                enrollmentStatus = enrollment.getEnrollmentStatus();
            } else {
                Enrollment newStudentToCourse = (Enrollment) new Enrollment(studentID,
courseCode, enrollmentStatus);
```

```
                enrollmentList.add(newStudentToCourse);
                EnrollmentDAO.saveToFile(enrollmentList);
                System.out.print("\nStudent has been added to a new course successfully!\n");
                newAddStudentToCourseAgain = studentUI.inputAddStudentToCourseAgain();
                break;
            }
        }
    } while ("Yes".equals(newAddStudentToCourseAgain));
}

//Remove student from the courses
public void removeStudentFromCourse() {
    String enrollmentStudentID;
    String enrollmentCourseCode;
    boolean check;
    displayStudents();
    do {
        enrollmentStudentID = studentUI.inputStudentID();
        if (!findStudent(enrollmentStudentID)) {
            System.out.println("Invalid Student ID! Please input a valid Student ID.\n");
            check = false;
        } else {
            check = true;
        }
    } while (!check);
    displayCourses();
    do {
        enrollmentCourseCode = studentUI.inputRemoveEnrollmentCourseCode();
        if (!findCourse(enrollmentCourseCode)) {
            System.out.println("Invalid Course Code! Please input a valid Course Code.\n");
            check = false;
        } else {
            check = true;
        }
    } while (!check);
    String enrollmentStatus = studentUI.inputRemoveEnrollmentStatus();
    int indexToRemove = -1;
    for (int i = 1; i < enrollmentList.getNumberOfEntries() + 1; i++) {
        Enrollment enrollment = enrollmentList.getEntry(i); // Retrieve Student object at index i
        if (enrollment.getStudentID().equals(enrollmentStudentID) &&
enrollment.getCourseCode().equals(enrollmentCourseCode) &&
enrollment.getEnrollmentStatus().equals(enrollmentStatus)) {
            indexToRemove = i;
            break; // Once found, exit the loop
        }
    }
```

```
        if (indexToRemove != -1) {
            enrollmentList.remove(indexToRemove);
            EnrollmentDAO.saveToFile(enrollmentList);
            System.out.println("\nStudent with Student ID: " + enrollmentStudentID + " removed
successfully from "
                    + enrollmentCourseCode + " as " + enrollmentStatus);
        } else {
            System.out.println("\nThe student could not be removed from enrollment. No enrollment
found matching the provided student ID, course ID, and enrollment status.");
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    //Filter Student
    public void filterStudents() {
        int criteriaType = 0;
        do {
            String criteria = null;
            studentFilteredList.clear();
            criteriaType = studentUI.inputCriteriaType();
            switch (criteriaType) {
                case 0:
                    break;
                case 1:
                    //Filter Students by Age
                    criteria = "Age";
                    int minAge = studentUI.inputMinAge();
                    int maxAge = studentUI.inputMaxAge();
                    for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
                        Student student = studentList.getEntry(i);
                        if (student.getAge() <= maxAge && student.getAge() >= minAge) {
                            studentFilteredList.add(student);
                        }
                    }
                    break;
                case 2:
                    //Filter Students by Gender
                    criteria = "Gender";
                    String gender = studentUI.inputGender();
                    for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
                        Student student = studentList.getEntry(i);
                        if (student.getGender().equals(gender)) {
                            studentFilteredList.add(student);
                        }
                    }
```

```
                break;
            case 3:
                //Filter Students by ProgrammCode
                boolean check;
                displayProgramme();
                criteria = "Programme Code";
                String programmeCode;
                do {
                    programmeCode = studentUI.inputProgrammeCode();
                    if (!findProgramme(programmeCode)) {
                        System.out.println("Invalid Programme Code! Please input a valid Programme
Code.\n");
                        check = false;
                    } else {
                        check = true;
                    }
                } while (!check);
                for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
                    Student student = studentList.getEntry(i);
                    if (student.getProgrammeCode().equals(programmeCode)) {
                        studentFilteredList.add(student);
                    }
                }
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }


        if (studentFilteredList.isEmpty()) {
            if (criteria != null) {
                System.out.println("Unfortunately there are no students who meet your specified
criteria.");
            } else {
                System.out.println();
            }
        } else {
            System.out.println("\nList of Filtered Students based on " + criteria + ":\n");
            displayFilteredStudents();
        }

        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    } while (criteriaType != 0);
  }
```

```
//Sort Student
public void sortStudents() {
    int sortType;
    int sortDetail = 0;
    do {
        String criteria = null;
        String sortOrder = null;
        sortDetail = studentUI.getSortDetailsChoice();
        studentSortedList.clear();
        for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
            studentSortedList.add(studentList.getEntry(i));
        }
        switch (sortDetail) {
            case 0:
                break;
            case 1:
                //Sort Students by StudentID
                criteria = "Student ID";
                sortType = studentUI.inputSortType();
                switch (sortType) {
                    case 1:
                        sortOrder = "Ascending";
                        studentSortedList.sortByAscending(student -> student.getStudentID());
                        break;
                    case 2:
                        sortOrder = "Descending";
                        studentSortedList.sortByDescending(student -> student.getStudentID());
                        break;
                    default:
                        MessageUI.displayInvalidChoiceMessage();
                }
                break;
            case 2:
                //Sort Students by Name
                criteria = "Name";
                sortType = studentUI.inputSortType();
                switch (sortType) {
                    case 1:
                        sortOrder = "Ascending";
                        studentSortedList.sortByAscending(student -> student.getName());
                        break;
                    case 2:
                        sortOrder = "Descending";
                        studentSortedList.sortByDescending(student -> student.getName());
                        break;
                    default:
```

```
            MessageUI.displayInvalidChoiceMessage();
        }
        break;
    case 3:
        //Sort Students by Age
        criteria = "Age";
        sortType = studentUI.inputSortType();
        switch (sortType) {
            case 1:
                sortOrder = "Ascending";
                studentSortedList.sortByAscending(student -> student.getAge());
                break;
            case 2:
                sortOrder = "Descending";
                studentSortedList.sortByDescending(student -> student.getAge());
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
        break;
    case 4:
        //Sort Students by Gender
        criteria = "Gender";
        sortType = studentUI.inputSortType();
        switch (sortType) {
            case 1:
                sortOrder = "Ascending";
                studentSortedList.sortByAscending(student -> student.getGender());
                break;
            case 2:
                sortOrder = "Descending";
                studentSortedList.sortByDescending(student -> student.getGender());
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
        break;
    case 5:
        //Sort Students by Phone Number
        criteria = "Phone Number";
        sortType = studentUI.inputSortType();
        switch (sortType) {
            case 1:
                sortOrder = "Ascending";
                studentSortedList.sortByAscending(student -> student.getPhoneNo());
                break;
```

```
            case 2:
                sortOrder = "Descending";
                studentSortedList.sortByDescending(student -> student.getPhoneNo());
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
        break;
    case 6:
        //Sort Students by Email
        criteria = "Email";
        sortType = studentUI.inputSortType();
        switch (sortType) {
            case 1:
                sortOrder = "Ascending";
                studentSortedList.sortByAscending(student -> student.getEmail());
                break;
            case 2:
                sortOrder = "Descending";
                studentSortedList.sortByDescending(student -> student.getEmail());
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
        break;
    case 7:
        //Sort Students by Age
        criteria = "Programme";
        sortType = studentUI.inputSortType();
        switch (sortType) {
            case 1:
                sortOrder = "Ascending";
                studentSortedList.sortByAscending(student -> student.getProgrammeCode());
                break;
            case 2:
                sortOrder = "Descending";
                studentSortedList.sortByDescending(student -> student.getProgrammeCode());
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
        break;

    default:
        MessageUI.displayInvalidChoiceMessage();
}
```

```
            if (criteria != null && sortOrder != null) {
                System.out.println("\nList of Sorted Students based on " + criteria + " by " + sortOrder
+ " order:\n");
                displaySortedStudents();
            } else {
                System.out.println();
            }
            System.out.print("\nPress enter to continue...");
            scanner.nextLine();
        } while (sortDetail != 0);
    }

    //Calculate Course Fee of students
    public void calculateCourseFee() {
        do {
            ListInterface<String> studentCourseCodeList = new ArrayList<>();
            ListInterface<String> studentCourseNameList = new ArrayList<>();
            ListInterface<Double> courseAmountList = new ArrayList<>();
            ListInterface<String> courseStatusList = new ArrayList<>();
            Double total = 0.00;
            studentCourseCodeList.clear();
            studentCourseNameList.clear();
            courseStatusList.clear();
            courseAmountList.clear();
            displayStudents();
            String studentID = studentUI.inputStudentID();
            int index = -1;
            for (int i = 1; i < studentList.getNumberOfEntries() + 1; i++) {
                Student student = studentList.getEntry(i); // Retrieve Student object at index i
                if (student.getStudentID().equals(studentID)) {
                    index = i;
                    break; // Once found, exit the loop
                }
            }
            if (index != -1) {
                Student student = studentList.getEntry(index);
                String studentName = student.getName();
                String programmeName = student.getProgrammeCode();
                for (int i = 1; i <= enrollmentList.getNumberOfEntries(); i++) {
                    Enrollment enrollment = enrollmentList.getEntry(i); // Retrieve Student object at
index i
                    if (enrollment.getStudentID().equals(studentID)) {
                        studentCourseCodeList.add(enrollment.getCourseCode());
                        courseStatusList.add(enrollment.getEnrollmentStatus());
                    }
                }
```

```
            for (int i = 1; i <= studentCourseCodeList.getNumberOfEntries(); i++) {
                String courseCode = studentCourseCodeList.getEntry(i);
                for (int j = 1; j <= courseList.getNumberOfEntries(); j++) {
                    Course course = courseList.getEntry(j);
                    if (course.getCourseCode().equals(courseCode)) {
                        studentCourseNameList.add(course.getCourseName());

                        double fee = PRICEPERCREDITHOUR * course.getCreditHour();
                        courseAmountList.add(fee);
                        total += fee;
                    }
                }
            }

            //Display bill
            System.out.println();
            System.out.print("Student Name: " + studentName);
            System.out.print("\nStudent ID: " + studentID);
            System.out.print("\nProgramme: " + programmeName);
            System.out.println();
            studentUI.StudentBill();
            for (int i = 1; i <= studentCourseCodeList.getNumberOfEntries(); i++) {
                System.out.printf("%-8s %-38s %-14s %8.2f%n",
studentCourseCodeList.getEntry(i),
                        studentCourseNameList.getEntry(i), courseStatusList.getEntry(i),
                        courseAmountList.getEntry(i));
            }
            System.out.println("---------------------------------------------------------------------------");
            System.out.printf("%62s %8.2f", "Total Amount: ", total);

System.out.println("\n---------------------------------------------------------------------------\n");
        } else {
            System.out.println("\nStudent with Student ID: " + studentID + " not found in the
list.");
        }
    } while ("Yes".equals(studentUI.inputCalculateCourseFeeAgain()));
  }

  //Summary Report
  public void displayReport() {
    int reportType = 0;
    do {
      reportType = studentUI.inputReportType();
      switch (reportType) {
        case 0:
          break;
```

```
            case 1:
                StudentReport();
                break;
            case 2:
                CourseReport();
                break;
            default:
                MessageUI.displayInvalidChoiceMessage();
        }
    } while (reportType != 0);
}

//Search for a student with the given student ID in the student list
public boolean findStudent(String studentID) {
    for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
        if (studentList.getEntry(i) != null) {
            if (studentList.getEntry(i).getStudentID().equals(studentID)) {
                return true;
            }
        }
    }
    return false;
}

//Search for a programme with the given programme code in the student list
public boolean findProgramme(String programmmeCode) {
    for (int i = 1; i <= programmeList.getNumberOfEntries(); i++) {
        if (programmeList.getEntry(i) != null) {
            if (programmeList.getEntry(i).getProgrammeCode().equals(programmmeCode)) {
                return true;
            }
        }
    }
    return false;
}

//Determines whether a course with a specified code exists in the course list.
public boolean findCourse(String courseCode) {
    for (int i = 1; i <= courseList.getNumberOfEntries(); i++) {
        if (courseList.getEntry(i) != null) {
            if (courseList.getEntry(i).getCourseCode().equals(courseCode)) {
                return true;
            }
        }
    }
    return false;
```

```
}

//Retrieves all students from the student list and returns them as a formatted string.
public String getAllStudents() {
    String outputStr = "";
    for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
        outputStr += studentList.getEntry(i) + "\n";
    }
    return outputStr;
}

//Displays the list of students
public void displayStudents() {
    System.out.println("\nList of Students:\n");
    studentUI.listAllStudents(getAllStudents());
}

//Retrieves all courses from the course list and returns them as a formatted string.
public String getAllCourses() {
    String outputStr = "";
    for (int i = 1; i <= courseList.getNumberOfEntries(); i++) {
        outputStr += courseList.getEntry(i) + "\n";
    }
    return outputStr;
}

//Retrieves all progrramme from the course list and returns them as a formatted string.
public String getAllProgrammes() {
    String outputStr = "";
    for (int i = 1; i <= programmeList.getNumberOfEntries(); i++) {
        outputStr += programmeList.getEntry(i) + "\n";
    }
    return outputStr;
}

//Displays the list of programme
public void displayProgramme() {
    courseUI.listAllProgramme(getAllProgrammes());
}

//Displays the list of courses
public void displayCourses() {
    courseUI.listAllCourse(getAllCourses());
}

//Retrieves filtered students from the student filtered list and returns them as a formatted
```

```
string.
    public String getFilteredStudents() {
        String outputStr = "";
        for (int i = 1; i <= studentFilteredList.getNumberOfEntries(); i++) {
            outputStr += studentFilteredList.getEntry(i) + "\n";
        }
        return outputStr;
    }

    //Displays the list of filtered students
    public void displayFilteredStudents() {
        studentUI.listAllStudents(getFilteredStudents());
    }

    //Retrieves sorted students from the student sorted list and returns them as a formatted string.
    public String getSortedStudents() {
        String outputStr = "";
        for (int i = 1; i <= studentSortedList.getNumberOfEntries(); i++) {
            outputStr += studentSortedList.getEntry(i) + "\n";
        }
        return outputStr;
    }

    //Displays the list of sorted students
    public void displaySortedStudents() {
        studentUI.listAllStudents(getSortedStudents());
    }

    //Retrieve the last student's student id last 4 number
    public int getStudentIDNum(){
        int count = 999;
        for (int i = 1; i <= studentList.getNumberOfEntries(); i++){
            count++;
        }
        return count;
    }

    //Student Summary Report
    public void StudentReport() {
        ListInterface<Student> StudentDetails = new ArrayList<>();
        ListInterface<String> TEMPCourseCode = new ArrayList<>();
        ListInterface<Integer> CourseCount = new ArrayList<>();
        ListInterface<Integer> CreditHourCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int totalCreditHour = 0;
```

```
        int max = 0;
        int min = 999;

        for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
            StudentDetails.add(studentList.getEntry(i));
        }

        StudentDetails.sortByAscending(student -> student.getStudentID());

        for (int i = 1; i <= StudentDetails.getNumberOfEntries(); i++) {
            for (int j = 1; j <= enrollmentList.getNumberOfEntries(); j++) {
                if
(StudentDetails.getEntry(i).getStudentID().equals(enrollmentList.getEntry(j).getStudentID())) {
                    TEMPCourseCode.add(enrollmentList.getEntry(j).getCourseCode());
                    for (int k = 1; k <= courseList.getNumberOfEntries(); k++) {
                        if
(enrollmentList.getEntry(j).getCourseCode().equals(courseList.getEntry(k).getCourseCode())) {
                            totalCreditHour += courseList.getEntry(k).getCreditHour();

                        }
                    }
                }
            }
            CreditHourCount.add(totalCreditHour);
            CourseCount.add(TEMPCourseCode.getNumberOfEntries());
            TEMPCourseCode.clear();
            totalCreditHour = 0;
        }

        studentUI.StudentReport();
        for (int i = 1; i <= StudentDetails.getNumberOfEntries(); i++) {
            System.out.printf("%2d. %-20s %-78s %7d %22d\n", i,
StudentDetails.getEntry(i).getStudentID(),
                    StudentDetails.getEntry(i).getName(), CourseCount.getEntry(i),
CreditHourCount.getEntry(i));
        }
        System.out.print("-".repeat(150));
        System.out.print("\nTotal " + StudentDetails.getNumberOfEntries() + " Students\n");

        for (int i = 1; i <= CourseCount.getNumberOfEntries(); i++) {
            if (CourseCount.getEntry(i) != 0) {
                if (CourseCount.getEntry(i) > max) {
                    max = CourseCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (CourseCount.getEntry(i) == max) {
```

```
                maxIndex.add(i);
            }
            if (CourseCount.getEntry(i) < min) {
                min = CourseCount.getEntry(i);
                minIndex.clear();
                minIndex.add(i);
            } else if (CourseCount.getEntry(i) == min) {
                minIndex.add(i);
            }
        }
    }

    System.out.print("-".repeat(150) + "\n" + "HIGHEST COURSES TAKEN: ");

    for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
        System.out.print("\n-> [" + max + " Courses] " + "<" +
StudentDetails.getEntry(maxIndex.getEntry(i)).getStudentID() + "> " +
StudentDetails.getEntry(maxIndex.getEntry(i)).getName());
    }

    System.out.print("\n\nLOWEST COURSES TAKEN : ");

    for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
        System.out.print("\n-> [" + min + " Courses] " + "<" +
StudentDetails.getEntry(minIndex.getEntry(i)).getStudentID() + "> " +
StudentDetails.getEntry(minIndex.getEntry(i)).getName());
    }

    System.out.print("\n[NOTE: 0 COURSES IS NOT COUNTED]" + "\n" + "-".repeat(150));

    max = 0;
    min = 999;

    maxIndex.clear();

    minIndex.clear();

    for (int i = 1; i <= CreditHourCount.getNumberOfEntries(); i++) {
        if (CreditHourCount.getEntry(i) != 0) {
            if (CreditHourCount.getEntry(i) > max) {
                max = CreditHourCount.getEntry(i);
                maxIndex.clear();
                maxIndex.add(i);
            } else if (CreditHourCount.getEntry(i) == max) {
                maxIndex.add(i);
            }
```

```
            if (CreditHourCount.getEntry(i) < min) {
                min = CreditHourCount.getEntry(i);
                minIndex.clear();
                minIndex.add(i);
            } else if (CreditHourCount.getEntry(i) == min) {
                minIndex.add(i);
            }
        }
    }

    System.out.print("\nHIGHEST CREDIT HOURS OBTAINED: ");

    for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
        System.out.print("\n-> [" + max + " Credit Hours] " + "<" +
StudentDetails.getEntry(maxIndex.getEntry(i)).getStudentID() + "> " +
StudentDetails.getEntry(maxIndex.getEntry(i)).getName());
    }

    System.out.print("\n\nLOWEST CREDIT HOURS OBTAINED : ");

    for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
        System.out.print("\n-> [" + min + " Credit Hours] " + "<" +
StudentDetails.getEntry(minIndex.getEntry(i)).getStudentID() + "> " +
StudentDetails.getEntry(minIndex.getEntry(i)).getName());
    }

    System.out.print(
            "\n[NOTE: 0 CREDIT HOURS IS NOT COUNTED]\n"
            + "-".repeat(150) + "\n"
            + " ".repeat(60) + "END OF STUDENT SUMMARY REPORT\n"
            + "=".repeat(150));

    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

//Student Course Report
public void CourseReport() {
    ListInterface<Course> CourseDetails = new ArrayList<>();
    ListInterface<String> TEMPStudentID = new ArrayList<>();
    ListInterface<String> TEMPStatus = new ArrayList<>();
    ListInterface<Integer> StudentCount = new ArrayList<>();
    ListInterface<Integer> StatusMainCount = new ArrayList<>();
    ListInterface<Integer> StatusRepeatCount = new ArrayList<>();
    ListInterface<Integer> StatusResitCount = new ArrayList<>();
    ListInterface<Integer> StatusElectiveCount = new ArrayList<>();
```

```
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int MainCount = 0;
        int RepeatCount = 0;
        int ResitCount = 0;
        int ElectiveCount = 0;
        int max = 0;
        int min = 999;

        for (int i = 1; i <= courseList.getNumberOfEntries(); i++) {
            CourseDetails.add(courseList.getEntry(i));
        }

        CourseDetails.sortByAscending(course -> course.getCourseCode());

        for (int i = 1; i <= CourseDetails.getNumberOfEntries(); i++) {
            for (int j = 1; j <= enrollmentList.getNumberOfEntries(); j++) {
                if
(CourseDetails.getEntry(i).getCourseCode().equals(enrollmentList.getEntry(j).getCourseCode())
) {
                    TEMPStudentID.add(enrollmentList.getEntry(j).getStudentID());
                    TEMPStatus.add(enrollmentList.getEntry(j).getEnrollmentStatus());
                }
            }
            for (int k = 1; k <= TEMPStatus.getNumberOfEntries(); k++) {
                if (TEMPStatus.getEntry(k).equals("Main")) {
                    MainCount += 1;
                } else if (TEMPStatus.getEntry(k).equals("Repeat")) {
                    RepeatCount += 1;
                } else if (TEMPStatus.getEntry(k).equals("Resit")) {
                    ResitCount += 1;
                } else if (TEMPStatus.getEntry(k).equals("Elective")){
                    ElectiveCount += 1;
                }
            }

            StudentCount.add(TEMPStudentID.getNumberOfEntries());
            StatusMainCount.add(MainCount);
            StatusRepeatCount.add(RepeatCount);
            StatusResitCount.add(ResitCount);
            StatusElectiveCount.add(ElectiveCount);
            TEMPStudentID.clear();
            TEMPStatus.clear();
            MainCount = 0;
            RepeatCount = 0;
            ResitCount = 0;
```

```
            ElectiveCount = 0;
        }

        studentUI.CourseReport();

        for (int i = 1; i <= CourseDetails.getNumberOfEntries(); i++) {
            System.out.printf("%2d. %-15s %-53s %-19.1f %-17d Main:%-2d Resit:%-2d
Repeat:%-2d Elective:%-2d\n", i, CourseDetails.getEntry(i).getCourseCode(),
                    CourseDetails.getEntry(i).getCourseName(),
CourseDetails.getEntry(i).getCreditHour(),
                    StudentCount.getEntry(i), StatusMainCount.getEntry(i),
StatusResitCount.getEntry(i),
                    StatusRepeatCount.getEntry(i), StatusElectiveCount.getEntry(i));
        }
        System.out.print("-".repeat(150));
        System.out.print("\nTotal " + CourseDetails.getNumberOfEntries() + " Courses\n");

        for (int i = 1; i <= StudentCount.getNumberOfEntries(); i++) {
            if (StudentCount.getEntry(i) != 0) {
                if (StudentCount.getEntry(i) > max) {
                    max = StudentCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (StudentCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (StudentCount.getEntry(i) < min) {
                    min = StudentCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (StudentCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("-".repeat(150) + "\n" + "TOP ENROLLED COURSE: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Students] " + "<" +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nLEAST ENROLLED COURSE : ");
```

```
        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Students] " + "<" +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n[NOTE: 0 STUDENTS IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;

        maxIndex.clear();

        minIndex.clear();

        for (int i = 1; i <= StatusMainCount.getNumberOfEntries(); i++) {
            if (StatusMainCount.getEntry(i) != 0) {
                if (StatusMainCount.getEntry(i) > max) {
                    max = StatusMainCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (StatusMainCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (StatusMainCount.getEntry(i) < min) {
                    min = StatusMainCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (StatusMainCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("\nCOURSE WITH HIGHEST MAIN ENROLLMENT: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Students] " + "<" +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nCOURSE WITH LOWEST MAIN ENROLLMENT: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
```

```
            System.out.print("\n-> [" + min + " Students] " + "<" +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n[NOTE: 0 STUDENTS IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;

        maxIndex.clear();

        minIndex.clear();

        for (int i = 1; i <= StatusResitCount.getNumberOfEntries(); i++) {
            if (StatusResitCount.getEntry(i) != 0) {
                if (StatusResitCount.getEntry(i) > max) {
                    max = StatusResitCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (StatusResitCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (StatusResitCount.getEntry(i) < min) {
                    min = StatusResitCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (StatusResitCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("\nCOURSE WITH HIGHEST RESIT ENROLLMENT: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Students] " + "<" +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nCOURSE WITH LOWEST RESIT ENROLLMENT: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Students] " + "<" +
```

```
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n[NOTE: 0 STUDENTS IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;

        maxIndex.clear();

        minIndex.clear();

        for (int i = 1; i <= StatusRepeatCount.getNumberOfEntries(); i++) {
            if (StatusRepeatCount.getEntry(i) != 0) {
                if (StatusRepeatCount.getEntry(i) > max) {
                    max = StatusRepeatCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (StatusRepeatCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (StatusRepeatCount.getEntry(i) < min) {
                    min = StatusRepeatCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (StatusRepeatCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("\nCOURSE WITH HIGHEST REPEAT ENROLLMENT: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Students] " + "<" +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nCOURSE WITH LOWEST REPEAT ENROLLMENT: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Students] " + "<" +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
```

```
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseName());
        }
        System.out.print("\n[NOTE: 0 STUDENTS IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;

        maxIndex.clear();

        minIndex.clear();

        for (int i = 1; i <= StatusElectiveCount.getNumberOfEntries(); i++) {
            if (StatusElectiveCount.getEntry(i) != 0) {
                if (StatusElectiveCount.getEntry(i) > max) {
                    max = StatusElectiveCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (StatusElectiveCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (StatusElectiveCount.getEntry(i) < min) {
                    min = StatusElectiveCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (StatusElectiveCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("\nCOURSE WITH HIGHEST ELECTIVE ENROLLMENT: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Students] " + "<" +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nCOURSE WITH LOWEST ELECTIVE ENROLLMENT: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Students] " + "<" +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetails.getEntry(minIndex.getEntry(i)).getCourseName());
        }
```

```
        System.out.print(
                "\n[NOTE: 0 STUDENTS IS NOT COUNTED]\n"
                + "-".repeat(150) + "\n"
                + " ".repeat(60) + "END OF STUDENT SUMMARY REPORT\n"
                + "=".repeat(150));
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public static void main(String[] args) {
        StudentManagement studentManagement = new StudentManagement();
        studentManagement.runStudentManagement();
    }
}
```

1. **Screenshots**
   <remove instructions>
   sample user interface (console / GUI) input screens and outputs. Label your screenshots clearly.

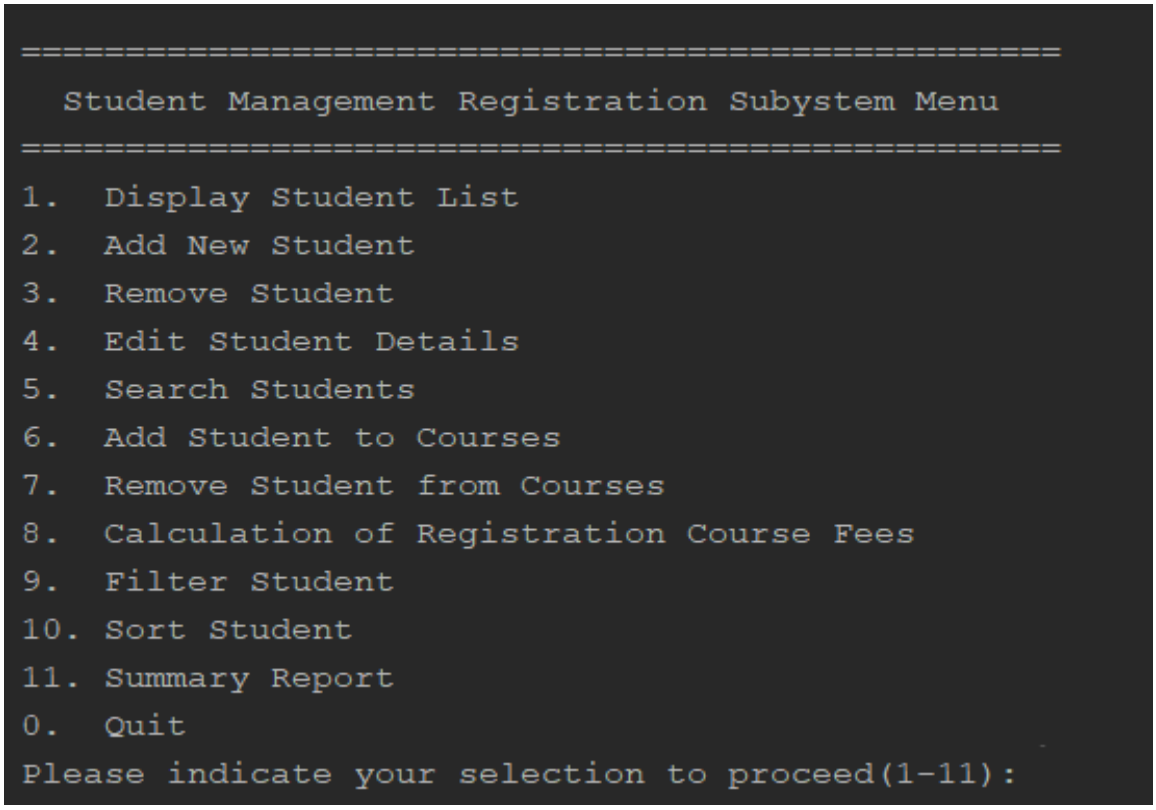


Figure 1.1: Menu of the Student Management Registration Subsystem

Figure 1.1 shows the menu of the Student Management Registration Subsystem and prompts the user for input options to proceed.

```
=================================================
  Student Management Registration Subystem Menu
=================================================
1.  Display Student List
2.  Add New Student
3.  Remove Student
4.  Edit Student Details
5.  Search Students
6.  Add Student to Courses
7.  Remove Student from Courses
8.  Calculation of Registration Course Fees
9.  Filter Student
10. Sort Student
11. Summary Report
0.  Quit
Please indicate your selection to proceed(1-11): ggg
Error: Invalid input format. Please enter valid data.
Please indicate your selection to proceed(1-11):
```

Figure 1.2: Error message displayed after invalid data entry. The user must re-enter.

```
Please indicate your selection to proceed(1-11): 1


List of Students:

-------------------------------------------------------------------------------------------------------------------
Student ID   Name                  Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan         21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1001        Lim Hoi Yau           21    Male     limhy-wp22@student.tarc.edu.my  018-12326756    RSW
S1002        Goh Boon Xiang        22    Male     gohbx-wp21@student.tarc.edu.my  016-34098124    RDS
S1003        Tan Hoong Guan        23    Male     tanhg-wp21@student.tarc.edu.my  018-2277975     RFN
S1004        Loh Jia Shou          21    Male     lohjs-wp21@student.tarc.edu.my  017-8319687     RME
S1005        Tan Ling Ling         21    Female   tanll-wp21@student.tarc.edu.my  012-12341234    RDS
```

Figure 1.3: The list of students is displayed

If user input is ‘1’, a student list will be displayed to the user.

```
Enter New Student Name: Tan
Enter New Student Age: 21
Enter New Student Gender: Male
Enter New Student Phone Number: 012-3455678
Enter New Student Email: tan-wp21@student.tarc.edu.my
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code        Programme Name
-----------------------------------------------------------------------------------------------------
RDS                   Bachelor Of Computer Science (HONOURS) In Data Science
RSW                   Bachelor Of Software Engineering (HONOURS)
RSD                   Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                   Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                   Bachelor Of Information Technology (Honours) In Information Security
RST                   Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                   Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                   Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                   Bachelor Of Science (Hons) In Analytical Chemistry
RFN                   Bachelor Of Science (Hons) In Food Science
RNR                   Bachelor Of Science (Hons) In Nutrition
RSE                   Bachelor Of Science (Hons) In Sports And Exercise Science
REE                   Bachelor Of Electrical And Electronics Engineering With Honours
RME                   Bachelor Of Mechanical Engineering With Honours
RMT                   Bachelor Of Engineering (Honours) Material

**PLEASE REFER PROGRAMME LIST SHOWN ABOVE
Enter New Student Programme Code: RDS

New student has been added successfully!

Do you want enter new student again?(Yes/No):
```

Figure 1.4: Add New Student

In Add New Student, the user is required to input the new student's name, age, gender, phone number, email and programme code to add a new student to the student list in the system. A programme list will be shown before the user inputs programme code for the user to refer to. Lastly, it prompts the user to enter whether they want to add new student again.

```
-----------------------------------------------------------------------------------------------------
Programme Code        Programme Name
-----------------------------------------------------------------------------------------------------
RDS                   Bachelor Of Computer Science (HONOURS) In Data Science
RSW                   Bachelor Of Software Engineering (HONOURS)
RSD                   Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                   Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                   Bachelor Of Information Technology (Honours) In Information Security
RST                   Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                   Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                   Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                   Bachelor Of Science (Hons) In Analytical Chemistry
RFN                   Bachelor Of Science (Hons) In Food Science
RNR                   Bachelor Of Science (Hons) In Nutrition
RSE                   Bachelor Of Science (Hons) In Sports And Exercise Science
REE                   Bachelor Of Electrical And Electronics Engineering With Honours
RME                   Bachelor Of Mechanical Engineering With Honours
RMT                   Bachelor Of Engineering (Honours) Material

**PLEASE REFER PROGRAMME LIST SHOWN ABOVE
Enter New Student Programme Code: AAA
Invalid Programme Code! Please input a valid Programme Code.

**PLEASE REFER PROGRAMME LIST SHOWN ABOVE
Enter New Student Programme Code:
```

Figure 1.5: Error message displayed after invalid programme code entry. The user must re-enter.

```
S1030        Tan Xing                  25    Female   tanx-wp21@student.tarc.edu.my       019-45327895    RAN
S1031        Lim Xiao                  20    Male     limx-wp22@student.tarc.edu.my       011-15423601    RSE
S1032        Goh Xi                    22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1033        Tan Xia Xia               24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing             27    Female   lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter the studentID of the student you wish to remove: S1034

Student with Student ID: S1034 removed successfully.

Press enter to continue...
```

Figure 1.6: Remove Student Module

In the Remove Student module, a student list is displayed and prompts the user to input student ID. Figure 1.6 shows that a student with Student ID: S1034 is removed from the student list

```
S1025        Tan Ting                  25    Female   tant-wp21@student.tarc.edu.my       019-4562585     RAN
S1026        Lim Tao                   20    Male     limt-wp22@student.tarc.edu.my       011-75395120    RSE
S1027        Goh Ti                    22    Female   goht-wp21@student.tarc.edu.my       016-85285245    RSE
S1028        Tan Tuan Tuan             24    Male     tantt-wp21@student.tarc.edu.my      013-75486914    RSE
S1029        Loh Ting Ting             27    Female   lohtt-wp21@student.tarc.edu.my      015-02175855    RME
S1030        Tan Xing                  25    Female   tanx-wp21@student.tarc.edu.my       019-45327895    RAN
S1031        Lim Xiao                  20    Male     limx-wp22@student.tarc.edu.my       011-15423601    RSE
S1032        Goh Xi                    22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1033        Tan Xia Xia               24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE

Please enter the studentID of the student you wish to remove: S1066

Student with Student ID: S1066 not found in the list.
```

Figure 1.7: Student with Student ID: S1066 is not found in the student list

```
S1027        Goh Ti                    22    Female   goht-wp21@student.tarc.edu.my       016-85285245    RSE
S1028        Tan Tuan Tuan             24    Male     tantt-wp21@student.tarc.edu.my      013-75486914    RSE
S1029        Loh Ting Ting             27    Female   lohtt-wp21@student.tarc.edu.my      015-02175855    RME
S1030        Tan Xing                  25    Female   tanx-wp21@student.tarc.edu.my       019-45327895    RAN
S1031        Lim Xiao                  20    Male     limx-wp22@student.tarc.edu.my       011-15423601    RSE
S1032        Goh Xi                    22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1033        Tan Xia Xia               24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE

Please enter the Student ID of the student whose details you wish to edit: S1032

Edit Student Details Menu
1. Name
2. Age
3. Gender
4. Phone Number
5. Email
6. Programme
0: Back
Please select the details you would like to modify(1-6):
```

Figure 1.8: Edit Student Details Module

Based on Figure 1.8, a student list is displayed and prompts the user to input student ID. After that, A details menu will be displayed and prompts the user to input the choice.

```
Please select the details you would like to modify(1-6): 1

Old Name: Goh Xi
Please enter NEW name: Goh Mei Li

Student's name modified successfully!
```

Figure 1.9: Student's name with student ID: S1032 has changed successfully.

```
S1031        Lim Xiao                 20    Male     limx-wp22@student.tarc.edu.my       011-15423601    RSE
S1032        Goh Xi                   22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1033        Tan Xia Xia              24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing            27    Female   lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter Student ID: S1000

Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                              Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS1053        Database Management                      3.00           202401
BACS2026        Data Structure And Algorithms            4.00           202401
BAIT1023        Web Design And Development               2.00           202401
BAIT1043        System Analysis And Design               3.00           202401
BABS1233        Microbiology                             2.00           202401
BACH2233        Food Chemistry And Analysis              4.00           202401
BACH1613        Physical Chemistry                       3.00           202401
BTME4263        Heat Transfer                            3.00           202401
BGEE2614        Electrical Power Systems                 2.00           202401
BTEE1513        Electrical Technology                    4.00           202401
BTEE4033        Integrated Circuit Technology            3.00           202401
BJEL1023        Academic English                         3.00           202401
BJEL1013        English For Tertiary Studies             3.00           202401
BAMS1413        Calculus And Algebra                     3.00           202401
BAMS2014        Linear Algebra                           3.00           202401

Please enter Course Code: BACS2026

Student with Student ID: S1000 is enrolled in BACS2026 as Main

Do you want search student for registered course again?(Yes/No):
```

Figure 1.10: Search Students Module

Based on Figure 1.10, a student list is displayed and prompts the user to input student ID. After that, A course list will be displayed and prompt the user to input the course code. The results will then show whether the student is already enrolled in the course and prompt the user to enter whether they want to search again. As shown in Figure 1.10, the student with input student ID: S1000 is enrolled in the input course code which is BACS2026 as Main.

```
S1033        Tan Xia Xia              24    Male      tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing            27    Female    lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter Student ID: S1000

Course Details:

-----------------------------------------------------------------------------------------------------
Course Code     Course Name                                 Credit Hour    Semester
-----------------------------------------------------------------------------------------------------
BACS1053        Database Management                         3.00           202401
BACS2026        Data Structure And Algorithms               4.00           202401
BAIT1023        Web Design And Development                  2.00           202401
BAIT1043        System Analysis And Design                  3.00           202401
BABS1233        Microbiology                                2.00           202401
BACH2233        Food Chemistry And Analysis                 4.00           202401
BACH1613        Physical Chemistry                          3.00           202401
BTME4263        Heat Transfer                               3.00           202401
BGEE2614        Electrical Power Systems                    2.00           202401
BTEE1513        Electrical Technology                       4.00           202401
BTEE4033        Integrated Circuit Technology               3.00           202401
BJEL1023        Academic English                            3.00           202401
BJEL1013        English For Tertiary Studies                3.00           202401
BAMS1413        Calculus And Algebra                        3.00           202401
BAMS2014        Linear Algebra                              3.00           202401

Please enter Course Code: BABS1233

Student with Student ID: S1000 is not enrolled in BABS1233
```

Figure 1.11: The message about the student with student ID: S1000 not enrolling in BABS1233 is displayed.

```
S1033        Tan Xia Xia              24    Male      tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing            27    Female    lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter Student ID: S1000

Course Details:

-----------------------------------------------------------------------------------------------------
Course Code     Course Name                                 Credit Hour    Semester
-----------------------------------------------------------------------------------------------------
BACS1053        Database Management                         3.00           202401
BACS2026        Data Structure And Algorithms               4.00           202401
BAIT1023        Web Design And Development                  2.00           202401
BAIT1043        System Analysis And Design                  3.00           202401
BABS1233        Microbiology                                2.00           202401
BACH2233        Food Chemistry And Analysis                 4.00           202401
BACH1613        Physical Chemistry                          3.00           202401
BTME4263        Heat Transfer                               3.00           202401
BGEE2614        Electrical Power Systems                    2.00           202401
BTEE1513        Electrical Technology                       4.00           202401
BTEE4033        Integrated Circuit Technology               3.00           202401
BJEL1023        Academic English                            3.00           202401
BJEL1013        English For Tertiary Studies                3.00           202401
BAMS1413        Calculus And Algebra                        3.00           202401
BAMS2014        Linear Algebra                              3.00           202401

Please enter Course Code: BACS2026
Please enter Enrollment Status (Main/Resit/Repeat/Elective): Main

Student has been added to a new course successfully!

Do you want add student to a new course again?(Yes/No):
```

Figure 1.12: Add Student to Courses Module

Based on Figure 1.12, a student list is displayed and prompts the user to input a student ID. After that, a course list will be displayed and prompt the user to input the course code. It then prompts the user to input the enrollment status. The student will be added to the courses after valid input is inserted and prompt the user to enter whether they want to add student to course again.

```
S1032        Goh Xi                  22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1033        Tan Xia Xia             24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing           27    Female   lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter Student ID: S1034

Course Details:

-----------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-----------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401

Please enter Course Code: BACH2233
This student is ineligible for course enrollment due to a program code mismatch. Please provide a valid Course Code.
Please enter Course Code:
```

Figure 1.13: Error message displayed after invalid course code entry. The user must re-enter.

Based on Figure 1.13, if the course code entered by the user does not match the student's programme, then the error message will be displayed and prompt the user to enter the course code again.

```
S1033        Tan Xia Xia             24    Male     tanxx-wp21@student.tarc.edu.my      013-85147320    RSE
S1034        Loh Xing Xing           27    Female   lohtt-wp21@student.tarc.edu.my      015-84527545    RME

Please enter Student ID: S1000

Course Details:

-----------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-----------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401

Please enter the Course Code you wish to remove from your student course: BACS2026

Please enter Enrollment Status(Main/Resit/Repeat/Elective): Main

Student with Student ID: S1000 removed successfully from BACS2026 as Main
```

Figure 1.14: Remove Student from Courses Module

Based on Figure 1.14, a student list is displayed and prompts the user to input a student ID. After that, a course list will be displayed and prompt the user to input the course code. It then prompts the user to input the enrollment status. The student will removed from the courses after valid input is inserted, otherwise, the error message will displayed.

```
S1030        Tan Xing                  25    Female   tanx-wp21@student.tarc.edu
S1031        Lim Xiao                  20    Male     limx-wp22@student.tarc.edu
S1032        Goh Xi                    22    Female   gohx-wp21@student.tarc.edu
S1033        Tan Xia Xia               24    Male     tanxx-wp21@student.tarc.ed
S1034        Loh Xing Xing             27    Female   lohtt-wp21@student.tarc.ed

Please enter Student ID: S1000

Student Name: Tan Lock Kwan
Student ID: S1000
Programme: RDS
--------------------------------------------------------------------------------
        TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY
                            STUDENT BILL
--------------------------------------------------------------------------------
Courses                                          Status          Amount(RM)
--------------------------------------------------------------------------------
BACS1053 Database Management                     Resit             600.00
BAMS1413 Calculus And Algebra                    Main              600.00
--------------------------------------------------------------------------------
                                                 Total Amount:    1200.00
--------------------------------------------------------------------------------

Do you want calculate course fee for another student?(Yes/No):
```

Figure 1.15: Calculation of Registration Course FeesModule

Based on Figure 1.15, a student list is displayed and prompts the user to input a student ID. After that, the student's name, student ID, Programme and student bill will be displayed and prompt the user to enter whether they want to calculate the course fee again for another student.

```
==================================================
  Student Management Registration Subystem Menu
==================================================
1.  Display Student List
2.  Add New Student
3.  Remove Student
4.  Edit Student Details
5.  Search Students
6.  Add Student to Courses
7.  Remove Student from Courses
8.  Calculation of Registration Course Fees
9.  Filter Student
10. Sort Student
11. Summary Report
0.  Quit
Please indicate your selection to proceed(1-11): 9


Criteria Menu
1. Age
2. Gender
3. ProgrammeCode
0. Back
Please select the criteria by which you want to filter students(1-3):
```

Figure 1.16: Student List Filtering Criteria

```
Criteria Menu
1. Age
2. Gender
3. ProgrammeCode
0. Back
Please select the criteria by which you want to filter students(1-3): 1
Please enter the minimum age: 20
Please enter the maximum age: 21

List of Filtered Students based on Age:

-------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                              Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my      012-11112222    RDS
S1001        Lim Hoi Yau             21    Male     limhy-wp22@student.tarc.edu.my      018-12326756    RSW
S1004        Loh Jia Shou            21    Male     lohjs-wp21@student.tarc.edu.my      017-8319687     RME
S1005        Tan Ling Ling           21    Female   tanll-wp21@student.tarc.edu.my      012-12341234    RDS
S1006        Lim Jia Jia             21    Male     limjj-wp22@student.tarc.edu.my      018-123456789   RSW
S1009        Loh Jia Jia             21    Male     lohjj-wp21@student.tarc.edu.my      017-10201312    REE
S1010        Tan Qing                21    Female   tanq-wp21@student.tarc.edu.my       012-4561230     RDS
S1011        Lim Qiang               21    Male     limq-wp22@student.tarc.edu.my       018-78124510    RDS
S1014        Loh Qi                  21    Male     lohq-wp21@student.tarc.edu.my       017-79791313    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my       018-4562585     RDS
S1016        Lim Yao                 21    Male     limy-wp22@student.tarc.edu.my       017-75395120    RDS
S1019        Loh Ying                21    Female   lohy-wp21@student.tarc.edu.my       014-02175855    RDS
S1020        Tan Ping                21    Female   tanp-wp21@student.tarc.edu.my       017-4561230     RSD
S1021        Lim Peng                21    Male     limp-wp22@student.tarc.edu.my       012-78124510    RMM
S1026        Lim Tao                 20    Male     limt-wp22@student.tarc.edu.my       011-75395120    RSE
S1031        Lim Xiao                20    Male     limx-wp22@student.tarc.edu.my       011-15423601    RSE


Press enter to continue...
```

Figure 1.16: Filtered Student List based on Age

```
Criteria Menu
1. Age
2. Gender
3. ProgrammeCode
0. Back
Please select the criteria by which you want to filter students(1-3): 2
Please enter the gender: Female

List of Filtered Students based on Gender:

-------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                              Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my      012-11112222    RDS
S1005        Tan Ling Ling           21    Female   tanll-wp21@student.tarc.edu.my      012-12341234    RDS
S1010        Tan Qing                21    Female   tanq-wp21@student.tarc.edu.my       012-4561230     RDS
S1012        Goh Qian                22    Female   gohq-wp21@student.tarc.edu.my       016-98235610    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my       018-4562585     RDS
S1017        Goh Yi                  22    Female   gohy-wp21@student.tarc.edu.my       014-85285245    REE
S1019        Loh Ying                21    Female   lohy-wp21@student.tarc.edu.my       014-02175855    RDS
S1020        Tan Ping                21    Female   tanp-wp21@student.tarc.edu.my       017-4561230     RSD
S1022        Goh Pan                 26    Female   gohp-wp21@student.tarc.edu.my       013-98235610    RIS
S1025        Tan Ting                25    Female   tant-wp21@student.tarc.edu.my       019-4562585     RAN
S1027        Goh Ti                  22    Female   goht-wp21@student.tarc.edu.my       016-85285245    RSE
S1029        Loh Ting Ting           27    Female   lohtt-wp21@student.tarc.edu.my      015-02175855    RME
S1030        Tan Xing                25    Female   tanx-wp21@student.tarc.edu.my       019-45327895    RAN
S1032        Goh Xi                  22    Female   gohx-wp21@student.tarc.edu.my       016-75192846    RSE
S1034        Loh Xing Xing           27    Female   lohtt-wp21@student.tarc.edu.my      015-84527545    RME


Press enter to continue...
```

Figure 1.17: Filtered Student List based on Gender

```
Criteria Menu
1. Age
2. Gender
3. ProgrammeCode
0. Back
Please select the criteria by which you want to filter students(1-3): 3
Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter the programme code: RMT

List of Filtered Students based on Programme Code:

-------------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------------
S1007        Goh Jia Jie             22    Male     gohjj-wp21@student.tarc.edu.my  016-45678956    RMT
S1008        Tan Jia Jun             23    Male     tanjj-wp21@student.tarc.edu.my  018-45623123    RMT
```

Figure 1.18: Filtered Student List based on Programme

```
==================================================
  Student Management Registration Subystem Menu
==================================================
1.  Display Student List
2.  Add New Student
3.  Remove Student
4.  Edit Student Details
5.  Search Students
6.  Add Student to Courses
7.  Remove Student from Courses
8.  Calculation of Registration Course Fees
9.  Filter Student
10. Sort Student
11. Summary Report
0.  Quit
Please indicate your selection to proceed(1-11): 10


Details Menu
1. Student ID
2. Name
3. Age
4. Gender
5. Phone Number
6. Email
7. Programme
0. Back
Please choose the criteria by which you'd like to sort the details in the student list(1-7):1


1. Ascending Order
2. Desencing Order
Please enter your choice of sort order(1-2): 1
```

Figure 1.19: Sort Student Module

According to Figure 1.19, there are 7 student criteria for users to choose from to sort the student list. After selecting the criteria, prompt the user to enter the desired ascending or descending sort order.

```
List of Sorted Students based on Student ID by Ascending order:

-------------------------------------------------------------------------------------------------------------------------
Student ID   Name                 Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan        21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1001        Lim Hoi Yau          21    Male     limhy-wp22@student.tarc.edu.my  018-12326756    RSW
S1002        Goh Boon Xiang       22    Male     gohbx-wp21@student.tarc.edu.my  016-34098124    RDS
S1003        Tan Hoong Guan       23    Male     tanhg-wp21@student.tarc.edu.my  018-2277975     RFN
S1004        Loh Jia Shou         21    Male     lohjs-wp21@student.tarc.edu.my  017-8319687     RME
S1005        Tan Ling Ling        21    Female   tanll-wp21@student.tarc.edu.my  012-12341234    RDS
S1006        Lim Jia Jia          21    Male     limjj-wp22@student.tarc.edu.my  018-123456789   RSW
S1007        Goh Jia Jie          22    Male     gohjj-wp21@student.tarc.edu.my  016-45678956    RMT
S1008        Tan Jia Jun          23    Male     tanjj-wp21@student.tarc.edu.my  018-45623123    RMT
S1009        Loh Jia Jia          21    Male     lohjj-wp21@student.tarc.edu.my  017-10201312    REE
S1010        Tan Qing             21    Female   tanq-wp21@student.tarc.edu.my   012-4561230     RDS
S1011        Lim Qiang            21    Male     limq-wp22@student.tarc.edu.my   018-78124510    RDS
S1012        Goh Qian             22    Female   gohq-wp21@student.tarc.edu.my   016-98235610    RDS
S1013        Tan Qiu              23    Male     tanqiu-wp23@student.tarc.edu.my 018-79461311    RDS
S1014        Loh Qi               21    Male     lohq-wp21@student.tarc.edu.my   017-79791313    RDS
S1015        Tan Yu               21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
S1016        Lim Yao              21    Male     limy-wp22@student.tarc.edu.my   017-75395120    RDS
S1017        Goh Yi               22    Female   gohy-wp21@student.tarc.edu.my   014-85285245    REE
S1018        Tan Yuan             23    Male     tany-wp21@student.tarc.edu.my   012-75486914    REE
S1019        Loh Ying             21    Female   lohy-wp21@student.tarc.edu.my   014-02175855    RDS
S1020        Tan Ping             21    Female   tanp-wp21@student.tarc.edu.my   017-4561230     RSD
S1021        Lim Peng             21    Male     limp-wp22@student.tarc.edu.my   012-78124510    RMM
S1022        Goh Pan              26    Female   gohp-wp21@student.tarc.edu.my   013-98235610    RIS
S1023        Tan Pai Pai          23    Male     tanpp-wp21@student.tarc.edu.my  014-79461311    RET
S1024        Loh Pu               22    FeMale   lohp-wp21@student.tarc.edu.my   016-79791313    RBS
S1025        Tan Ting             25    Female   tant-wp21@student.tarc.edu.my   019-4562585     RAN
S1026        Lim Tao              20    Male     limt-wp22@student.tarc.edu.my   011-75395120    RSE
S1027        Goh Ti               22    Female   goht-wp21@student.tarc.edu.my   016-85285245    RSE
S1028        Tan Tuan Tuan        24    Male     tantt-wp21@student.tarc.edu.my  013-75486914    RSE
S1029        Loh Ting Ting        27    Female   lohtt-wp21@student.tarc.edu.my  015-02175855    RME
S1030        Tan Xing             25    Female   tanx-wp21@student.tarc.edu.my   019-45327895    RAN
S1031        Lim Xiao             20    Male     limx-wp22@student.tarc.edu.my   011-15423601    RSE
S1032        Goh Xi               22    Female   gohx-wp21@student.tarc.edu.my   016-75192846    RSE
S1033        Tan Xia Xia          24    Male     tanxx-wp21@student.tarc.edu.my  013-85147320    RSE
S1034        Loh Xing Xing        27    Female   lohtt-wp21@student.tarc.edu.my  015-84527545    RME

Press enter to continue...
```

Figure 1.19: Sort student list based on student ID in ascending order

```
List of Sorted Students based on Student ID by Descending order:

-------------------------------------------------------------------------------------------------------------------------
Student ID   Name                 Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------------
S1034        Loh Xing Xing        27    Female   lohtt-wp21@student.tarc.edu.my  015-84527545    RME
S1033        Tan Xia Xia          24    Male     tanxx-wp21@student.tarc.edu.my  013-85147320    RSE
S1032        Goh Xi               22    Female   gohx-wp21@student.tarc.edu.my   016-75192846    RSE
S1031        Lim Xiao             20    Male     limx-wp22@student.tarc.edu.my   011-15423601    RSE
S1030        Tan Xing             25    Female   tanx-wp21@student.tarc.edu.my   019-45327895    RAN
S1029        Loh Ting Ting        27    Female   lohtt-wp21@student.tarc.edu.my  015-02175855    RME
S1028        Tan Tuan Tuan        24    Male     tantt-wp21@student.tarc.edu.my  013-75486914    RSE
S1027        Goh Ti               22    Female   goht-wp21@student.tarc.edu.my   016-85285245    RSE
S1026        Lim Tao              20    Male     limt-wp22@student.tarc.edu.my   011-75395120    RSE
S1025        Tan Ting             25    Female   tant-wp21@student.tarc.edu.my   019-4562585     RAN
S1024        Loh Pu               22    FeMale   lohp-wp21@student.tarc.edu.my   016-79791313    RBS
S1023        Tan Pai Pai          23    Male     tanpp-wp21@student.tarc.edu.my  014-79461311    RET
S1022        Goh Pan              26    Female   gohp-wp21@student.tarc.edu.my   013-98235610    RIS
S1021        Lim Peng             21    Male     limp-wp22@student.tarc.edu.my   012-78124510    RMM
S1020        Tan Ping             21    Female   tanp-wp21@student.tarc.edu.my   017-4561230     RSD
S1019        Loh Ying             21    Female   lohy-wp21@student.tarc.edu.my   014-02175855    RDS
S1018        Tan Yuan             23    Male     tany-wp21@student.tarc.edu.my   012-75486914    REE
S1017        Goh Yi               22    Female   gohy-wp21@student.tarc.edu.my   014-85285245    REE
S1016        Lim Yao              21    Male     limy-wp22@student.tarc.edu.my   017-75395120    RDS
S1015        Tan Yu               21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
S1014        Loh Qi               21    Male     lohq-wp21@student.tarc.edu.my   017-79791313    RDS
S1013        Tan Qiu              23    Male     tanqiu-wp23@student.tarc.edu.my 018-79461311    RDS
S1012        Goh Qian             22    Female   gohq-wp21@student.tarc.edu.my   016-98235610    RDS
S1011        Lim Qiang            21    Male     limq-wp22@student.tarc.edu.my   018-78124510    RDS
S1010        Tan Qing             21    Female   tanq-wp21@student.tarc.edu.my   012-4561230     RDS
S1009        Loh Jia Jia          21    Male     lohjj-wp21@student.tarc.edu.my  017-10201312    REE
S1008        Tan Jia Jun          23    Male     tanjj-wp21@student.tarc.edu.my  018-45623123    RMT
S1007        Goh Jia Jie          22    Male     gohjj-wp21@student.tarc.edu.my  016-45678956    RMT
S1006        Lim Jia Jia          21    Male     limjj-wp22@student.tarc.edu.my  018-123456789   RSW
S1005        Tan Ling Ling        21    Female   tanll-wp21@student.tarc.edu.my  012-12341234    RDS
S1004        Loh Jia Shou         21    Male     lohjs-wp21@student.tarc.edu.my  017-8319687     RME
S1003        Tan Hoong Guan       23    Male     tanhg-wp21@student.tarc.edu.my  018-2277975     RFN
S1002        Goh Boon Xiang       22    Male     gohbx-wp21@student.tarc.edu.my  016-34098124    RDS
S1001        Lim Hoi Yau          21    Male     limhy-wp22@student.tarc.edu.my  018-12326756    RSW
S1000        Tan Lock Kwan        21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS

Press enter to continue...
```

Figure 1.20: Sort student list based on student ID in descending order

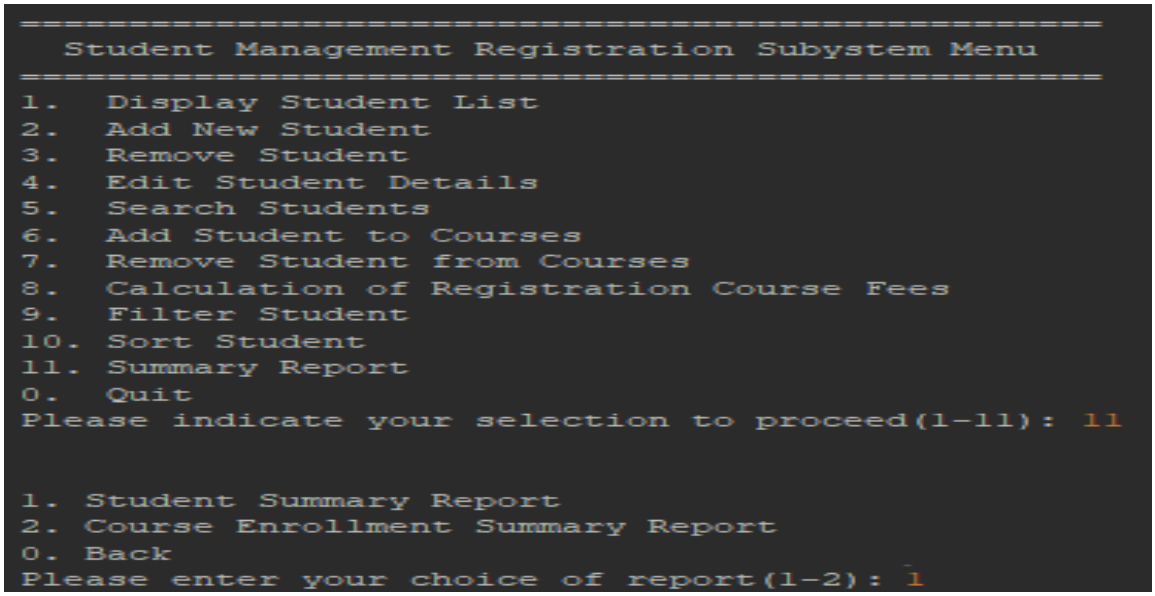
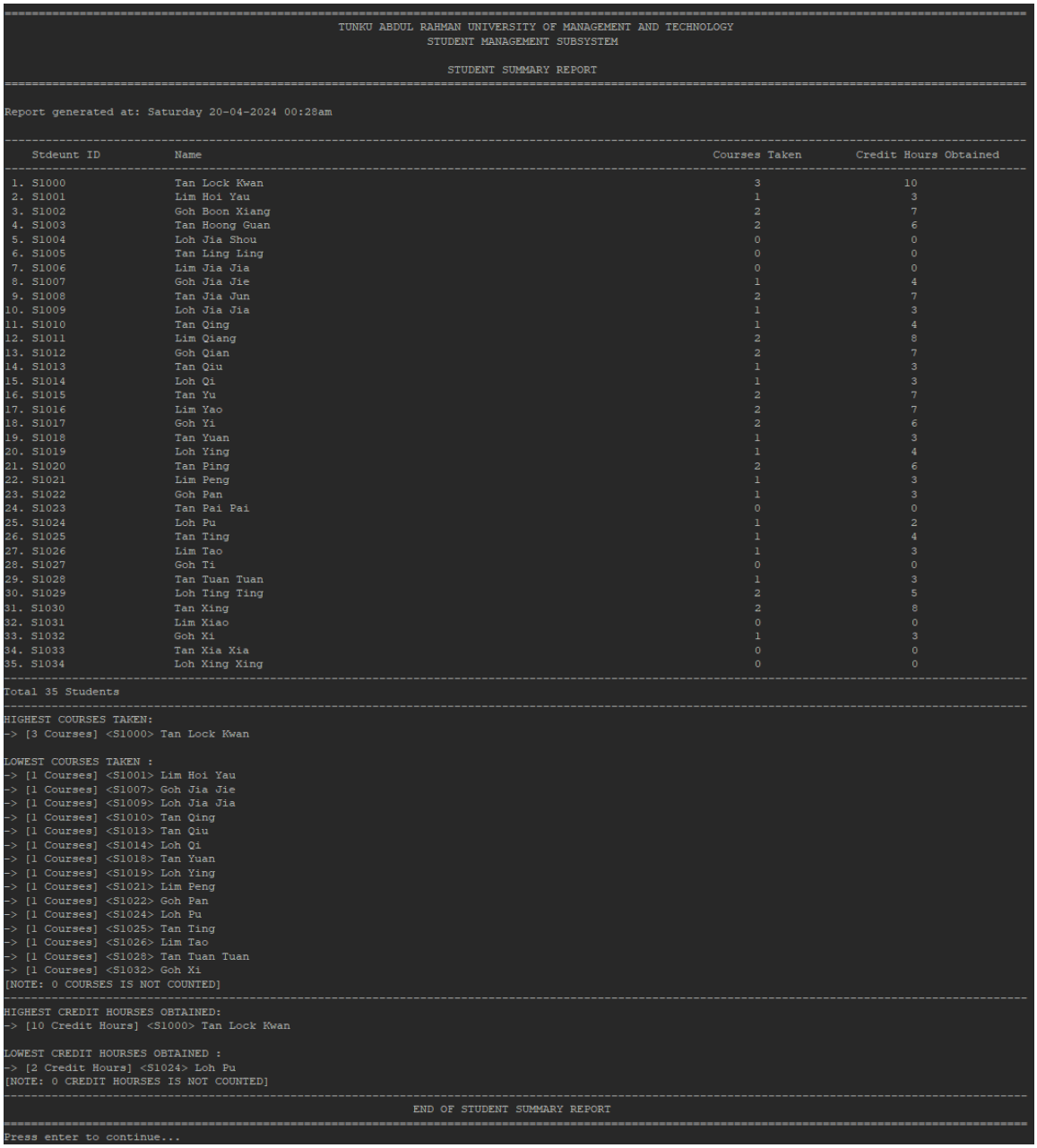


Figure 1.21: Summary Report Module

According to Figure 1.21, there are 2 types of summary reports for users to choose.



```
================================================================================================================================
                                        TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY
                                                STUDENT MANAGEMENT SUBSYSTEM

                                                    STUDENT SUMMARY REPORT
================================================================================================================================

Report generated at: Saturday 20-04-2024 00:28am

--------------------------------------------------------------------------------------------------------------------------------
    Stdeunt ID          Name                                                    Courses Taken       Credit Hours Obtained
--------------------------------------------------------------------------------------------------------------------------------
 1. S1000               Tan Lock Kwan                                           3                   10
 2. S1001               Lim Hoi Yau                                             1                   3
 3. S1002               Goh Boon Xiang                                          2                   7
 4. S1003               Tan Hoong Guan                                          2                   6
 5. S1004               Loh Jia Shou                                            0                   0
 6. S1005               Tan Ling Ling                                           0                   0
 7. S1006               Lim Jia Jia                                             0                   0
 8. S1007               Goh Jia Jie                                             1                   4
 9. S1008               Tan Jia Jun                                             2                   7
10. S1009               Loh Jia Jia                                             1                   3
11. S1010               Tan Qing                                                1                   4
12. S1011               Lim Qiang                                               2                   8
13. S1012               Goh Qian                                                2                   7
14. S1013               Tan Qiu                                                 1                   3
15. S1014               Loh Qi                                                  1                   3
16. S1015               Tan Yu                                                  2                   7
17. S1016               Lim Yao                                                 2                   7
18. S1017               Goh Yi                                                  2                   6
19. S1018               Tan Yuan                                                1                   3
20. S1019               Loh Ying                                                1                   4
21. S1020               Tan Ping                                                2                   6
22. S1021               Lim Peng                                                1                   3
23. S1022               Goh Pan                                                 1                   3
24. S1023               Tan Pai Pai                                             0                   0
25. S1024               Loh Pu                                                  1                   2
26. S1025               Tan Ting                                                1                   4
27. S1026               Lim Tao                                                 1                   3
28. S1027               Goh Ti                                                  0                   0
29. S1028               Tan Tuan Tuan                                           1                   3
30. S1029               Loh Ting Ting                                           2                   5
31. S1030               Tan Xing                                                2                   8
32. S1031               Lim Xiao                                                0                   0
33. S1032               Goh Xi                                                  1                   3
34. S1033               Tan Xia Xia                                             0                   0
35. S1034               Loh Xing Xing                                           0                   0
--------------------------------------------------------------------------------------------------------------------------------
Total 35 Students
--------------------------------------------------------------------------------------------------------------------------------
HIGHEST COURSES TAKEN:
-> [3 Courses] <S1000> Tan Lock Kwan

LOWEST COURSES TAKEN :
-> [1 Courses] <S1001> Lim Hoi Yau
-> [1 Courses] <S1007> Goh Jia Jie
-> [1 Courses] <S1009> Loh Jia Jia
-> [1 Courses] <S1010> Tan Qing
-> [1 Courses] <S1013> Tan Qiu
-> [1 Courses] <S1014> Loh Qi
-> [1 Courses] <S1018> Tan Yuan
-> [1 Courses] <S1019> Loh Ying
-> [1 Courses] <S1021> Lim Peng
-> [1 Courses] <S1022> Goh Pan
-> [1 Courses] <S1024> Loh Pu
-> [1 Courses] <S1025> Tan Ting
-> [1 Courses] <S1026> Lim Tao
-> [1 Courses] <S1028> Tan Tuan Tuan
-> [1 Courses] <S1032> Goh Xi
[NOTE: 0 COURSES IS NOT COUNTED]
--------------------------------------------------------------------------------------------------------------------------------
HIGHEST CREDIT HOURSES OBTAINED:
-> [10 Credit Hours] <S1000> Tan Lock Kwan

LOWEST CREDIT HOURSES OBTAINED :
-> [2 Credit Hours] <S1024> Loh Pu
[NOTE: 0 CREDIT HOURSES IS NOT COUNTED]
--------------------------------------------------------------------------------------------------------------------------------
                                                END OF STUDENT SUMMARY REPORT
================================================================================================================================
Press enter to continue...
```

Figure 1.21: Student Summary Report

```
================================================================================================================================================
                                                TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY
                                                        STUDENT MANAGEMENT SUBSYSTEM

                                                        COURSE ENROLLMENT SUMMARY REPORT
================================================================================================================================================

Report generated at: Saturday 20-04-2024 00:28am

------------------------------------------------------------------------------------------------------------------------------------------------
    Course Code     Course Name                                 Credit Hours      Total Students                Enrollent Status
------------------------------------------------------------------------------------------------------------------------------------------------
 1. BABS1233        Microbiology                                    2.0               1               Main:1  Resit:0  Repeat:0  Elective:0
 2. BACH1613        Physical Chemistry                              3.0               1               Main:0  Resit:0  Repeat:1  Elective:0
 3. BACH2233        Food Chemistry And Analysis                     4.0               3               Main:1  Resit:0  Repeat:0  Elective:2
 4. BACS1053        Database Management                             3.0               6               Main:4  Resit:2  Repeat:0  Elective:0
 5. BACS2026        Data Structure And Algorithms                   4.0               9               Main:6  Resit:2  Repeat:0  Elective:1
 6. BAIT1023        Web Design And Development                      2.0               0               Main:0  Resit:0  Repeat:0  Elective:0
 7. BAIT1043        System Analysis And Design                      3.0               0               Main:0  Resit:0  Repeat:0  Elective:0
 8. BAMS1413        Calculus And Algebra                            3.0               8               Main:3  Resit:1  Repeat:1  Elective:3
 9. BAMS2014        Linear Algebra                                  3.0               2               Main:1  Resit:0  Repeat:0  Elective:1
10. BGEE2614        Electrical Power Systems                        2.0               2               Main:2  Resit:0  Repeat:0  Elective:0
11. BJEL1013        English For Tertiary Studies                    3.0               4               Main:2  Resit:2  Repeat:0  Elective:0
12. BJEL1023        Academic English                                3.0               1               Main:1  Resit:0  Repeat:0  Elective:0
13. BTEE1513        Electrical Technology                           4.0               2               Main:1  Resit:1  Repeat:0  Elective:0
14. BTEE4033        Integrated Circuit Technology                   3.0               0               Main:0  Resit:0  Repeat:0  Elective:0
15. BTME4263        Heat Transfer                                   3.0               2               Main:1  Resit:1  Repeat:0  Elective:0
------------------------------------------------------------------------------------------------------------------------------------------------
Total 15 Courses
------------------------------------------------------------------------------------------------------------------------------------------------
TOP ENROLLED COURSE:
-> [9 Students] <BACS2026> Data Structure And Algorithms

LEAST ENROLLED COURSE :
-> [1 Students] <BABS1233> Microbiology
-> [1 Students] <BACH1613> Physical Chemistry
-> [1 Students] <BJEL1023> Academic English
[NOTE: 0 STUDENTS IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
COURSE WITH HIGHEST MAIN ENROLLMENT:
-> [6 Students] <BACS2026> Data Structure And Algorithms

COURSE WITH LOWEST MAIN ENROLLMENT:
-> [1 Students] <BABS1233> Microbiology
-> [1 Students] <BACH2233> Food Chemistry And Analysis
-> [1 Students] <BAMS2014> Linear Algebra
-> [1 Students] <BJEL1023> Academic English
-> [1 Students] <BTEE1513> Electrical Technology
-> [1 Students] <BTME4263> Heat Transfer
[NOTE: 0 STUDENTS IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
COURSE WITH HIGHEST RESIT ENROLLMENT:
-> [2 Students] <BACS1053> Database Management
-> [2 Students] <BACS2026> Data Structure And Algorithms
-> [2 Students] <BJEL1013> English For Tertiary Studies

COURSE WITH LOWEST RESIT ENROLLMENT:
-> [1 Students] <BAMS1413> Calculus And Algebra
-> [1 Students] <BTEE1513> Electrical Technology
-> [1 Students] <BTME4263> Heat Transfer
[NOTE: 0 STUDENTS IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
COURSE WITH HIGHEST REPEAT ENROLLMENT:
-> [1 Students] <BACH1613> Physical Chemistry
-> [1 Students] <BAMS1413> Calculus And Algebra

COURSE WITH LOWEST REPEAT ENROLLMENT:
-> [1 Students] <BACH1613> Physical Chemistry
-> [1 Students] <BAMS1413> Calculus And Algebra
[NOTE: 0 STUDENTS IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
COURSE WITH HIGHEST ELECTIVE ENROLLMENT:
-> [3 Students] <BAMS1413> Calculus And Algebra

COURSE WITH LOWEST ELECTIVE ENROLLMENT:
-> [1 Students] <BACS2026> Data Structure And Algorithms
-> [1 Students] <BAMS2014> Linear Algebra
[NOTE: 0 STUDENTS IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
                                                        END OF STUDENT SUMMARY REPORT
================================================================================================================================================
Press enter to continue...
```

Figure 1.22: Course Enrollment Summary Report

Name	Student ID	Prog / Tut.Grp	Signature
Loh Jia Shou	2213596	RDS2S2G3	*LOH*

Subsystem : Course Management Subsystem

1. **Source codes for Control classes**

```
package control;

import adt.*;
import boundary.CourseManagementUI;
import dao.*;
import entity.Course;
import entity.Programme;
import entity.courseProgramme;
import entity.Faculty;
import java.util.Scanner;
import utility.MessageUI;

/**
 *
 * @author jia shou
 */
public class CourseManagement {

    Scanner scanner = new Scanner(System.in);

    private ListInterface<Course> CourseList = new ArrayList<>();
    private ListInterface<Programme> ProgrammeList = new ArrayList<>();
    private ListInterface<courseProgramme> CourseProgrammeList = new ArrayList<>();
    private ListInterface<Faculty> FacultyList = new ArrayList<>();
    private CourseDAO CourseDAO = new CourseDAO();
    private CourseInitializer CourseInitializer = new CourseInitializer();
    private ProgrammeDAO ProgrammeDAO = new ProgrammeDAO();
    private ProgrammeInitializer ProgrammeInitializer = new ProgrammeInitializer();
                private courseProgrammeDAO CourseProgrammeDAO = new
courseProgrammeDAO();
            private courseProgrammeInitializer CourseProgrammeInitializer = new
courseProgrammeInitializer();
    private FacultyDAO FacultyDAO = new FacultyDAO();
    private FacultyInitializer FacultyInitializer = new FacultyInitializer();
    private CourseManagementUI CourseManagementUI = new CourseManagementUI();
    private ListInterface<Faculty> facultySortedList = new ArrayList<>();
    private ListInterface<Course> courseSortedList = new ArrayList<>();
```

```
    private ListInterface<Programme> programmeSortedList = new ArrayList<>();

    public CourseManagement() {
        CourseList = CourseDAO.retrieveFromFile();
        ProgrammeList = ProgrammeDAO.retrieveFromFile();
        CourseProgrammeList = CourseProgrammeDAO.retrieveFromFile();
        FacultyList = FacultyDAO.retrieveFromFile();
    }

    public static void main(String[] args) {
        CourseManagement courseManagement = new CourseManagement();
        courseManagement.runCourseManagement();
    }

    public void runCourseManagement() {
        int choice = 0;
        do {
            choice = CourseManagementUI.getMenuChoice();
            switch (choice) {
                case 0:
                    break;
                case 1:
                    addProgrammeToCourse(null, 'N');
                    break;
                case 2:
                    removeProgrammFromCourse();
                    break;
                case 3:
                    addNewCourse();
                    break;
                case 4:
                    removeProgrammFromCourse();
                    break;
                case 5:
                    searchCoursesBySemester();
                    break;
                case 6:
                    editCourse();
                    break;
                case 7:
                    listCourseForFaculty();
                    break;
                case 8:
                    listCourseForProgramme();
                    break;
                case 9:
```

```
                Report();
                break;
            }
        } while (choice != 0);
    }

    public void addProgrammeToCourse(String courseCode, char check) {
        String programmeCode = "";
        if (courseCode == null) {
            displayAllCourse(null);
            do {
                courseCode = CourseManagementUI.inputCourseCode();
                if (!findCourse(courseCode)) {
                    System.out.print("Invalid course!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        }
        check = 'N';
        displayAllProgramme(null);
        do {
            programmeCode = CourseManagementUI.inputProgrammeCode();
            if (!findProgramme(programmeCode)) {
                System.out.print("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
        check = 'N';
        for (int i = 1; i <= CourseProgrammeList.getNumberOfEntries(); i++) {
                                                                                    if
(programmeCode.equals(CourseProgrammeList.getEntry(i).getProgrammeCode())) {
                if (CourseProgrammeList.getEntry(i).getCourseCode().equals(courseCode)) {
                    check = 'Y';
                    break;
                }
            }
        }
        System.out.println();
        if (check == 'Y') {
            System.out.println("Record already exist!");
        } else {
                            courseProgramme newCourseProgramme = (courseProgramme)
```

```
CourseManagementUI.addProgrammetoCourse(programmeCode, courseCode);
            CourseProgrammeList.add(newCourseProgramme);
            CourseProgrammeDAO.saveToFile(CourseProgrammeList);
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void removeProgrammFromCourse() {
        char check = 'Y';
        String programmeCode = "programme";
        String courseCode;
        displayAllProgramme(null);
        while (check == 'Y') {
            programmeCode = CourseManagementUI.inputProgrammeCode();
            if (!findProgramme(programmeCode)) {
                System.out.print("Invalid programme!");
            } else {
                check = 'N';
            }
            System.out.println();
        }
        if (getCourseForProgramme(programmeCode).isEmpty()) {
            System.out.println("There is no record!");
        } else {
            displayCourseForProgramme(programmeCode);

            do {
                courseCode = CourseManagementUI.inputCourseCode();
                if (!findCourse(courseCode)) {
                    System.out.print("Invalid course!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check == 'N');
            System.out.print("Continue removing programme from course(Y/N): ");
            check = scanner.next().charAt(0);
            scanner.nextLine();
            int indexToRemove = -1;
            if (check == 'Y') {
                for (int i = 0; i < CourseProgrammeList.getNumberOfEntries() + 1; i++) {
                    if (CourseProgrammeList.getEntry(i) != null) {
                                                                        if
(CourseProgrammeList.getEntry(i).getCourseCode().equals(courseCode)                &&
CourseProgrammeList.getEntry(i).getProgrammeCode().equals(programmeCode)) {
```

```
                    indexToRemove = i;
                    break;
                }
            }
        }
        if (indexToRemove != -1) {
            CourseProgrammeList.remove(indexToRemove);
            CourseProgrammeDAO.saveToFile(CourseProgrammeList);
                System.out.println("" + programmeCode + " had been removed from " +
courseCode + " successfully!");
        } else {
                System.out.println("Failed to remove " + programmeCode + " from " +
courseCode + " !");
        }
      } else {
        System.out.println("Programme remove cancelled!");
      }
    }
    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
  }

  public void addNewCourse() {
    String courseCode = CourseManagementUI.inputCourseCode();
    if (findCourse(courseCode)) {
      System.out.println("Course already exist!");
      System.out.print("\nPress enter to continue...");
      scanner.nextLine();
    } else {
      String courseName = CourseManagementUI.inputCourseName();
      double creditHour = CourseManagementUI.inputCreditHour();
      String academicYear = CourseManagementUI.inputAcademicYear();
      System.out.println();
                                                    Course newCourse = (Course)
CourseManagementUI.addNewCourse(courseCode, courseName, creditHour,
academicYear);
      CourseList.add(newCourse);
      CourseDAO.saveToFile(CourseList);
      addProgrammeToCourse(courseCode, 'N');
    }
  }

  public void searchCoursesBySemester() {
    String academicYear = CourseManagementUI.inputAcademicYear();
    displayAllCourse(academicYear);
    System.out.print("\nPress enter to continue...");
```

```
        scanner.nextLine();
    }

    public void editCourse() {
        System.out.println();
        String editCourseCode = CourseManagementUI.inputCourseCode();

        int indexToEdit = -1;
        for (int i = 1; i < CourseList.getNumberOfEntries() + 1; i++) {
            if (CourseList.getEntry(i).getCourseCode().equals(editCourseCode)) {
                indexToEdit = i;
                break;
            }
        }
        if (indexToEdit != -1) {
            int typeEditChoice = 0;
            Course editCourse = CourseList.getEntry(indexToEdit);
            do {
                typeEditChoice = CourseManagementUI.getTypeEditChoice();
                switch (typeEditChoice) {
                    case 0:
                        MessageUI.displayExitMessage();
                        break;
                    case 1:
                        String oldCourseName = editCourse.getCourseName();
                            System.out.println("Old Course Name for " + editCourseCode + ": " +
oldCourseName);
                        editCourse.setCourseName(CourseManagementUI.inputCourseName());
                                System.out.println(oldCourseName + " had been changed to " +
editCourse.getCourseName());
                        break;
                    case 2:
                        double oldCreditHour = editCourse.getCreditHour();
                            System.out.println("Old Credit Hour for " + editCourseCode + ": " +
oldCreditHour);
                        editCourse.setCreditHour(CourseManagementUI.inputCreditHour());
                          System.out.println(oldCreditHour + " credit hour for " + editCourseCode
+ " had been changed to " + editCourse.getCreditHour());
                        break;
                    case 3:
                        String oldAcademicYear = editCourse.getAcademicYear();
                            System.out.println("Old Academic Year for " + editCourseCode + ": " +
oldAcademicYear);

editCourse.setAcademicYear(CourseManagementUI.inputAcademicYear());
                                System.out.println(oldAcademicYear + " academic year for " +
```

```
editCourseCode + " had been changed to " + editCourse.getAcademicYear());
                    break;
                }
            } while (typeEditChoice != 0);
            CourseList.replace(indexToEdit, editCourse);
            CourseDAO.saveToFile(CourseList);
        } else {
            System.out.println("Course with course code " + editCourseCode + " is not found
in the list!\n");
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void listCourseForFaculty() {
        char check = 'N';
        String FacultyCode;
        displayAllFaculty(null);
        do {
            FacultyCode = CourseManagementUI.inputFacultyCode();
            if (!findFaculty(FacultyCode)) {
                System.out.println("Invalid faculty!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
        displayAllFaculty(FacultyCode);
        displayCourseForFaculty(FacultyCode);
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void listCourseForProgramme() {
        char check = 'N';
        String programmeCode;
        displayAllProgramme(null);
        do {
            programmeCode = CourseManagementUI.inputProgrammeCode();
            if (!findProgramme(programmeCode)) {
                System.out.print("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
```

```
    displayCourseForProgramme(programmeCode);
    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

public void Report() {
    int choice = -1;
    do {
        choice = CourseManagementUI.reportChoice();
        switch (choice) {
            case 0:
                MessageUI.displayExitMessage();
                break;
            case 1:
                displayFacultyReport();
                break;
            case 2:
                displayProgrammeReport();
                break;
            case 3:
                displayCourseReport();
                break;
        }
    } while (choice != 0);
}

public String getAllCourse(String academicYear) {
    String outputStr = "";
    if (academicYear == null) {
        for (int i = 1; i <= CourseList.getNumberOfEntries(); i++) {
            outputStr += CourseList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= CourseList.getNumberOfEntries(); i++) {
            if (academicYear.equals(CourseList.getEntry(i).getAcademicYear())) {
                outputStr += CourseList.getEntry(i) + "\n";
            }
        }
    }
    return outputStr;
}

public boolean findCourse(String courseCode) {
    for (int i = 0; i < CourseList.getNumberOfEntries() + 1; i++) {
        if (CourseList.getEntry(i) != null) {
            if (CourseList.getEntry(i).getCourseCode().equals(courseCode)) {
```

```
                return true;
            }
        }
    }
    return false;
}

public boolean findProgramme(String programmeCode) {
    for (int i = 0; i < ProgrammeList.getNumberOfEntries() + 1; i++) {
        if (ProgrammeList.getEntry(i) != null) {
            if (ProgrammeList.getEntry(i).getProgrammeCode().equals(programmeCode))
{
                return true;
            }
        }
    }
    return false;
}

public boolean findFaculty(String FacultyCode) {
    for (int i = 0; i < FacultyList.getNumberOfEntries() + 1; i++) {
        if (FacultyList.getEntry(i) != null) {
            if (FacultyList.getEntry(i).getFacultyCode().equals(FacultyCode)) {
                return true;
            }
        }
    }
    return false;
}

public String getAllProgramme(String programmeCode) {
    String outputStr = "";
    if (programmeCode == null) {
        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
            outputStr += ProgrammeList.getEntry(i) + "\n";

        }
    } else {
        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
            if (programmeCode.equals(ProgrammeList.getEntry(i).getProgrammeCode()))
{
                outputStr += ProgrammeList.getEntry(i) + "\n";
            }
        }
    }
    return outputStr;
```

```
    }

    public String getCourseForProgramme(String programmeCode) {
        String outputStr = "";
        String courseCode = "";

        for (int i = 1; i <= CourseProgrammeList.getNumberOfEntries(); i++) {
                                                                                    if
(programmeCode.equals(CourseProgrammeList.getEntry(i).getProgrammeCode())) {
                courseCode = CourseProgrammeList.getEntry(i).getCourseCode();
            }
            for (int j = 1; j <= CourseList.getNumberOfEntries(); j++) {
                if (courseCode.equals(CourseList.getEntry(j).getCourseCode())) {
                    courseCode = "";
                    outputStr += CourseList.getEntry(j) + "\n";
                }
            }
        }
        return outputStr;
    }

    public String getAllFaculty(String FacultyCode) {
        String outputStr = "";
        if (FacultyCode == null) {
            for (int i = 1; i <= FacultyList.getNumberOfEntries(); i++) {
                outputStr += FacultyList.getEntry(i) + "\n";

            }
        } else {
            for (int i = 1; i <= FacultyList.getNumberOfEntries(); i++) {
                if (FacultyCode.equals(FacultyList.getEntry(i).getFacultyCode())) {
                    outputStr += FacultyList.getEntry(i) + "\n";
                }
            }
        }
        return outputStr;
    }

    public String getCourseForFaculty(String FacultyCode) {
        ListInterface<String> TEMPCourse = new ArrayList<>();
        ListInterface<String> TEMPProgramme = new ArrayList<>();
        String outputStr = "";
        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
            if (FacultyCode.equals(ProgrammeList.getEntry(i).getFacultyCode())) {
                TEMPProgramme.add(ProgrammeList.getEntry(i).getProgrammeCode());
            }
```

```
        }
        for (int i = 1; i <= TEMPProgramme.getNumberOfEntries(); i++) {
            for (int j = 1; j <= CourseProgrammeList.getNumberOfEntries(); j++) {
                                                                                    if
(TEMPProgramme.getEntry(i).equals(CourseProgrammeList.getEntry(j).getProgrammeC
ode())) {
                    TEMPCourse.add(CourseProgrammeList.getEntry(j).getCourseCode());
                }
            }
        }
        for (int i = 1; i <= TEMPCourse.getNumberOfEntries(); i++) {
            String CurrentCourseCode = TEMPCourse.getEntry(i);
            for (int j = i + 1; j <= TEMPCourse.getNumberOfEntries(); j++) {
                if (CurrentCourseCode.equals(TEMPCourse.getEntry(j))) {
                    TEMPCourse.remove(j);
                    j--;
                }
            }
        }
        for (int i = 1; i <= TEMPCourse.getNumberOfEntries(); i++) {
            for (int j = 1; j <= CourseList.getNumberOfEntries(); j++) {
                if (TEMPCourse.getEntry(i).equals(CourseList.getEntry(j).getCourseCode())) {
                    outputStr += CourseList.getEntry(j) + "\n";
                }
            }
        }
        return outputStr;
    }

    public String FacultyReport() {
        sortFaculty();
        ListInterface<String> outputStr = new ArrayList<>();
        ListInterface<String> TEMPCourse = new ArrayList<>();
        ListInterface<String> TEMPProgramme = new ArrayList<>();
        ListInterface<String> TEMPFaculty = new ArrayList<>();
        ListInterface<Integer> facultyProgrammeCount = new ArrayList<>();
        ListInterface<Integer> facultyCourseCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int max = 0;
        int min = 999;

        int x = 1;
        for (int i = 1; i <= facultySortedList.getNumberOfEntries(); i++) {
            TEMPFaculty.add(facultySortedList.getEntry(i).getFacultyCode());
        }
```

```
        while (x <= TEMPFaculty.getNumberOfEntries()) {
            for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
                                                                                    if
(TEMPFaculty.getEntry(x).equals(ProgrammeList.getEntry(i).getFacultyCode())) {
                    TEMPProgramme.add(ProgrammeList.getEntry(i).getProgrammeCode());
                }
            }
            facultyProgrammeCount.add(TEMPProgramme.getNumberOfEntries());
            for (int i = 1; i <= CourseProgrammeList.getNumberOfEntries(); i++) {
                for (int j = 1; j <= TEMPProgramme.getNumberOfEntries(); j++) {
                                                                                    if
(CourseProgrammeList.getEntry(i).getProgrammeCode().equals(TEMPProgramme.getEn
try(j))) {
                        TEMPCourse.add(CourseProgrammeList.getEntry(i).getCourseCode());
                    }
                }
            }
            for (int i = 1; i <= TEMPCourse.getNumberOfEntries(); i++) {
                String CurrentCourseCode = TEMPCourse.getEntry(i);
                for (int j = i + 1; j <= TEMPCourse.getNumberOfEntries(); j++) {
                    if (CurrentCourseCode.equals(TEMPCourse.getEntry(j))) {
                        TEMPCourse.remove(j);
                        j--;
                    }
                }
            }
            facultyCourseCount.add(TEMPCourse.getNumberOfEntries());
            TEMPCourse.clear();
            TEMPProgramme.clear();
            x += 1;
        }

        for (int i = 1; i <= facultySortedList.getNumberOfEntries(); i++) {
            if (i >= 10) {
                                        outputStr.add(i + ". " + facultySortedList.getEntry(i) +
facultyProgrammeCount.getEntry(i) + " ".repeat(22) + facultyCourseCount.getEntry(i) +
"\n");
            } else {
                                    outputStr.add(" " + i + ". " + facultySortedList.getEntry(i) +
facultyProgrammeCount.getEntry(i) + " ".repeat(22) + facultyCourseCount.getEntry(i) +
"\n");
            }
        }

        outputStr.add("-".repeat(150)
```

```
            + "\nTotal " + TEMPFaculty.getNumberOfEntries() + " Faculties");

        for (int i = 1; i <= facultyProgrammeCount.getNumberOfEntries(); i++) {
            if (facultyProgrammeCount.getEntry(i) != 0) {
                if (facultyProgrammeCount.getEntry(i) > max) {
                    max = facultyProgrammeCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (facultyProgrammeCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (facultyProgrammeCount.getEntry(i) < min) {
                    min = facultyProgrammeCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (facultyProgrammeCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        outputStr.add("-".repeat(150) + "\n" + "HIGHEST PROGRAMMES OFFERED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + max + " Programmes] " + "<" +
facultySortedList.getEntry(maxIndex.getEntry(i)).getFacultyCode() + "> " +
facultySortedList.getEntry(maxIndex.getEntry(i)).getFacultyName());
        }

        outputStr.add("\nLOWEST PROGRAMMES OFFERED: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Programmes] " + "<" +
facultySortedList.getEntry(minIndex.getEntry(i)).getFacultyCode() + "> " +
facultySortedList.getEntry(minIndex.getEntry(i)).getFacultyName());
        }

        outputStr.add("\n" + "-".repeat(150));

        max = 0;
        min = 999;
        maxIndex.clear();
        minIndex.clear();

        for (int i = 1; i <= facultyCourseCount.getNumberOfEntries(); i++) {
            if (facultyCourseCount.getEntry(i) != 0) {
```

```
            if (facultyCourseCount.getEntry(i) > max) {
                max = facultyCourseCount.getEntry(i);
                maxIndex.clear();
                maxIndex.add(i);
            } else if (facultyCourseCount.getEntry(i) == max) {
                maxIndex.add(i);
            }
            if (facultyCourseCount.getEntry(i) < min) {
                min = facultyCourseCount.getEntry(i);
                minIndex.clear();
                minIndex.add(i);
            } else if (facultyCourseCount.getEntry(i) == min) {
                minIndex.add(i);
            }
        }
    }

    outputStr.add("HIGHEST COURSES OFFERED: ");

    for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                        outputStr.add("-> [" + max + " Courses] " + "<" +
facultySortedList.getEntry(maxIndex.getEntry(i)).getFacultyCode() + "> " +
facultySortedList.getEntry(maxIndex.getEntry(i)).getFacultyName());
    }

    outputStr.add("\nLOWEST COURSES OFFERED: ");

    for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                        outputStr.add("-> [" + min + " Courses] " + "<" +
facultySortedList.getEntry(minIndex.getEntry(i)).getFacultyCode() + "> " +
facultySortedList.getEntry(minIndex.getEntry(i)).getFacultyName());
    }

    outputStr.add("\n[NOTE: 0 COURSES IS NOT COUNTED]\n"
        + "-".repeat(150) + "\n"
        + " ".repeat(60) + "END OF FACULTY SUMMARY REPORT\n"
        + "=".repeat(150));

    return outputStr.toString();
}

public String CourseReport() {
    sortCourse();
    ListInterface<String> outputStr = new ArrayList<>();
    ListInterface<String> TEMPCourse = new ArrayList<>();
    ListInterface<String> TEMPProgramme = new ArrayList<>();
```

```
        ListInterface<String> TEMPFaculty = new ArrayList<>();
        ListInterface<Integer> courseProgrammeCount = new ArrayList<>();
        ListInterface<Integer> courseFacultyCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int max = 0;
        int min = 999;

        for (int i = 1; i <= courseSortedList.getNumberOfEntries(); i++) {
            TEMPCourse.add(courseSortedList.getEntry(i).getCourseCode());
        }

        for (int i = 1; i <= TEMPCourse.getNumberOfEntries(); i++) {

            for (int j = 1; j <= CourseProgrammeList.getNumberOfEntries(); j++) {
                                                                                if
(TEMPCourse.getEntry(i).equals(CourseProgrammeList.getEntry(j).getCourseCode())) {

TEMPProgramme.add(CourseProgrammeList.getEntry(j).getProgrammeCode());
                }
            }

            courseProgrammeCount.add(TEMPProgramme.getNumberOfEntries());

            for (int j = 1; j <= TEMPProgramme.getNumberOfEntries(); j++) {
                for (int k = 1; k <= ProgrammeList.getNumberOfEntries(); k++) {
                                                                                if
(TEMPProgramme.getEntry(j).equals(ProgrammeList.getEntry(k).getProgrammeCode())
) {
                        TEMPFaculty.add(ProgrammeList.getEntry(k).getFacultyCode());
                    }
                }
            }

            for (int j = 1; j <= TEMPFaculty.getNumberOfEntries(); j++) {
                String CurrentFacultyCode = TEMPFaculty.getEntry(j);
                for (int k = j + 1; k <= TEMPFaculty.getNumberOfEntries(); k++) {
                    if (CurrentFacultyCode.equals(TEMPFaculty.getEntry(k))) {
                        TEMPFaculty.remove(k);
                        k--;
                    }
                }
            }

            courseFacultyCount.add(TEMPFaculty.getNumberOfEntries());
            TEMPProgramme.clear();
```

```
            TEMPFaculty.clear();
        }

        for (int i = 1; i <= courseSortedList.getNumberOfEntries(); i++) {
            if (i >= 10) {
                        outputStr.add(i + ". " + courseSortedList.getEntry(i) + " ".repeat(15) +
courseProgrammeCount.getEntry(i) + " ".repeat(17) + courseFacultyCount.getEntry(i) +
"\n");
            } else {
                    outputStr.add(" " + i + ". " + courseSortedList.getEntry(i) + " ".repeat(15) +
courseProgrammeCount.getEntry(i) + " ".repeat(17) + courseFacultyCount.getEntry(i) +
"\n");
            }
        }

        outputStr.add("-".repeat(150)
                + "\nTotal " + TEMPCourse.getNumberOfEntries() + " Courses");

        for (int i = 1; i <= courseProgrammeCount.getNumberOfEntries(); i++) {
            if (courseProgrammeCount.getEntry(i) != 0) {
                if (courseProgrammeCount.getEntry(i) > max) {
                    max = courseProgrammeCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (courseProgrammeCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (courseProgrammeCount.getEntry(i) < min) {
                    min = courseProgrammeCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (courseProgrammeCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        outputStr.add("-".repeat(150) + "\n" + "HIGHEST PROGRAMMES OFFERED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + max + " Programmes] " + "<" +
courseSortedList.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
courseSortedList.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        outputStr.add("\nLOWEST PROGRAMMES OFFERED: ");
```

```
        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Programmes] " + "<" +
courseSortedList.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
courseSortedList.getEntry(minIndex.getEntry(i)).getCourseName());
        }

            outputStr.add("\n[NOTE: 0 PROGRAMMES IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;
        maxIndex.clear();
        minIndex.clear();

        for (int i = 1; i <= courseFacultyCount.getNumberOfEntries(); i++) {
           if (courseFacultyCount.getEntry(i) != 0) {
              if (courseFacultyCount.getEntry(i) > max) {
                 max = courseFacultyCount.getEntry(i);
                 maxIndex.clear();
                 maxIndex.add(i);
              } else if (courseFacultyCount.getEntry(i) == max) {
                 maxIndex.add(i);
              }
              if (courseFacultyCount.getEntry(i) < min) {
                 min = courseFacultyCount.getEntry(i);
                 minIndex.clear();
                 minIndex.add(i);
              } else if (courseFacultyCount.getEntry(i) == min) {
                 minIndex.add(i);
              }
           }
        }

        outputStr.add("HIGHEST FACULTIES OFFERED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + max + " Faculties] " + "<" +
courseSortedList.getEntry(maxIndex.getEntry(i)).getCourseCode() + "> " +
courseSortedList.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        outputStr.add("\nLOWEST FACULTIES OFFERED: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Faculties] " + "<" +
```

```
courseSortedList.getEntry(minIndex.getEntry(i)).getCourseCode()     +     ">     "     +
courseSortedList.getEntry(minIndex.getEntry(i)).getCourseName());
        }

        outputStr.add("\n[NOTE: 0 FACULTIES IS NOT COUNTED]\n"
                + "-".repeat(150) + "\n"
                + " ".repeat(60) + "END OF COURSE SUMMARY REPORT\n"
                + "=".repeat(150));

        return outputStr.toString();
    }

    public String ProgrammeReport() {
        sortProgramme();
        ListInterface<String> outputStr = new ArrayList<>();
        ListInterface<String> TEMPCourse = new ArrayList<>();
        ListInterface<String> TEMPProgramme = new ArrayList<>();
        ListInterface<String> TEMPFaculty = new ArrayList<>();
        ListInterface<Integer> programmeCourseCount = new ArrayList<>();
        ListInterface<Integer> programmeFacultyCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int max = 0;
        int min = 999;

        for (int i = 1; i <= programmeSortedList.getNumberOfEntries(); i++) {
            TEMPProgramme.add(programmeSortedList.getEntry(i).getProgrammeCode());
        }

        for (int i = 1; i <= TEMPProgramme.getNumberOfEntries(); i++) {
            for (int j = 1; j <= CourseProgrammeList.getNumberOfEntries(); j++) {
                                                                                    if
(TEMPProgramme.getEntry(i).equals(CourseProgrammeList.getEntry(j).getProgrammeC
ode())) {
                    TEMPCourse.add(CourseProgrammeList.getEntry(j).getCourseCode());
                }
            }
            programmeCourseCount.add(TEMPCourse.getNumberOfEntries());
            TEMPCourse.clear();

            for (int j = 1; j <= ProgrammeList.getNumberOfEntries(); j++) {
                                                                                    if
(TEMPProgramme.getEntry(i).equals(ProgrammeList.getEntry(j).getProgrammeCode()))
{
                    TEMPFaculty.add(ProgrammeList.getEntry(j).getFacultyCode());
                }
```

```
            }
            programmeFacultyCount.add(TEMPFaculty.getNumberOfEntries());
            TEMPFaculty.clear();
        }

        for (int i = 1; i <= programmeSortedList.getNumberOfEntries(); i++) {
            if (i >= 10) {
                    outputStr.add(i + ". " + programmeSortedList.getEntry(i) + " ".repeat(10) +
programmeCourseCount.getEntry(i)          +          "          ".repeat(20)          +
programmeFacultyCount.getEntry(i) + "\n");
            } else {
                outputStr.add(" " + i + ". " + programmeSortedList.getEntry(i) + " ".repeat(10)
+       programmeCourseCount.getEntry(i)          +          "          ".repeat(20)          +
programmeFacultyCount.getEntry(i) + "\n");
            }
        }

        outputStr.add("-".repeat(150)
                + "\nTotal " + TEMPProgramme.getNumberOfEntries() + " Programmes");

        for (int i = 1; i <= programmeCourseCount.getNumberOfEntries(); i++) {
            if (programmeCourseCount.getEntry(i) != 0) {
                if (programmeCourseCount.getEntry(i) > max) {
                    max = programmeCourseCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (programmeCourseCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (programmeCourseCount.getEntry(i) < min) {
                    min = programmeCourseCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (programmeCourseCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        outputStr.add("-".repeat(150) + "\n" + "HIGHEST COURSES OFFERED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                    outputStr.add("->  [" + max + "  Courses]  " + "<" +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeName());
        }
```

```
        outputStr.add("\nLOWEST COURSES OFFERED: ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Courses] " + "<" +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeName());
        }

                outputStr.add("\n[NOTE: 0 COURSES IS NOT COUNTED]" + "\n" +
"-".repeat(150));

        max = 0;
        min = 999;
        maxIndex.clear();
        minIndex.clear();

        for (int i = 1; i <= programmeFacultyCount.getNumberOfEntries(); i++) {
           if (programmeFacultyCount.getEntry(i) != 0) {
              if (programmeFacultyCount.getEntry(i) > max) {
                 max = programmeFacultyCount.getEntry(i);
                 maxIndex.clear();
                 maxIndex.add(i);
              } else if (programmeFacultyCount.getEntry(i) == max) {
                 maxIndex.add(i);
              }
              if (programmeFacultyCount.getEntry(i) < min) {
                 min = programmeFacultyCount.getEntry(i);
                 minIndex.clear();
                 minIndex.add(i);
              } else if (programmeFacultyCount.getEntry(i) == min) {
                 minIndex.add(i);
              }
           }
        }

        outputStr.add("HIGHEST FACULTIES OFFERED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + max + " Faculties] " + "<" +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeName());
        }

        outputStr.add("\nLOWEST FACULTIES OFFERED: ");
```

```
        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Faculties] " + "<" +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeName());
        }

        outputStr.add("\n[NOTE: 0 FACULTIES IS NOT COUNTED]\n"
                + "-".repeat(150) + "\n"
                + " ".repeat(60) + "END OF PROGRAMME SUMMARY REPORT\n"
                + "=".repeat(150));

        return outputStr.toString();
    }

    public void sortFaculty() {
        facultySortedList.clear();
        for (int i = 1; i <= FacultyList.getNumberOfEntries(); i++) {
            facultySortedList.add(FacultyList.getEntry(i));
        }
        String criteria = null;
        String sortOrder = null;
        int sortType;
        int sortDetail = CourseManagementUI.getFacultySortDetailsChoice();
        switch (sortDetail) {
            case 0:
                MessageUI.displayExitMessage();
                break;
            case 1:
                criteria = "Faculty Code";
                sortType = CourseManagementUI.inputSortType();
                switch (sortType) {
                    case 1:
                        sortOrder = "Ascending";
                        facultySortedList.sortByAscending(faculty -> faculty.getFacultyCode());
                        break;
                    case 2:
                        sortOrder = "Descending";
                        facultySortedList.sortByDescending(faculty -> faculty.getFacultyCode());
                        break;
                }
                break;
            case 2:
                criteria = "Faculty Name";
                sortType = CourseManagementUI.inputSortType();
                switch (sortType) {
                    case 1:
```

```
                sortOrder = "Ascending";
                facultySortedList.sortByAscending(faculty -> faculty.getFacultyName());
                break;
            case 2:
                sortOrder = "Descending";
                facultySortedList.sortByDescending(faculty -> faculty.getFacultyName());
                break;
        }
        break;

    default:
        MessageUI.displayInvalidChoiceMessage();
  }
}

public void sortCourse() {
  courseSortedList.clear();
  for (int i = 1; i <= CourseList.getNumberOfEntries(); i++) {
    courseSortedList.add(CourseList.getEntry(i));
  }
  String criteria = null;
  String sortOrder = null;
  int sortType;
  int sortDetail = CourseManagementUI.getCourseSortDetailsChoice();
  switch (sortDetail) {
    case 0:
      MessageUI.displayExitMessage();
      break;
    case 1:
      criteria = "Course Code";
      sortType = CourseManagementUI.inputSortType();
      switch (sortType) {
        case 1:
          sortOrder = "Ascending";
          courseSortedList.sortByAscending(course -> course.getCourseCode());
          break;
        case 2:
          sortOrder = "Descending";
          courseSortedList.sortByDescending(course -> course.getCourseCode());
          break;
      }
      break;
    case 2:
      criteria = "Course Name";
      sortType = CourseManagementUI.inputSortType();
      switch (sortType) {
```

```
                case 1:
                    sortOrder = "Ascending";
                    courseSortedList.sortByAscending(course -> course.getCourseName());
                    break;
                case 2:
                    sortOrder = "Descending";
                    courseSortedList.sortByDescending(course -> course.getCourseName());
                    break;
            }
            break;

        default:
            MessageUI.displayInvalidChoiceMessage();
    }
}

public void sortProgramme() {
    programmeSortedList.clear();
    for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
        programmeSortedList.add(ProgrammeList.getEntry(i));
    }
    String criteria = null;
    String sortOrder = null;
    int sortType;
    int sortDetail = CourseManagementUI.getProgrammeSortDetailsChoice();
    switch (sortDetail) {
        case 0:
            MessageUI.displayExitMessage();
            break;
        case 1:
            criteria = "Programme Code";
            sortType = CourseManagementUI.inputSortType();
            switch (sortType) {
                case 1:
                    sortOrder = "Ascending";
                                        programmeSortedList.sortByAscending(programme  ->
programme.getProgrammeCode());
                    break;
                case 2:
                    sortOrder = "Descending";
                                        programmeSortedList.sortByDescending(programme  ->
programme.getProgrammeCode());
                    break;
            }
            break;
        case 2:
```

```
            criteria = "Programme Name";
            sortType = CourseManagementUI.inputSortType();
            switch (sortType) {
               case 1:
                  sortOrder = "Ascending";
                                    programmeSortedList.sortByAscending(programme  ->
programme.getProgrammeName());
                  break;
               case 2:
                  sortOrder = "Descending";
                                    programmeSortedList.sortByDescending(programme  ->
programme.getProgrammeName());
                  break;
            }
            break;

         default:
            MessageUI.displayInvalidChoiceMessage();
      }
   }

   public void displayAllCourse(String academicYear) {
      CourseManagementUI.listAllCourse(getAllCourse(academicYear));
   }

   public void displayAllProgramme(String programmeCode) {
      CourseManagementUI.listAllProgramme(getAllProgramme(programmeCode));
   }

   public void displayCourseForProgramme(String programmeCode) {
      CourseManagementUI.listAllCourse(getCourseForProgramme(programmeCode));
   }

   public void displayAllFaculty(String FacultyCode) {
      CourseManagementUI.listAllFaculty(getAllFaculty(FacultyCode));
   }

   public void displayCourseForFaculty(String FacultyCode) {
      CourseManagementUI.listAllCourse(getCourseForFaculty(FacultyCode));
   }

   public void displayFacultyReport() {
      CourseManagementUI.FacultyReport(FacultyReport());
   }

   public void displayCourseReport() {
```

```
        CourseManagementUI.CourseReport(CourseReport());
    }

    public void displayProgrammeReport() {
        CourseManagementUI.ProgrammeReport(ProgrammeReport());
    }

}
```

2. **Screenshots**

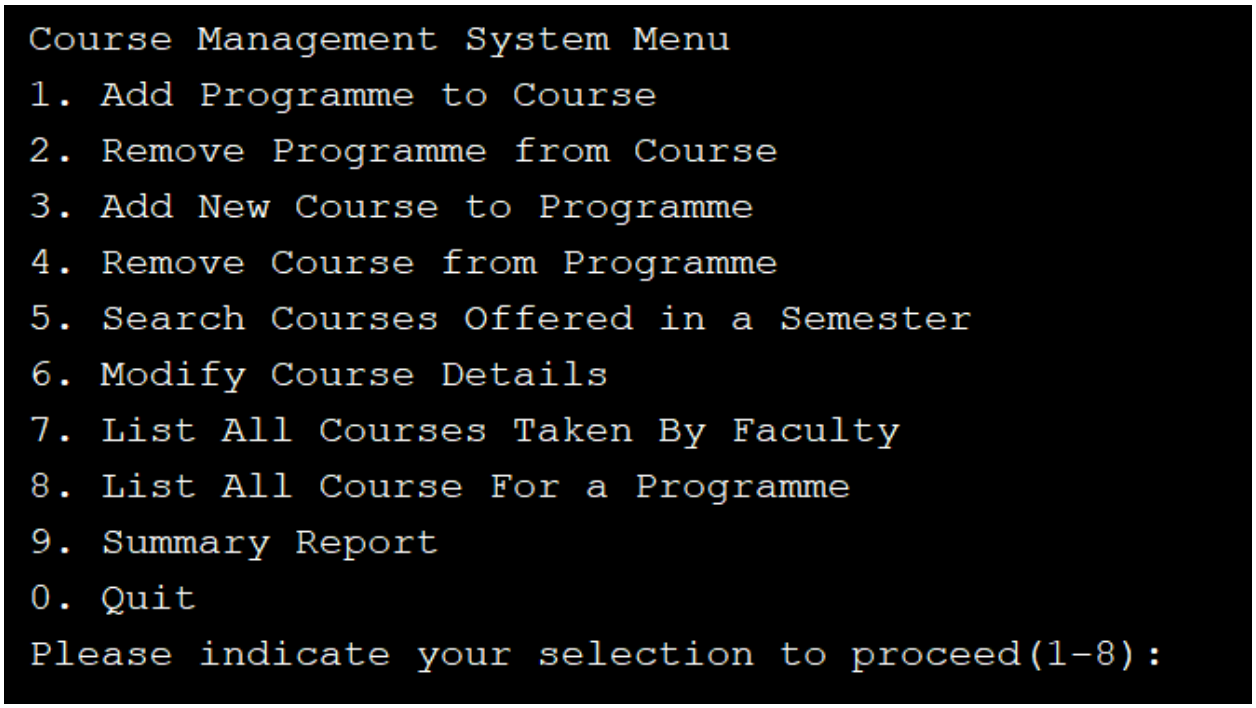


Figure 2.1 Menu of the Course Management Subsystem

Figure 2.1 shows the menu of the Course Management Subsystem and prompts the user for input options to proceed.

```
Please indicate your selection to proceed(1-8): 1

Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms           4.00            202401
BAIT1023        Web Design And Development              2.00            202401
BAIT1043        System Analysis And Design              3.00            202401
BABS1233        Microbiology                            2.00            202401
BACH2233        Food Chemistry And Analysis             4.00            202401
BACH1613        Physical Chemistry                      3.00            202401
BTME4263        Heat Transfer                           3.00            202401
BGEE2614        Electrical Power Systems                2.00            202401
BTEE1513        Electrical Technology                   4.00            202401
BTEE4033        Integrated Circuit Technology           3.00            202401
BJEL1023        Academic English                        3.00            202401
BJEL1013        English For Tertiary Studies            3.00            202401
BAMS1413        Calculus And Algebra                    3.00            202401
BAMS2014        Linear Algebra                          3.00            202401

Enter Course Code:
```

Figure 2.2 The list of courses is displayed

If user input is '1', the list of courses will be displayed to the user and prompts the user for input.

```
Please indicate your selection to proceed(1-8): 1

Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms           4.00            202401
BAIT1023        Web Design And Development              2.00            202401
BAIT1043        System Analysis And Design              3.00            202401
BABS1233        Microbiology                            2.00            202401
BACH2233        Food Chemistry And Analysis             4.00            202401
BACH1613        Physical Chemistry                      3.00            202401
BTME4263        Heat Transfer                           3.00            202401
BGEE2614        Electrical Power Systems                2.00            202401
BTEE1513        Electrical Technology                   4.00            202401
BTEE4033        Integrated Circuit Technology           3.00            202401
BJEL1023        Academic English                        3.00            202401
BJEL1013        English For Tertiary Studies            3.00            202401
BAMS1413        Calculus And Algebra                    3.00            202401
BAMS2014        Linear Algebra                          3.00            202401

Enter Course Code: BACS1053
Invalid course!
Enter Course Code:
```

Figure 2.3 Invalid Course Code Error Message

If the user inputs the course code which is not on the list, an error message will be displayed.

```
Enter Course Code: BAIT1023

Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code:
```

Figure 2.4 The list of programmes is displayed

After the user inputs the course code, the list of programmes will be displayed and prompts the user for input.

```
Enter Course Code: BAIT1023

Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RBX
Invalid programme!
Enter Programme Code:
```

Figure 2.5 Invalid Programme Code Error Message

If the user inputs the programme code which is not on the list, an error message will be displayed.

```
Enter Course Code: BAIT1023

Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RSD


Record already exist!

Press enter to continue...
```

Figure 2.6 Failed to Add Programme to Course

If the programme has already taken the course, an error message ‘Record already exist!’ will be displayed.

```
Enter Course Code: BAIT1023

Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RSW


New programme has been added to course successfully!

Press enter to continue...
```

Figure 2.7 Successfully Added Programme to Course

```
Please indicate your selection to proceed(1-8): 2
Programme Details:

-------------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-------------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RSW


Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                              Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms            4.00            202401
BAIT1043        System Analysis And Design               3.00            202401
BAMS1413        Calculus And Algebra                     3.00            202401
BAIT1023        Web Design And Development               2.00            202401

Enter Course Code:
```

Figure 2.8 Remove Programme From Course

After the user input the programme code, the courses taken by the programme will be shown.

```
Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                              Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms            4.00            202401
BAIT1043        System Analysis And Design               3.00            202401
BAMS1413        Calculus And Algebra                     3.00            202401
BAIT1023        Web Design And Development               2.00            202401

Enter Course Code: BACS1053
Invalid course!
Enter Course Code:
```

Figure 2.9 Invalid Course Code Error Message

If the user inputs the course that is not taken by the programme, an error message will be shown.

```
Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                                 Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms               4.00            202401
BAIT1043        System Analysis And Design                  3.00            202401
BAMS1413        Calculus And Algebra                        3.00            202401
BAIT1023        Web Design And Development                  2.00            202401

Enter Course Code: BAIT1023

Continue removing programme from course(Y/N): Y
RSW had been removed from BAIT1023 successfully!

Press enter to continue...
```

Figure 2.10 Confirmation

```
Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                                 Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms               4.00            202401
BAIT1043        System Analysis And Design                  3.00            202401
BAMS1413        Calculus And Algebra                        3.00            202401
BAIT1023        Web Design And Development                  2.00            202401

Enter Course Code: BAIT1023

Continue removing programme from course(Y/N): N
Programme remove cancelled!

Press enter to continue...
```

Figure 2.11 Confirmation

For Figure 2.10 and Figure 2.11, if the user inputs ‘Y’, then the remove process will be successful, else, vice versa.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 3
Enter Course Code: BACS1053
Enter Course Name: Database Management
Enter Credit Hour: 3.0
Enter Academic Year: 202401

New course had been added successfully!

Press enter to continue adding the new course to a programme...
```

Figure 2.12 Add New Course to Programme

In this module, the system will prompt users to input the information of the new course code.

```
Please indicate your selection to proceed(1-8): 3
Enter Course Code: BACS1053
Enter Course Name: Database Management
Enter Credit Hour: 3.0
Enter Academic Year: 202401

New course had been added successfully!

Press enter to continue adding the new course to a programme...

Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code:
```

Figure 2.13 Successfully Added a New Course

```
Please indicate your selection to proceed(1-8): 3
Enter Course Code: BACS1053
Enter Course Name: Database Management
Enter Credit Hour: 3.0
Enter Academic Year: 202401

New course had been added successfully!

Press enter to continue adding the new course to a programme...

Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RSW


New programme has been added to course successfully!

Press enter to continue...
```

Figure 2.14 Successfully Added a New Course to Programme

For Figure 2.13 and Figure 2.14, after the user successfully adds a new course, the system will prompt the users to assign the new course to a programme.

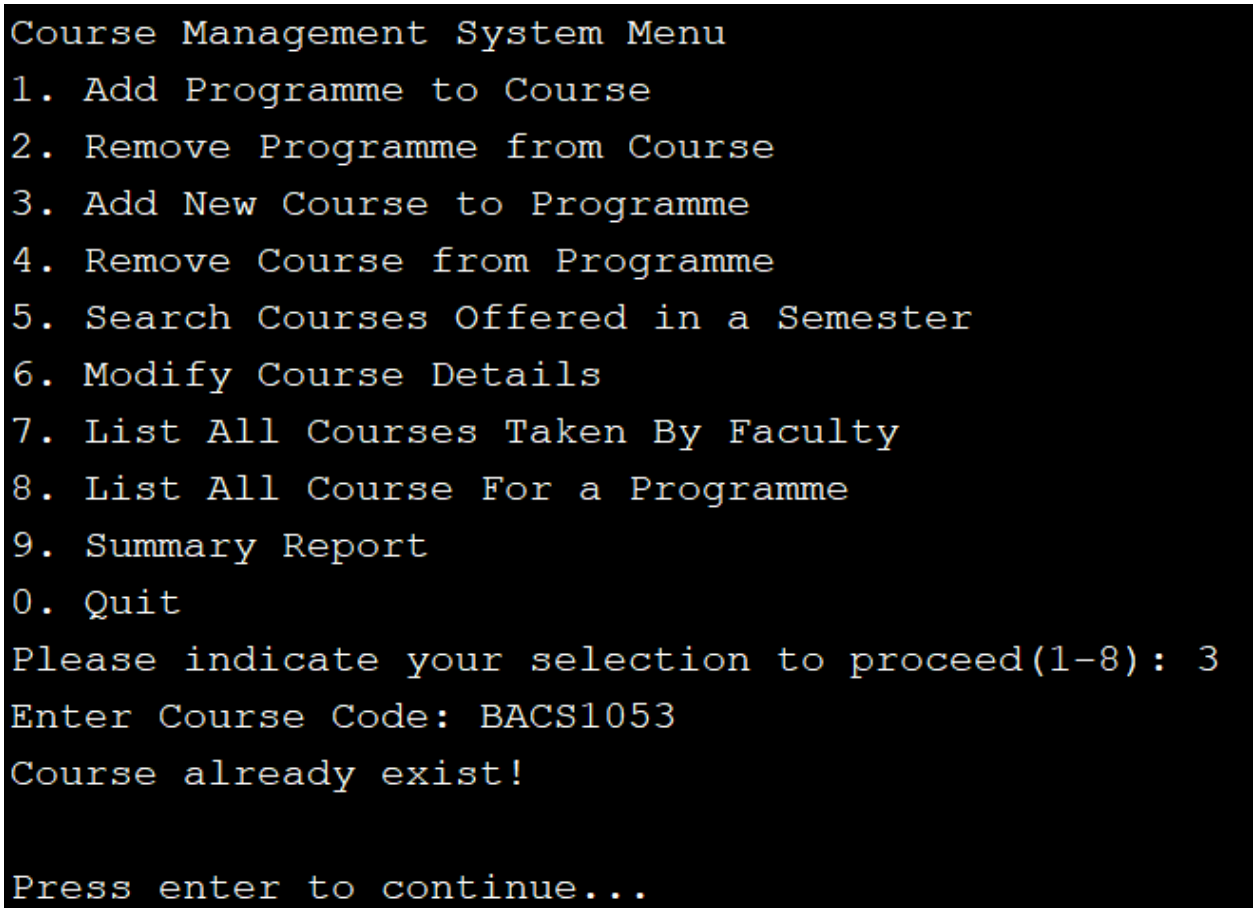


Figure 2.15 Invalid Course Error Message

If the user inputs the course that already exists in the system, an error message will be shown.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 4
Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code:
```

Figure 2.16 Remove Course From Programme Module

```
Enter Programme Code: RSW


Course Details:

-----------------------------------------------------------------------------------------------
Course Code     Course Name                              Credit Hour    Semester
-----------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms            4.00            202401
BAIT1043        System Analysis And Design               3.00            202401
BAMS1413        Calculus And Algebra                     3.00            202401
BACS1053        Database Management                      3.00            202401

Enter Course Code: BACS1053

Continue removing programme from course(Y/N): Y
RSW had been removed from BACS1053 successfully!

Press enter to continue...
```

Figure 2.17 Successfully Remove Course From Programme

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 5
Enter Academic Year: 202401

Course Details:

-------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401
BACS1053        Database Management                     3.00           202401


Press enter to continue...
```

Figure 2.18 Search Courses By Semester Module

In this module, the system will prompt the user to input the semester and display the courses by the semester.

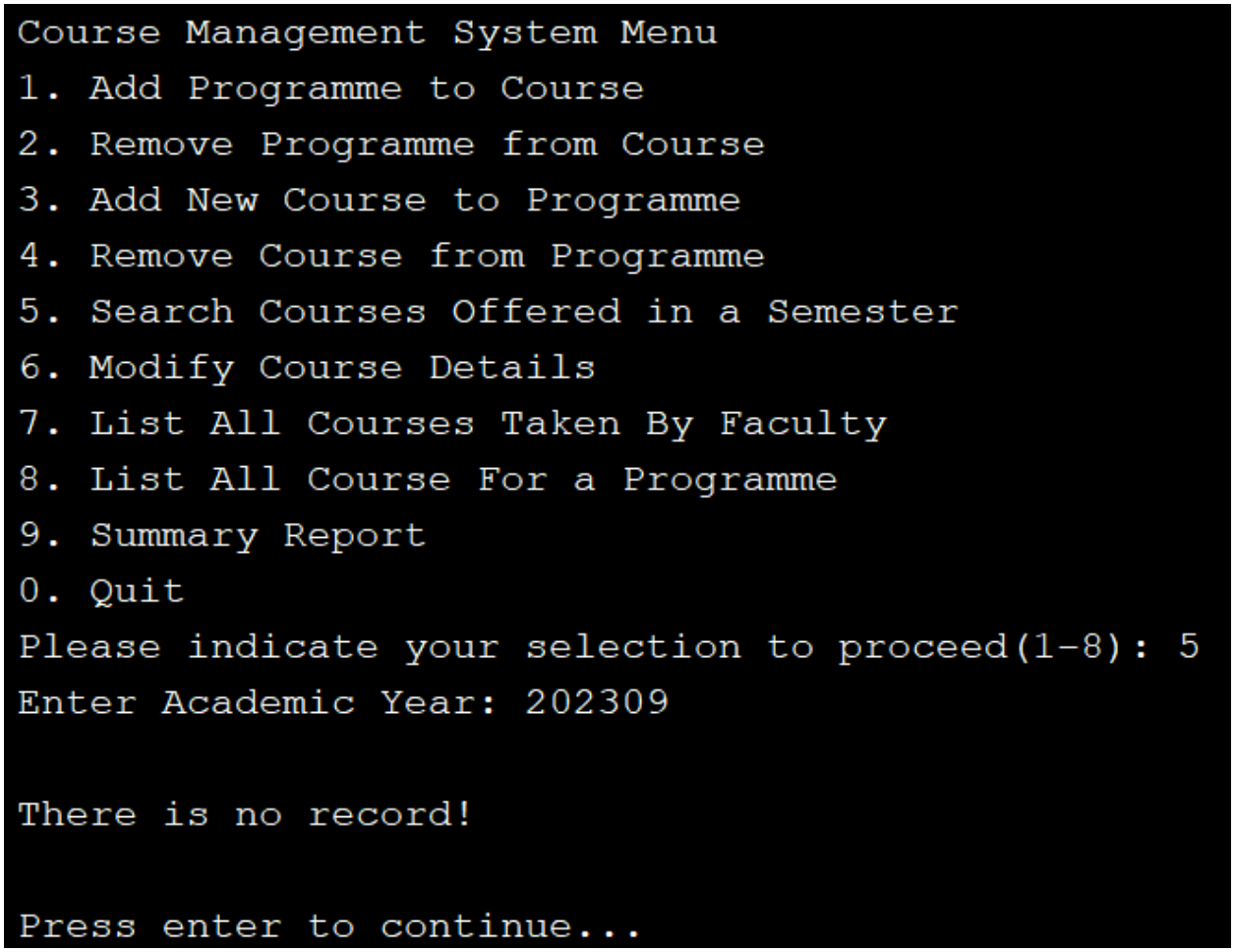


Figure 2.19 Invalid Semester

If there is an empty course offered in that semester, a message ‘There is no record!’ will be displayed.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 6

Enter Course Code:
```

Figure 2.20 Amend Course Details Module

For this module, the system will prompt the user to input the course code that they wanted to modify.

```
Enter Course Code: BACS1053

Edit Course Details Menu
1. Course Name
2. Credit Hour
3. Academic Year
0. Back
Please select the details you would like to modify(1-3):
```

Figure 2.21 Modify Course Details Menu

After the user input the valid course code, the system will show the course modify menu for the user to choose.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 6

Enter Course Code: BACS1055
Course with course code BACS1055 is not found in the list!


Press enter to continue...
```

Figure 2.22 Invalid Course Code

If the user inputs the course that does not exist in the system, an error message will be shown.

```
Enter Course Code: BACS1053

Edit Course Details Menu
1. Course Name
2. Credit Hour
3. Academic Year
0. Back
Please select the details you would like to modify(1-3): 1

Old Course Name for BACS1053: Database Management
Enter Course Name: Advanced Database Management
Database Management had been changed to Advanced Database Management
```

Figure 2.23 Example of Changing the Course Name

```
Edit Course Details Menu
1. Course Name
2. Credit Hour
3. Academic Year
0. Back
Please select the details you would like to modify(1-3): 2

Old Credit Hour for BACS1053: 3.0
Enter Credit Hour: 4.0
3.0 credit hour for BACS1053 had been changed to 4.0
```

Figure 2.24 Example of Changing the Credit Hour

```
Edit Course Details Menu
1. Course Name
2. Credit Hour
3. Academic Year
0. Back
Please select the details you would like to modify(1-3): 3

Old Academic Year for BACS1053: 202401
Enter Academic Year: 202309
202401 academic year for BACS1053 had been changed to 202309
```

Figure 2.25 Example of Changing the Semester

```
Enter Academic Year: 202309

Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS1053        Advanced Database Management            4.00           202309


Press enter to continue...
```

Figure 2.26 The Course Information after Modification

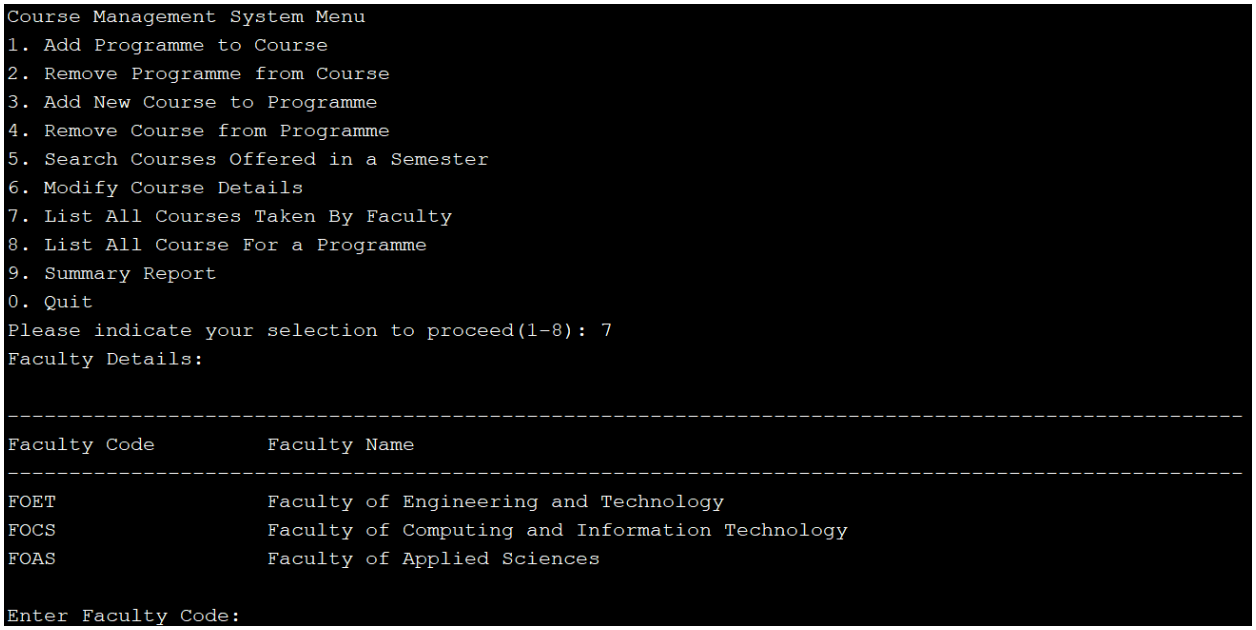


Figure 2.27 List Course Taken By Different Faculties Module

The system will display the information of faculties for the user as a reference.

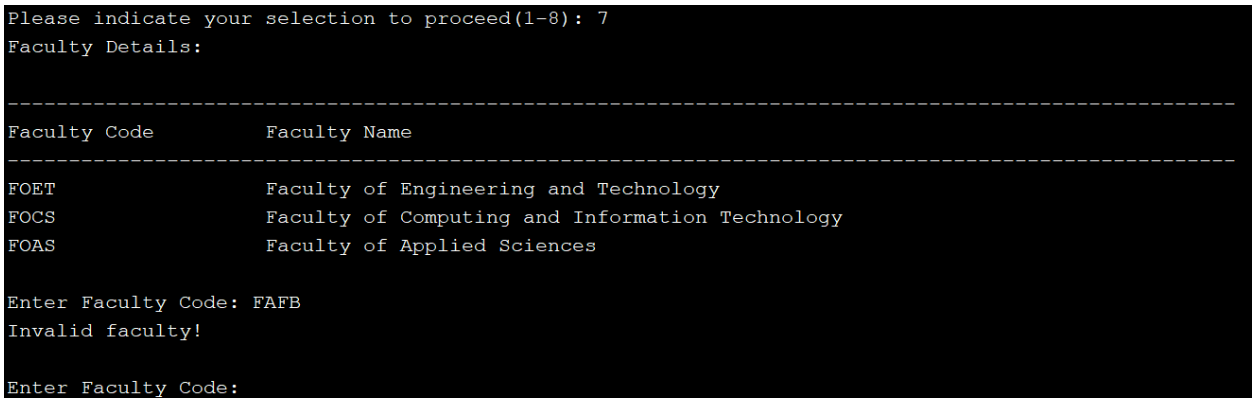


Figure 2.28 Invalid Faculty Code

If the user inputs the faculty code that is not in the list, an error message will be displayed.

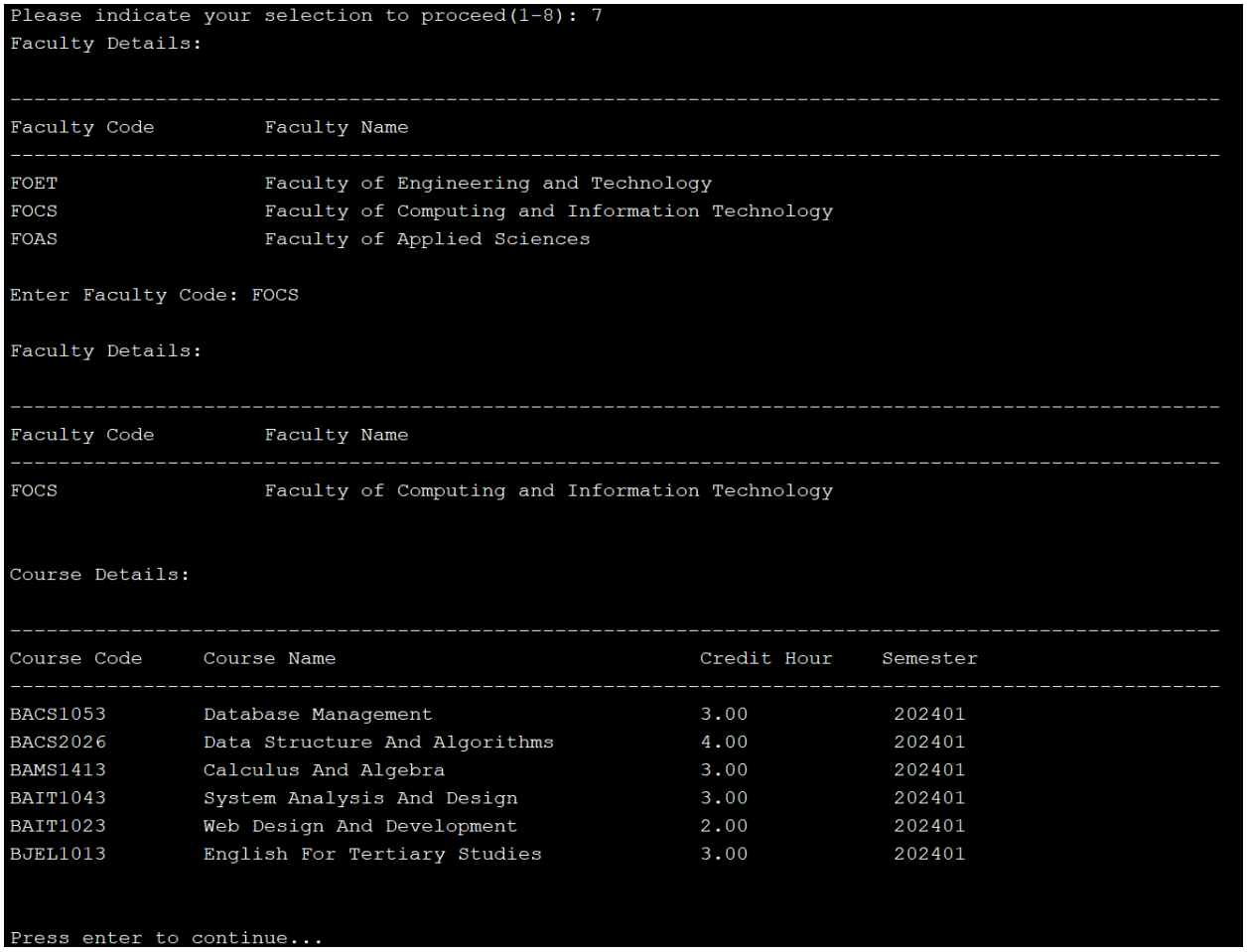


Figure 2.29 Course Displayed

After the user input the valid faculty code, all the courses taken by the faculty will be displayed.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 8
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code:
```

Figure 2.30 List Courses For a Programme Module

In this module, the list of programmes will be shown for the user as a reference.

```
Please indicate your selection to proceed(1-8): 8
Programme Details:

-------------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-------------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RBX
Invalid programme!
Enter Programme Code:
```

Figure 2.31 Invalid Programme Code

If the user inputs the programme code that is not in the list, an error message will be displayed.

```
Please indicate your selection to proceed(1-8): 8
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: RSW


Course Details:

-----------------------------------------------------------------------------------------------------
Course Code     Course Name                              Credit Hour    Semester
-----------------------------------------------------------------------------------------------------
BACS2026        Data Structure And Algorithms            4.00            202401
BAIT1043        System Analysis And Design               3.00            202401
BAMS1413        Calculus And Algebra                     3.00            202401


Press enter to continue...
```

Figure 2.32 Course Displayed

After the user input the valid programme code, all the courses taken by the programme will be displayed.

```
Course Management System Menu
1. Add Programme to Course
2. Remove Programme from Course
3. Add New Course to Programme
4. Remove Course from Programme
5. Search Courses Offered in a Semester
6. Modify Course Details
7. List All Courses Taken By Faculty
8. List All Course For a Programme
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-8): 9

Summary Report Menu
1. Faculty Summary Report
2. Programme Summary Report
3. Course Summary Report
0. Back
Please select the report you would like to display(1-3):
```

Figure 2.33 Summary Report Menu

The user can choose to view the faculty summary report, programme summary report or course summary report.

```
Summary Report Menu
1. Faculty Summary Report
2. Programme Summary Report
3. Course Summary Report
0. Back
Please select the report you would like to display(1-3): 1


Sorting Menu
1. Faculty Code
2. Faculty Name
Please choose the criteria by which you'd like to sort the details in the faculty list(1-2):1


1. Ascending Order
2. Desencing Order
Please enter your choice of sort order(1-2):
```

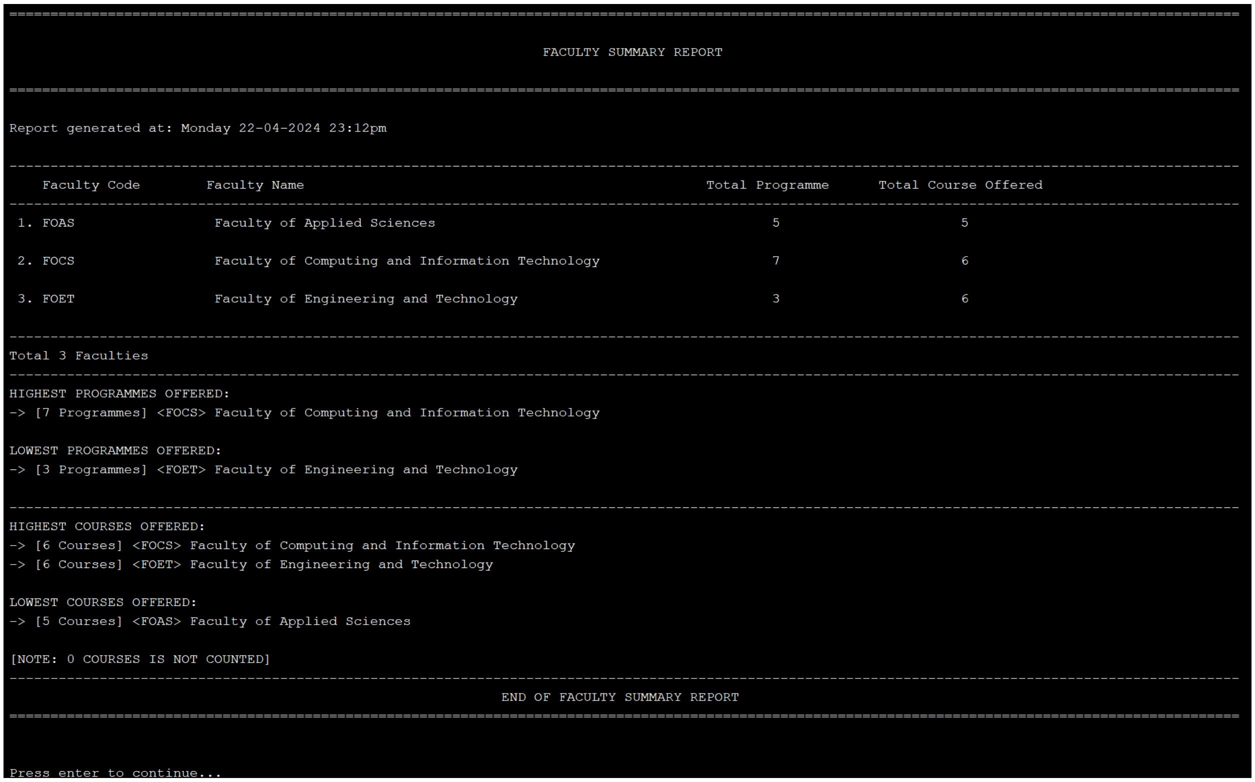
Figure 2.34 Sorting Menu For Faculty Summary Report

```
Summary Report Menu
1. Faculty Summary Report
2. Programme Summary Report
3. Course Summary Report
0. Back
Please select the report you would like to display(1-3): 2


Sorting Menu
1. Programme Code
2. Programme Name
Please choose the criteria by which you'd like to sort the details in the faculty list(1-2):1


1. Ascending Order
2. Desencing Order
Please enter your choice of sort order(1-2):
```

Figure 2.35 Sorting Menu For Programme Summary Report

```
Summary Report Menu
1. Faculty Summary Report
2. Programme Summary Report
3. Course Summary Report
0. Back
Please select the report you would like to display(1-3): 3


Sorting Menu
1. Course Code
2. Course Name
Please choose the criteria by which you'd like to sort the details in the faculty list(1-2):1


1. Ascending Order
2. Desencing Order
Please enter your choice of sort order(1-2):
```

Figure 2.36 Sorting Menu For Course Summary Report



```
==========================================================================================================================
                                                  FACULTY SUMMARY REPORT
==========================================================================================================================

Report generated at: Monday 22-04-2024 23:12pm

--------------------------------------------------------------------------------------------------------------------------
    Faculty Code        Faculty Name                                              Total Programme      Total Course Offered
--------------------------------------------------------------------------------------------------------------------------
 1. FOAS                 Faculty of Applied Sciences                                     5                      5

 2. FOCS                 Faculty of Computing and Information Technology                 7                      6

 3. FOET                 Faculty of Engineering and Technology                           3                      6

--------------------------------------------------------------------------------------------------------------------------
Total 3 Faculties
--------------------------------------------------------------------------------------------------------------------------
HIGHEST PROGRAMMES OFFERED:
-> [7 Programmes] <FOCS> Faculty of Computing and Information Technology

LOWEST PROGRAMMES OFFERED:
-> [3 Programmes] <FOET> Faculty of Engineering and Technology

--------------------------------------------------------------------------------------------------------------------------
HIGHEST COURSES OFFERED:
-> [6 Courses] <FOCS> Faculty of Computing and Information Technology
-> [6 Courses] <FOET> Faculty of Engineering and Technology

LOWEST COURSES OFFERED:
-> [5 Courses] <FOAS> Faculty of Applied Sciences

[NOTE: 0 COURSES IS NOT COUNTED]
--------------------------------------------------------------------------------------------------------------------------
                                             END OF FACULTY SUMMARY REPORT
==========================================================================================================================


Press enter to continue...
```

Figure 2.37 Example of Faculty Summary Report Sorted By Faculty Code In Ascending Order

```
                                                                PROGRAMME SUMMARY REPORT

Report generated at: Monday 22-04-2024 23:21pm

    Programme Code      Programme Name                                                                  Courses Offered      Faculty Offered
 1. RAN                 Bachelor Of Science (Hons) In Analytical Chemistry                                    2                   1

 2. RBS                 Bachelor Of Science (Hons) In Bioscience With Chemistry                               2                   1

 3. RDS                 Bachelor Of Computer Science (HONOURS) In Data Science                                3                   1

 4. REE                 Bachelor Of Electrical And Electronics Engineering With Honours                       4                   1

 5. REI                 Bachelor Of Information Systems (Honours) In Enterprise Information Systems           3                   1

 6. RFN                 Bachelor Of Science (Hons) In Food Science                                            2                   1

 7. RIS                 Bachelor Of Information Technology (Honours) In Information Security                  1                   1

 8. RME                 Bachelor Of Mechanical Engineering With Honours                                       3                   1

 9. RMM                 Bachelor Of Science (Honours) In Management Mathematics With Computing               2                   1

10. RMT                 Bachelor Of Engineering (Honours) Material                                            3                   1

11. RNR                 Bachelor Of Science (Hons) In Nutrition                                               1                   1

12. RSD                 Bachelor Of Information Technology (HONOURS) In Software Systems Development         3                   1

13. RSE                 Bachelor Of Science (Hons) In Sports And Exercise Science                             1                   1

14. RST                 Bachelor Of Computer Science (Honours) In Interactive Software Technology             1                   1

15. RSW                 Bachelor Of Software Engineering (HONOURS)                                            3                   1

Total 15 Programmes

HIGHEST COURSES OFFERED:
-> [4 Courses] <REE> Bachelor Of Electrical And Electronics Engineering With Honours

LOWEST COURSES OFFERED:
-> [1 Courses] <RIS> Bachelor Of Information Technology (Honours) In Information Security
-> [1 Courses] <RNR> Bachelor Of Science (Hons) In Nutrition
-> [1 Courses] <RSE> Bachelor Of Science (Hons) In Sports And Exercise Science
-> [1 Courses] <RST> Bachelor Of Computer Science (Honours) In Interactive Software Technology

[NOTE: 0 COURSES IS NOT COUNTED]

HIGHEST FACULTIES OFFERED:
-> [1 Faculties] <RAN> Bachelor Of Science (Hons) In Analytical Chemistry
-> [1 Faculties] <RBS> Bachelor Of Science (Hons) In Bioscience With Chemistry
-> [1 Faculties] <RDS> Bachelor Of Computer Science (HONOURS) In Data Science
-> [1 Faculties] <REE> Bachelor Of Electrical And Electronics Engineering With Honours
-> [1 Faculties] <REI> Bachelor Of Information Systems (Honours) In Enterprise Information Systems
-> [1 Faculties] <RFN> Bachelor Of Science (Hons) In Food Science
-> [1 Faculties] <RIS> Bachelor Of Information Technology (Honours) In Information Security
-> [1 Faculties] <RME> Bachelor Of Mechanical Engineering With Honours
-> [1 Faculties] <RMM> Bachelor Of Science (Honours) In Management Mathematics With Computing
-> [1 Faculties] <RMT> Bachelor Of Engineering (Honours) Material
-> [1 Faculties] <RNR> Bachelor Of Science (Hons) In Nutrition
-> [1 Faculties] <RSD> Bachelor Of Information Technology (HONOURS) In Software Systems Development
-> [1 Faculties] <RSE> Bachelor Of Science (Hons) In Sports And Exercise Science
-> [1 Faculties] <RST> Bachelor Of Computer Science (Honours) In Interactive Software Technology
-> [1 Faculties] <RSW> Bachelor Of Software Engineering (HONOURS)

LOWEST FACULTIES OFFERED:
-> [1 Faculties] <RAN> Bachelor Of Science (Hons) In Analytical Chemistry
-> [1 Faculties] <RBS> Bachelor Of Science (Hons) In Bioscience With Chemistry
-> [1 Faculties] <RDS> Bachelor Of Computer Science (HONOURS) In Data Science
-> [1 Faculties] <REE> Bachelor Of Electrical And Electronics Engineering With Honours
-> [1 Faculties] <REI> Bachelor Of Information Systems (Honours) In Enterprise Information Systems
-> [1 Faculties] <RFN> Bachelor Of Science (Hons) In Food Science
-> [1 Faculties] <RIS> Bachelor Of Information Technology (Honours) In Information Security
-> [1 Faculties] <RME> Bachelor Of Mechanical Engineering With Honours
-> [1 Faculties] <RMM> Bachelor Of Science (Honours) In Management Mathematics With Computing
-> [1 Faculties] <RMT> Bachelor Of Engineering (Honours) Material
-> [1 Faculties] <RNR> Bachelor Of Science (Hons) In Nutrition
-> [1 Faculties] <RSD> Bachelor Of Information Technology (HONOURS) In Software Systems Development
-> [1 Faculties] <RSE> Bachelor Of Science (Hons) In Sports And Exercise Science
-> [1 Faculties] <RST> Bachelor Of Computer Science (Honours) In Interactive Software Technology
-> [1 Faculties] <RSW> Bachelor Of Software Engineering (HONOURS)

[NOTE: 0 FACULTIES IS NOT COUNTED]

                                                            END OF PROGRAMME SUMMARY REPORT


Press enter to continue...
```

Figure 2.38 Example of Programme Summary Report Sorted By Programme Code In Ascending Order

```
================================================================================================================================================

                                                    COURSE SUMMARY REPORT

================================================================================================================================================

Report generated at: Monday 22-04-2024 23:21pm

------------------------------------------------------------------------------------------------------------------------------------------------
    Course Code     Course Name                         Credit Hour       Semester       Programmes Offered   Faculty Offered
 1. BABS1233        Microbiology                            2.00           202401                1                1

 2. BACH1613        Physical Chemistry                      3.00           202401                1                1

 3. BACH2233        Food Chemistry And Analysis             4.00           202401                1                1

 4. BACS1053        Database Management                     3.00           202401                3                1

 5. BACS2026        Data Structure And Algorithms           4.00           202401                2                1

 6. BAIT1023        Web Design And Development              2.00           202401                2                1

 7. BAIT1043        System Analysis And Design              3.00           202401                2                1

 8. BAMS1413        Calculus And Algebra                    3.00           202401                6                2

 9. BAMS2014        Linear Algebra                          3.00           202401                3                1

10. BGEE2614        Electrical Power Systems                2.00           202401                1                1

11. BJEL1013        English For Tertiary Studies            3.00           202401                7                2

12. BJEL1023        Academic English                        3.00           202401                2                1

13. BTEE1513        Electrical Technology                   4.00           202401                1                1

14. BTEE4033        Integrated Circuit Technology           3.00           202401                1                1

15. BTME4263        Heat Transfer                           3.00           202401                1                1

------------------------------------------------------------------------------------------------------------------------------------------------
Total 15 Courses
------------------------------------------------------------------------------------------------------------------------------------------------
HIGHEST PROGRAMMES OFFERED:
-> [7 Programmes] <BJEL1013> English For Tertiary Studies

LOWEST PROGRAMMES OFFERED:
-> [1 Programmes] <BABS1233> Microbiology
-> [1 Programmes] <BACH1613> Physical Chemistry
-> [1 Programmes] <BACH2233> Food Chemistry And Analysis
-> [1 Programmes] <BGEE2614> Electrical Power Systems
-> [1 Programmes] <BTEE1513> Electrical Technology
-> [1 Programmes] <BTEE4033> Integrated Circuit Technology
-> [1 Programmes] <BTME4263> Heat Transfer

[NOTE: 0 PROGRAMMES IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
HIGHEST FACULTIES OFFERED:
-> [2 Faculties] <BAMS1413> Calculus And Algebra
-> [2 Faculties] <BJEL1013> English For Tertiary Studies

LOWEST FACULTIES OFFERED:
-> [1 Faculties] <BABS1233> Microbiology
-> [1 Faculties] <BACH1613> Physical Chemistry
-> [1 Faculties] <BACH2233> Food Chemistry And Analysis
-> [1 Faculties] <BACS1053> Database Management
-> [1 Faculties] <BACS2026> Data Structure And Algorithms
-> [1 Faculties] <BAIT1023> Web Design And Development
-> [1 Faculties] <BAIT1043> System Analysis And Design
-> [1 Faculties] <BAMS2014> Linear Algebra
-> [1 Faculties] <BGEE2614> Electrical Power Systems
-> [1 Faculties] <BJEL1023> Academic English
-> [1 Faculties] <BTEE1513> Electrical Technology
-> [1 Faculties] <BTEE4033> Integrated Circuit Technology
-> [1 Faculties] <BTME4263> Heat Transfer

[NOTE: 0 FACULTIES IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------
                                                  END OF COURSE SUMMARY REPORT
================================================================================================================================================


Press enter to continue...
```

Figure 2.39 Example of Course Summary Report Sorted By Course Code In Ascending Order

Name	Student ID	Prog / Tut.Grp	Signature
Goh Boon Xiang	2214263	RDS2S2G3	*goh*

Subsystem : Tutorial Group Management Subsystem

1. **Source codes for Control classes**

```
package control;

import adt.*;
import boundary.CourseManagementUI;
import boundary.TutorialGroupManagementUI;
import dao.*;
import entity.Programme;
import entity.Student;
import entity.TutorialGroup;
import java.util.Scanner;
import utility.MessageUI;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
/**
 *
 * @author Goh Boon Xiang
 */
public class TutorialGroupManagement {

    Scanner scanner = new Scanner(System.in);
    private ListInterface<Student> studentList = new ArrayList<>();
    private ListInterface<Programme> ProgrammeList = new ArrayList<>();
    private ListInterface<TutorialGroup> TutorialGroupList = new ArrayList<>();
    private StudentDAO StudentDAO = new StudentDAO();
    private StudentInitializer StudentInitializer = new StudentInitializer();
    private ProgrammeDAO ProgrammeDAO = new ProgrammeDAO();
    private ProgrammeInitializer ProgrammeInitializer = new ProgrammeInitializer();
    private TutorialGroupDAO TutorialGroupDAO =  new TutorialGroupDAO();
        private TutorialGroupInitializer TutorialGroupInitializer= new
TutorialGroupInitializer();
    private CourseManagementUI CourseManagementUI = new CourseManagementUI();
            private TutorialGroupManagementUI GroupManagementUI = new
TutorialGroupManagementUI();


    public TutorialGroupManagement(){
```

```
        studentList = StudentDAO.retrieveFromFile();
        ProgrammeList = ProgrammeDAO.retrieveFromFile();
        TutorialGroupList = TutorialGroupDAO.retrieveFromFile();

    }

    public static void main(String[] args) {
                            TutorialGroupManagement tutorialGroupManagement = new
TutorialGroupManagement();
        tutorialGroupManagement.runTutorialGroupManagement();
    }

    public void runTutorialGroupManagement(){
        int choice = 0;
        do {
            choice = GroupManagementUI.getMenuChoice();
            switch (choice) {
                case 0:
                    break;
                case 1:
                    addGroupToProgramme(null,'N');
                    break;
                case 2:
                    removeGroupFromProgramme();
                    break;
                case 3:
                    listAllTutorialGroup();
                    break;
                case 4:
                    addNewStudentToGroup();
                    break;
                case 5:
                    removeStudentFromGroup();
                    break;
                case 6:
                    studentChangeGroup();
                    break;
                case 7:
                    listGroupStudent();
                    break;
                case 8:
                    mergeGroup();
                    break;
                case 9:
                    report();
                    break;
```

```
            case 10:
                TutorialGroupReport();
                break;

            default:
                MessageUI.displayInvalidChoiceMessage();
        }
    } while (choice != 0);
}


//Add new tutorial group in programme
   public void addGroupToProgramme(String programmeCode,char check){

    displayProgramme(null);
        System.out.println("Please enter programme code that you want to add tutorial group");

    if (programmeCode == null || !findProgramme(programmeCode)) {
        do {
                                                                programmeCode = CourseManagementUI.inputProgrammeCode().toUpperCase();
            if (!findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
    }


    System.out.println();

        String newGroupID = generateUniqueGroupID(programmeCode); // Generate a unique group ID
                    String newGroupName = programmeCode + " Group " + getGroupNumber(newGroupID); // Generate the group name

    TutorialGroup newTutorialGroup = new TutorialGroup();
    newTutorialGroup.setProgrammeCode(programmeCode);
    newTutorialGroup.setGroupID(newGroupID);
    newTutorialGroup.setGroupName(newGroupName);

    TutorialGroupList.add(newTutorialGroup);
    TutorialGroupDAO.saveToFile(TutorialGroupList);
```

```
        System.out.println("New tutorial group added successfully:");
        System.out.println(newTutorialGroup);

    }

    //Remove tutorial group from programme
    public void removeGroupFromProgramme() {
        char check = 'N';
        String programmeCode;
        String groupID = "";

        do {
            displayProgramme(null);
                                                                    programmeCode =
CourseManagementUI.inputProgrammeCode().toUpperCase();
            if (!findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        displayProgrammeGroup(programmeCode);
        System.out.print("Continue removing tutorial group from programme (Y/N): ");
        check = scanner.next().charAt(0);
        scanner.nextLine();

        while (check == 'Y') {
            groupID = GroupManagementUI.inputGroupID();
            if (findGroup(groupID)) {
                check = 'N';
            } else {
                System.out.println("Invalid group ID!");

            }
            System.out.println();
        }

        int indexToRemove = -1;
        for (int i = 0; i < TutorialGroupList.getNumberOfEntries() +1; i++) {
            if (TutorialGroupList.getEntry(i) != null) {
                if (TutorialGroupList.getEntry(i).getProgrammeCode().equals(programmeCode)
                        && TutorialGroupList.getEntry(i).getGroupID().equals(groupID)) {
                    indexToRemove = i;
```

```
                break;
            }
        }
    }

    if (indexToRemove != -1) {
        TutorialGroupList.remove(indexToRemove);
        TutorialGroupDAO.saveToFile(TutorialGroupList);
          System.out.println(groupID + " has been removed from " + programmeCode + "
successfully!");
    } else {
        System.out.println("Failed to remove " + groupID + " from " + programmeCode +
" !");
    }

    System.out.println("\nPress enter to continue...");
    scanner.nextLine();


}

// list down all tutorial group of the programme
public void listAllTutorialGroup(){

    String programmeCode = null;
    char check = 'N';

    displayProgramme(null);
    System.out.print("Please enter programme code that you want to see");

    if (programmeCode == null || !findProgramme(programmeCode)) {
        do {
                                                        programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
            if (!findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
    }

    displayProgrammeGroup(programmeCode);

}
```

```
    public void addNewStudentToGroup() {
        String programmeCode = null;
        char check = 'N';

        displayProgramme(null);
        System.out.print("Please enter programme code to add students to a group");

        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        }

        displayProgrammeGroup(programmeCode);
        System.out.println("");

        System.out.print("Enter group ID to add students:");
        String groupID = scanner.nextLine();

        // Find the group by its ID
        int groupIndex = findGroupIndex(groupID);
        if (groupIndex == -1) {
            System.out.println("Group not found!");
            return;
        }

        // Get the group and its students list
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        ListInterface<Student> groupStudentsList = group.getGroupStudentsList();

        // Ask the user to input student IDs to add to the group
        System.out.println("Enter student IDs to add to the group (separated by comma):");
        String inputStudentIDs = scanner.nextLine();
        String[] studentIDs = inputStudentIDs.split(",");

        // Add each student to the group if they exist and their programme code matches
        for (String studentID : studentIDs) {
            Student student = findStudent(studentID);
```

```
        if (student != null && student.getProgrammeCode().equals(programmeCode)) {
            boolean isStudentAlreadyInOtherGroup = false;
            // Iterate through all tutorial groups except the current group
            for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
                TutorialGroup otherGroup = TutorialGroupList.getEntry(i);
                                        if (!otherGroup.getGroupID().equals(groupID) &&
otherGroup.getGroupStudentsList().contains(student)) {
                    isStudentAlreadyInOtherGroup = true;
                    break;
                }
            }
            if (!isStudentAlreadyInOtherGroup) {
                if (!groupStudentsList.contains(student)) { // Check if student is not already
in the group then add
                    groupStudentsList.add(student);
                    // Save the updated tutorial group list to file
                    TutorialGroupDAO.saveToFile(TutorialGroupList);
                                System.out.println("Student " + studentID + " added to group
successfully!");
                } else {
                    System.out.println("Student " + studentID + " is already in the group!");
                }
            } else {
                System.out.println("Student " + studentID + " is already in another tutorial
group!");
            }
        } else {
            System.out.println("Student " + studentID + " not found or not enrolled in the
programme!");
        }
    }


    // Ask if the user wants to view detailed information about the group
        System.out.println("Do you want to view detailed information about this group?
(Y/N)");
    char choice = scanner.next().charAt(0);
    scanner.nextLine(); // Consume newline character

    if (Character.toUpperCase(choice) == 'Y') {
        displayGroupDetail(group);
    }

}
```

```
    public void removeStudentFromGroup(){

        String programmeCode = null;
        String groupID;
        char check = 'N';

        // Display available programmes and prompt user to input programme code
        displayProgramme(null);
        System.out.println("Please enter the programme code:");

        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        }

        // Display tutorial groups for the specified programme code and prompt user to input
group ID
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the group ID to remove a student:");
        groupID = scanner.nextLine();

        // Find the group by its ID
        int groupIndex = findGroupIndex(groupID);
        if (groupIndex == -1) {
            System.out.println("Group not found!");
            return;
        }

        // Get the group and its students list
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        ListInterface<Student> groupStudentsList = group.getGroupStudentsList();

        // Display group details and prompt user to input student ID to remove
        displayGroupDetail(group);
        System.out.println("Enter the student ID to remove from the group:");
        String studentIDToRemove = scanner.nextLine();

        // Check if the student exists in the group
```

```
        boolean studentExistsInGroup = false;
        for (int i = 1; i <= groupStudentsList.getNumberOfEntries(); i++) {
            Student student = groupStudentsList.getEntry(i);
            if (student.getStudentID().equals(studentIDToRemove)) {
                groupStudentsList.remove(i); // Remove the student from the group
                    TutorialGroupDAO.saveToFile(TutorialGroupList); // Save the updated group
list to file
                System.out.println("Student removed from the group successfully!");
                studentExistsInGroup = true;
                break;
            }
        }

        if (!studentExistsInGroup) {
            System.out.println("Student not found in the group or already removed!");
        }


    }

    public void studentChangeGroup(){
        String programmeCode = null;
        String currentGroupID;
        char check = 'N';

        // Display available programmes and prompt user to input programme code
        displayProgramme(null);
        System.out.println("Please enter the programme code:");
        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                    programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        }

        // Display tutorial groups for the specified programme code and prompt user to input
group ID
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the current group ID of the student:");
        currentGroupID = scanner.nextLine();
```

```
        // Find the current group by its ID
        int currentGroupIndex = findGroupIndex(currentGroupID);
        if (currentGroupIndex == -1) {
            System.out.println("Current group not found!");
            return;
        }

        // Get the current group and its students list
        TutorialGroup currentGroup = TutorialGroupList.getEntry(currentGroupIndex);
                                    ListInterface<Student>      currentGroupStudentsList      =
currentGroup.getGroupStudentsList();

        // Display group details and prompt user to input student ID to change group
        displayGroupDetail(currentGroup);
        System.out.println("Enter the student ID to change group:");
        String studentIDToChange = scanner.nextLine();

        // Find the student by ID in the current group
        Student studentToChange = null;
        for (int i = 1; i <= currentGroupStudentsList.getNumberOfEntries(); i++) {
            Student student = currentGroupStudentsList.getEntry(i);
            if (student.getStudentID().equals(studentIDToChange)) {
                studentToChange = student;
                break;
            }
        }

        if (studentToChange == null) {
            System.out.println("Student not found in the current group!");
            return;
        }

        // Display available groups for the specified programme code
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the new group ID to move the student:");
        String newGroupID = scanner.nextLine();

        // Validate new group ID
        int newGroupIndex = findGroupIndex(newGroupID);
        if (newGroupIndex == -1 || newGroupID.equals(currentGroupID)) {
            System.out.println("Invalid new group ID!");
            return;
        }

        // Check if the new group belongs to the same programme code
```

```
        TutorialGroup newGroup = TutorialGroupList.getEntry(newGroupIndex);
        if (!newGroup.getProgrammeCode().equals(programmeCode)) {
            System.out.println("New group does not belong to the same programme!");
            return;
        }

        // Add the student to the new group and remove from the current group
        newGroup.getGroupStudentsList().add(studentToChange);

        int studentIndexToRemove = -1;
        for (int i = 1; i <= currentGroupStudentsList.getNumberOfEntries(); i++) {
            Student student = currentGroupStudentsList.getEntry(i);
            if (student.getStudentID().equals(studentToChange.getStudentID())) {
                studentIndexToRemove = i;
                break;
            }
        }

        if (studentIndexToRemove != -1) {
            currentGroupStudentsList.remove(studentIndexToRemove);
            System.out.println("Student moved to the new group successfully!");
        } else {
            System.out.println("Failed to move the student to the new group!");
        }
        TutorialGroupDAO.saveToFile(TutorialGroupList); // Save the updated group list to
file

        System.out.println("Press enter to continue...");
        scanner.nextLine();

    }


    public void listGroupStudent(){
        String programmeCode = null;
        char check = 'N';

        displayProgramme(null);
            System.out.println("Please enter programme code to view student or tutorial
group:");
        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
```

```
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
    }

     System.out.println("Do you want to view all students under " + programmeCode +
"? (Y/N)");
    check = scanner.next().charAt(0);
    scanner.nextLine(); // Consume newline character

    if (Character.toUpperCase(check) == 'Y') {
        // Display all students under the specified programme code
        System.out.println("List of Student Under " + programmeCode + ":");
        System.out.println(String.format("%-10s %-20s %-4s %-8s %-35s %-15s",
                "StudentID", "Student Name", "Age", "Gender", "Email", "PhoneNo"));
        for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
            Student student = studentList.getEntry(i);
            if (student.getProgrammeCode().equals(programmeCode)) {
                System.out.println(String.format("%-10s %-20s %-4s %-8s %-35s %-15s",
                        student.getStudentID(), student.getName(), student.getAge(),
                        student.getGender(), student.getEmail(), student.getPhoneNo()));
            }
        }

        // Display tutorial groups under the specified programme code
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the group ID to view students:");
        String groupID = scanner.nextLine();

        // Find the group by its ID
        int groupIndex = findGroupIndex(groupID);
        if (groupIndex == -1) {
            System.out.println("Group not found!");
            return;
        }

        // Get the group and display its students list
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        displayGroupDetail(group);
    } else {
        // Display tutorial groups under the specified programme code
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the group ID to view students:");
        String groupID = scanner.nextLine();
```

```
            // Find the group by its ID
            int groupIndex = findGroupIndex(groupID);
            if (groupIndex == -1) {
                System.out.println("Group not found!");
                return;
            }

            // Get the group and display its students list
            TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
            displayGroupDetail(group);
        }


    }

    public void mergeGroup(){
        String programmeCode = null;
        char check = 'N';
        int totalStudents = 0;

        // Display available programmes
        displayProgramme(null);
        System.out.println("Please enter programme code to merge tutorial groups:");
        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        }

        // Display tutorial groups for the specified programme code
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter two tutorial group IDs to merge (separated by comma):");

        // Get user input for group IDs
        String inputGroupIDs = scanner.nextLine();
        String[] groupIDs = inputGroupIDs.split(",");

        // Validate input group IDs
```

```
        if (groupIDs.length != 2 || groupIDs[0].equals(groupIDs[1])) {
            System.out.println("Please provide exactly two different group IDs separated by
comma.");
            return;
        }

        // Check if both groups exist and if they can be merged
        boolean group1Exists = findGroup(groupIDs[0]);
        boolean group2Exists = findGroup(groupIDs[1]);

        if (!group1Exists || !group2Exists) {
            System.out.println("Invalid group IDs!");
            return;
        }

        // Check if either group has more than 5 students
        for (String groupID : groupIDs) {
            int groupIndex = findGroupIndex(groupID);
            if (groupIndex != -1) {
                TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
                totalStudents += group.getGroupStudentsList().getNumberOfEntries();
            }
        }

        if (totalStudents > 5) {
            System.out.println("Cannot merge groups with more than 5 students total.");
            return;
        }

        // Merge groups
        mergeTwoGroups(groupIDs[0], groupIDs[1]);
        System.out.println("Tutorial groups merged successfully!");

        //Show the group Student List to make sure student add inside
        displayProgrammeGroup(programmeCode);
        System.out.println("Enter the group ID to view students:");
        String groupID = scanner.nextLine();

        int groupIndex = findGroupIndex(groupID);
        if (groupIndex == -1) {
            System.out.println("Group not found!");
            return;
        }

        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        displayGroupDetail(group);
```

```
    }

    public void report(){
        int choice = -1;
            do {
                choice = GroupManagementUI.reportChoice();
                switch (choice) {
                    case 0:
                        MessageUI.displayExitMessage();
                        break;
                    case 1:
                        programmeGroupReport();
                        break;
                    case 2:
                        TutorialGroupReport();
                        break;
                }
            } while (choice != 0);




    }

    public void programmeGroupReport(){
        System.out.println("=".repeat(150));
        System.out.printf("\n%57s", "");
                    System.out.println("TUNKU ABDUL RAHMAN UNIVERSITY OF
MANAGEMENT AND TECHNOLOGY");
        System.out.println("=".repeat(150));
        System.out.printf("\n%65s", "");
        System.out.println("Tutorial Group Management Subsystem");
        System.out.printf("\n%66s", "");
        System.out.println("Programme Group Summary Report");
        System.out.printf("\n%61s", "");
        System.out.println("----------------------------------------");
        System.out.println();
        System.out.println("Generated at: " + getCurrentDateTime());
        System.out.println();
                    System.out.format("%-10s %-80s %-30s %-30s", "Programme_Code",
"\tProgramme_Name", "\tTutorial_Group_Offer", "\tTotal_Student");
        System.out.println("");
```

```
        int totalStudentsAllProgrammes = 0;
        String mostTutorialGroupOfferProgrammeCode = "";
        int mostTutorialGroupOfferCount = 0;
                ListInterface<String>  fewestTutorialGroupOfferProgrammeCodes  =  new
ArrayList<>();
        int fewestTutorialGroupOfferCount = Integer.MAX_VALUE;
        String mostStudentsProgrammeCode = "";
        int mostStudentsCount = 0;
        String fewestStudentsProgrammeCode = "";
        int fewestStudentsCount = Integer.MAX_VALUE;

        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
           Programme programme = ProgrammeList.getEntry(i);
           String programmeCode = programme.getProgrammeCode();
           String programmeName = programme.getProgrammeName();

           int tutorialGroupOffer = 0;
           int totalStudents = 0;

           for (int j = 1; j <= TutorialGroupList.getNumberOfEntries(); j++) {
              TutorialGroup group = TutorialGroupList.getEntry(j);
              if (group.getProgrammeCode().equals(programmeCode)) {
                 tutorialGroupOffer++;

              }
           }

           for (int k = 1; k <= studentList.getNumberOfEntries(); k++) {
              Student student = studentList.getEntry(k);
              if (student.getProgrammeCode().equals(programmeCode)) {
                 totalStudents++;
              }
           }

           totalStudentsAllProgrammes += totalStudents;
            if (tutorialGroupOffer > mostTutorialGroupOfferCount && tutorialGroupOffer >
0) {
              mostTutorialGroupOfferCount = tutorialGroupOffer;
              mostTutorialGroupOfferProgrammeCode = programmeCode;
           }
             if (tutorialGroupOffer < fewestTutorialGroupOfferCount && tutorialGroupOffer
> 0) {
              fewestTutorialGroupOfferCount = tutorialGroupOffer;
              fewestTutorialGroupOfferProgrammeCodes.clear();
              fewestTutorialGroupOfferProgrammeCodes.add(programmeCode);
```

```
                    }if (tutorialGroupOffer == fewestTutorialGroupOfferCount &&
tutorialGroupOffer > 0) {
            fewestTutorialGroupOfferProgrammeCodes.add(programmeCode);
        }
        if (totalStudents > mostStudentsCount) {
            mostStudentsCount = totalStudents;
            mostStudentsProgrammeCode = programmeCode;
        }
        if (totalStudents < fewestStudentsCount && totalStudents > 0) {
            fewestStudentsCount = totalStudents;
            fewestStudentsProgrammeCode = programmeCode;
        }


        System.out.println("");
                System.out.format("%-15s %-90s %-27d %-20d%n", programmeCode,
programmeName, tutorialGroupOffer, totalStudents);
        }


    // Calculate total tutorial group offers across all programmes
    int totalTutorialGroupOffers = 0;
    for (int j = 1; j <= TutorialGroupList.getNumberOfEntries(); j++) {
        TutorialGroup group = TutorialGroupList.getEntry(j);
        totalTutorialGroupOffers++;
    }

    System.out.println("");
    System.out.println("-".repeat(150));


    //Display total tutorial group offer of all programme
            System.out.println("Total Tutorial Group Offers of All Programmes: " +
totalTutorialGroupOffers);

    // Display programme with the most tutorial group offers
                System.out.println("Programme Offer Most Tutorial Group : " +
mostTutorialGroupOfferProgrammeCode +
            " [" + mostTutorialGroupOfferCount + " tutorial group(s)]");

    // Display programme with the fewest tutorial group offers
    System.out.println("Programme Offer Fewest Tutorial Group : ");
    if (!fewestTutorialGroupOfferProgrammeCodes.isEmpty()) {
        System.out.print(fewestTutorialGroupOfferProgrammeCodes.getEntry(1));
                                                for (int i = 2; i <
fewestTutorialGroupOfferProgrammeCodes.getNumberOfEntries(); i++) {
```

```
                                                    System.out.print("    ,    " +
fewestTutorialGroupOfferProgrammeCodes.getEntry(i)); // Print subsequent programs
with comma separation
            }
            System.out.println(" [" + fewestTutorialGroupOfferCount + " tutorial group(s)]");
        } else {
            System.out.println("No programs found with the fewest tutorial groups.");
        }
        System.out.println("Note: [0] tutorial group offered is not counted");

        System.out.println("*".repeat(150));

        // Display total student count for all programmes
                    System.out.println("Total Student of All Programme    :    " +
totalStudentsAllProgrammes);

        // Display programme with the most students
                    System.out.println("Programme with the most student        :    " +
mostStudentsProgrammeCode +
                " [" + mostStudentsCount + " student(s)]");

        // Display programme with the fewest students
        System.out.println("Programme with the fewest student :  ");
        if (!fewestTutorialGroupOfferProgrammeCodes.isEmpty()) {
            System.out.print(fewestTutorialGroupOfferProgrammeCodes.getEntry(1));
                                                    for    (int    i    =    2;    i    <=
fewestTutorialGroupOfferProgrammeCodes.getNumberOfEntries(); i++) {
                                                    System.out.print("    ,    "    +
fewestTutorialGroupOfferProgrammeCodes.getEntry(i));
            }
            System.out.println(" [" + fewestStudentsCount + " student(s)]");
        } else {
            System.out.println("No programmes found with the fewest students.");
        }
        System.out.println("Note : [0] student(s) is not counted");

        System.out.println("*".repeat(150));

        System.out.printf("\n%63s", "");
        System.out.println("End of the Programme Group Summary Report\n");

        System.out.println("*".repeat(150));

        System.out.println("Press <ENTER> Key To Continue ...");
        scanner.nextLine();
        }
```

```
    public void TutorialGroupReport(){

        String programmeCode = null;
        char check = 'N';

        System.out.println("Please enter the programme code to view tutorial groups:");
        if (programmeCode == null || !findProgramme(programmeCode)) {
            do {
                                                                programmeCode      =
CourseManagementUI.inputProgrammeCode().toUpperCase();
                if (!findProgramme(programmeCode)) {
                    System.out.println("Invalid programme!");
                } else {
                    check = 'Y';
                }
                System.out.println();
            } while (check != 'Y');
        };



System.out.println("==================================================
================================================");
        System.out.println("                    TUNKU ABDUL RAHMAN UNIVERSITY OF
MANAGEMENT AND TECHNOLOGY");

System.out.println("==================================================
================================================\n");
        System.out.println("                       Tutorial Group Management Subsystem");
        System.out.println("                         Tutorial Group Summary Report");
        System.out.println("                    ----------------------------------------");
        System.out.println();
        System.out.println("Generated at: " + getCurrentDateTime());
        System.out.println();



        System.out.println("Summary Report of " + programmeCode);


                System.out.format("%-15s  %-30s  %-40s%n",  "Group_ID",  "Group_Name",
"GroupStudent");

System.out.println("----------------------------------------------------------------------------
```

```
------------");

        int totalStudents = 0;
        int totalStudentsInGroups = 0;
        int totalGroups = 0;
        int maxStudentsInGroup = 0;
        String groupWithMaxStudents = "";
        int idleGroupsCount = 0;
        StringBuilder idleGroupsStringBuilder = new StringBuilder();

        for (int k = 1; k <= studentList.getNumberOfEntries(); k++) {
            Student student = studentList.getEntry(k);
            if (student.getProgrammeCode().equals(programmeCode)) {
                totalStudents++;
            }
        }

        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
            TutorialGroup group = TutorialGroupList.getEntry(i);
            if (group.getProgrammeCode().equals(programmeCode)) {
                ListInterface<Student> students = group.getGroupStudentsList();
                totalStudentsInGroups += students.getNumberOfEntries();
                totalGroups++;

                if (students.isEmpty()) {
                    idleGroupsCount++;
                    idleGroupsStringBuilder.append(group.getGroupID());
                    idleGroupsStringBuilder.append(", ");
                } else if (students.getNumberOfEntries() > maxStudentsInGroup) {
                    maxStudentsInGroup = students.getNumberOfEntries();
                    groupWithMaxStudents = group.getGroupID();
                }

                StringBuilder studentIDsBuilder = new StringBuilder();
                for (int j = 1; j <= students.getNumberOfEntries(); j++) {
                    studentIDsBuilder.append(students.getEntry(j).getStudentID());
                    if (j < students.getNumberOfEntries()) {
                        studentIDsBuilder.append(", ");
                    }
                }
                String studentIDs = studentIDsBuilder.toString();
                if (studentIDs.isEmpty()) {
                    studentIDs = "NULL";
                }
                                System.out.format("%-15s  %-30s  [%s]%n",  group.getGroupID(),
group.getGroupName(), studentIDs);
```

```
            }
        }


System.out.println("-------------------------------------------------------------------------------
------------");
        System.out.println("Total Student of Programme  :  " + totalStudents);
            System.out.println("Number of student already assigned in a tutorial group: " +
totalStudentsInGroups);
            System.out.println("Number of student waiting for assigned in a tutorial group: " +
(totalStudents - totalStudentsInGroups));

System.out.println("**********************************************************
****************************************");
                        System.out.println("Tutorial Group with the most students: " +
groupWithMaxStudents + " --> [" + maxStudentsInGroup + "] students");
            System.out.println("Idle Tutorial Groups:  [ " + idleGroupsCount + " ]  --->" +
idleGroupsStringBuilder.toString());




System.out.println("\n**********************************************************
****************************************");

        System.out.println("\n                        End of the Programme Group Summary
Report\n");


System.out.println("**********************************************************
****************************************");

        System.out.println("Press <ENTER> Key To Continue ...");
        scanner.nextLine();

    }



    public boolean findProgramme(String programmeCode) {
        for (int i = 0; i < ProgrammeList.getNumberOfEntries()+1; i++) {
                                        if (ProgrammeList.getEntry(i) != null &&
ProgrammeList.getEntry(i).getProgrammeCode().equals(programmeCode)) {
                return true;
```

```
            }
        }
        return false;
    }

    public boolean findGroup(String groupID) {
        for (int i = 0; i < TutorialGroupList.getNumberOfEntries() +1; i++) {
                                        if (TutorialGroupList.getEntry(i) != null &&
TutorialGroupList.getEntry(i).getGroupID().equals(groupID)) {
                return true;
            }
        }
        return false;
    }

    private Student findStudent(String studentID) {
        if (studentID == null || studentID.isEmpty()) {
            // Handle invalid studentID (e.g., return null or throw an exception)
            return null;
        }

        // Assuming studentList is not null
        for (int i = 0; i < studentList.getNumberOfEntries()+1; i++) {
            Student student = studentList.getEntry(i);
                            if (student != null && student.getStudentID() != null &&
student.getStudentID().equals(studentID)) {
                return student;
            }
        }
        return null; // Student not found
    }

    private void displayGroupDetail(TutorialGroup group) {
        System.out.println("Student details for " + group.getGroupName() + ":");
                System.out.println(String.format("%-10s %-25s %-5s %-8s %-35s %-15s",
"studentID", "studentName", "age", "gender", "email", "phoneNo"));
        ListInterface<Student> students = group.getGroupStudentsList();
        for (int i = 1; i <= students.getNumberOfEntries(); i++) {
            Student student = students.getEntry(i);
            System.out.println(String.format("%-10s %-25s %-5d %-8s %-35s %-15s",
                    student.getStudentID(), student.getName(), student.getAge(),
                    student.getGender(), student.getEmail(), student.getPhoneNo()));
        }
    }

    private void mergeTwoGroups(String groupID1, String groupID2) {
```

```
        // Find the indices of the groups in the list
        int index1 = findGroupIndex(groupID1);
        int index2 = findGroupIndex(groupID2);

        if (index1 == -1 || index2 == -1) {
            System.out.println("Error: One or both groups not found.");
            return;
        }

        // Merge group2 into group1
        TutorialGroup group1 = TutorialGroupList.getEntry(index1);
        TutorialGroup group2 = TutorialGroupList.getEntry(index2);

        // Check if the total number of students in both groups is more than 5
                int totalStudents = group2.getGroupStudentsList().getNumberOfEntries() +
group1.getGroupStudentsList().getNumberOfEntries();
        if (totalStudents > 5) {
            System.out.println("Cannot merge groups with more than 5 students total.");
            return;
        }

        // Add students from group2 to group1
        ListInterface<Student> group1Students = group1.getGroupStudentsList();
        ListInterface<Student> group2Students = group2.getGroupStudentsList();

        for (int i = 1; i <= group2Students.getNumberOfEntries(); i++) {
            group1Students.add(group2Students.getEntry(i));
        }

        // Remove group2 from the list
        TutorialGroupList.remove(index2);
         TutorialGroupDAO.saveToFile(TutorialGroupList); // Save the updated group list to
file
    }

    private String getCurrentDateTime() {
        LocalDateTime currentDateTime = LocalDateTime.now();
                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        return currentDateTime.format(formatter);
    }

    public int findGroupIndex(String groupID) {
        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
                                            if (TutorialGroupList.getEntry(i) != null &&
TutorialGroupList.getEntry(i).getGroupID().equals(groupID)) {
```

```
            return i;
        }
    }
    return -1;
}

public String getAllProgramme(String programmeCode) {
    String outputStr = "";

    if (programmeCode == null) {
        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
            outputStr += ProgrammeList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
            if (programmeCode.equals(ProgrammeList.getEntry(i).getProgrammeCode()))
{
                outputStr += ProgrammeList.getEntry(i) + "\n";
            }
        }
    }
    if (outputStr.isEmpty()) {
        return "There are no programme taking this course!\n";
    }
    return outputStr;

}

public String getAllProgrammeGroup(String programmeCode) {
    String outputStr = "";

    if (programmeCode == null) {
        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
            outputStr += TutorialGroupList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
         TutorialGroup group = TutorialGroupList.getEntry(i);
         if (group.getProgrammeCode().equals(programmeCode)) {
            outputStr += group + "\n";
            }
        }
    }
    if (outputStr.isEmpty()) {
        return "There are no group taking this programme!\n";
    }
```

```
        return outputStr;

    }

    private String generateUniqueGroupID(String programmeCode) {
        int nextGroupNumber = getNextGroupNumber(programmeCode); // Get the next
available group number
        String formattedGroupNumber = String.format("%04d", nextGroupNumber);
        String newGroupID = programmeCode + formattedGroupNumber;

        // Check if the generated group ID already exists, generate a new one if needed
        while (findGroup(newGroupID)) {
            nextGroupNumber++;
            formattedGroupNumber = String.format("%04d", nextGroupNumber);
            newGroupID = programmeCode + formattedGroupNumber;
        }

        return newGroupID;
    }

    private int getGroupNumber(String groupID) {
        return Integer.parseInt(groupID.substring(groupID.length() - 4)); // Extract the group
number from the group ID
    }

    private int getNextGroupNumber(String programmeCode) {
                                                        int        lastGroupNumber        =
TutorialGroup.getLastGroupNumberMap().getOrDefault(programmeCode, 0);
        return lastGroupNumber + 1;
    }

    public void displayProgramme(String programmeCode) {
        CourseManagementUI.listAllProgramme(getAllProgramme(programmeCode));
    }

    public void displayProgrammeGroup(String programmeCode) {

GroupManagementUI.listAllTutorialGroup(getAllProgrammeGroup(programmeCode));
    }

}
```

2. Screenshots

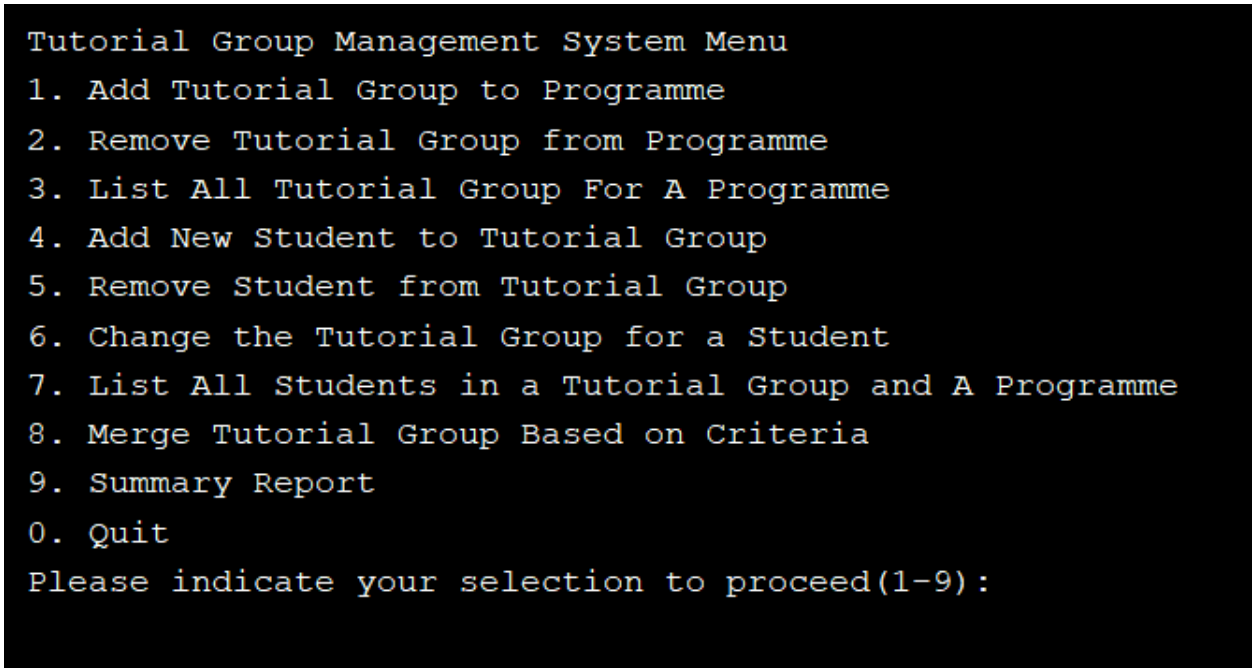


Figure 3.1:Menu of the Tutorial Group Management Subsystem

Figure 3.1 shows the menu of the Tutorial Group Management Subsystem and requires the user to input the option that he wants to proceed.

```
Please indicate your selection to proceed(1-9): 1
Programme Details:

-------------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-------------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter programme code that you want to add tutorial group
Enter Programme Code:
```

Figure 3.2: The list of the programme to add new tutorial group

Based on the Figure 3.2 After the user enters option 1, it will show the list of the programmes that are available to choose by the user. Users will be required to select the programme that wants to add a new tutorial group.

```
Please enter programme code that you want to add tutorial group
Enter Programme Code: RSW


Tutorial groups saved to file successfully.
New tutorial group added successfully:
RSW             RSW0002         RSW Group 2
```

Figure 3.3: New Tutorial Group

Based on the Figure 3.3, after selecting the programme, it will prompt out the message that successfully added a new tutorial group.Then it will show the details such as programme code, tutorial group id and tutorial group name. After that it will return to the tutorial group subsystem menu.

```
Please indicate your selection to proceed(1-9): 2
Programme Details:

-------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Enter Programme Code: rsw
```

Figure 3.4: The list of the programme to remove tutorial group

After the user enters option 2, it will show as figure 3.4 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that want to remove the tutorial group.

```
Enter Programme Code: rsw


List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RSW             RSW0001         RSW Group 1
RSW             RSW0002         RSW Group 2

Continue removing tutorial group from programme (Y/N): Y
Enter Tutorial Group ID: RSW0002

Tutorial groups saved to file successfully.
RSW0002 has been removed from RSW successfully!

Press enter to continue...
```

Figure 3.5: Remove tutorial group

Figure 3.5 shows that after the user input the programme code, it will show the list of the tutorial group that is under the programme. Then, it will double confirm whether the user wants to remove the tutorial group, if yes then it will require the user to input the groupID. Lastly, it will prompt out the message to tell the user successfully remove the tutorial group and then return to the tutorial group subsystem menu.

```
Please indicate your selection to proceed(1-9): 3
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter programme code that you want to seeEnter Programme Code: RDS
```

Figure 3.6: The list of the programme to view all tutorial group

After the user enters option 3, it will show the list of the programmes as figure 3.6 that are available to choose by the user. Users will be required to select the programme that wants to view the tutorial groups that under the programme.

```
Please enter programme code that you want to seeEnter Programme Code: RDS


List of Tutorial Groups for Programme :

-------------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-------------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4
```

Figure 3.7: The list of tutorial group under the programme

After the user input the programme code, it will list out all of the tutorial groups under that programme as shown in figure 3.7.

```
Please indicate your selection to proceed(1-9): 4
Programme Details:

-------------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-------------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter programme code to add students to a groupEnter Programme Code: rds
```

Figure 3.8: The list of the programme to add new student inside a tutorial group

After the user enters option 4, it will show as figure 3.8 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that wants to add a new student in the tutorial group.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4


Enter group ID to add students:RDS0001
Enter student IDs to add to the group (separated by comma):
S1002,S1005
Tutorial groups saved to file successfully.
Student S1002 added to group successfully!
Tutorial groups saved to file successfully.
Student S1005 added to group successfully!
Do you want to view detailed information about this group? (Y/N)
Y
Student details for RDS Group 1:
studentID  studentName              age   gender   email                          phoneNo
S1000      Tan Lock Kwan            21    Female   tanlk-wp21@student.tarc.edu.my      012-11112222
S1002      Goh Boon Xiang           22    Male     gohbx-wp21@student.tarc.edu.my      016-34098124
S1005      Tan Ling Ling            21    Female   tanll-wp21@student.tarc.edu.my      012-12341234
```

Figure 3.9 : The list of the tutorial group and student list

Figure 3.9 shows that after the user enters the programme code, it will show all of the tutorial classes that are under the programme and ask the user to input the groupID that wants to add a student. Then, it will require the user to input the studentID that he wants to add inside the tutorial group. Users are allowed to assign multiple students inside the tutorial group by separating studentID using comma.At the same time, it will have validation checking to check whether the student is already exist in another tutorial group or not, or already inside the tutorial group, or he is under the same programme or not.If no, then student will be successful add inside the tutorial group. Then it will ask the user whether he wants to see more detail about the tutorial group or not, if yes, then show the student list that contains the student information. After that, it will return to the tutorial group management subsystem menu.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4


Enter group ID to add students:RDS0002
Enter student IDs to add to the group (separated by comma):
S1002
Student S1002 is already in another tutorial group!
Do you want to view detailed information about this group? (Y/N)
N
```

Figure 3.10: Student already in other tutorial group

Figure 3.10 shows that if the student is already in another tutorial group, it will not able to add inside the tutorial group again.

```
Please indicate your selection to proceed(1-9): 5
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter the programme code:
Enter Programme Code: rds
```

Figure 3.11 :The list of the programme to remove student from tutorial group

After the user enters option 5, it will show as figure 3.11 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that wants to add a new student in the tutorial group.

```
Please enter the programme code:
Enter Programme Code: rds


List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter the group ID to remove a student:
RDS0001
Student details for RDS Group 1:
studentID  studentName              age   gender   email                              phoneNo
S1000      Tan Lock Kwan            21    Female   tanlk-wp21@student.tarc.edu.my     012-11112222
S1002      Goh Boon Xiang           22    Male     gohbx-wp21@student.tarc.edu.my     016-34098124
S1005      Tan Ling Ling            21    Female   tanll-wp21@student.tarc.edu.my     012-12341234
Enter the student ID to remove from the group:
S1005
Tutorial groups saved to file successfully.
Student removed from the group successfully!
```

Figure 3.12: Remove student from a tutorial group

Figure 3.12 shows that the user will be required to input the current tutorial group of the student that wants to remove. Then it will show the student list of that tutorial group and ask the user to input the studentID to remove the student. Lastly, it will prompt out the successful message and then return to the tutorial group management subsystem menu.

```
Please indicate your selection to proceed(1-9): 6
Programme Details:

-----------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter the programme code:
Enter Programme Code: RDS
```

Figure 3.13: The list of the programme to change the tutorial group for a student

After the user enters option 6, it will show as figure 3.12 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that wants to change the tutorial group for a student.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter the current group ID of the student:
RDS0003
```

Figure 3.14: The list of tutorial group

Figure 3.14 shows that the user will be required to input the current tutorial group ID of the student to change the tutorial group.

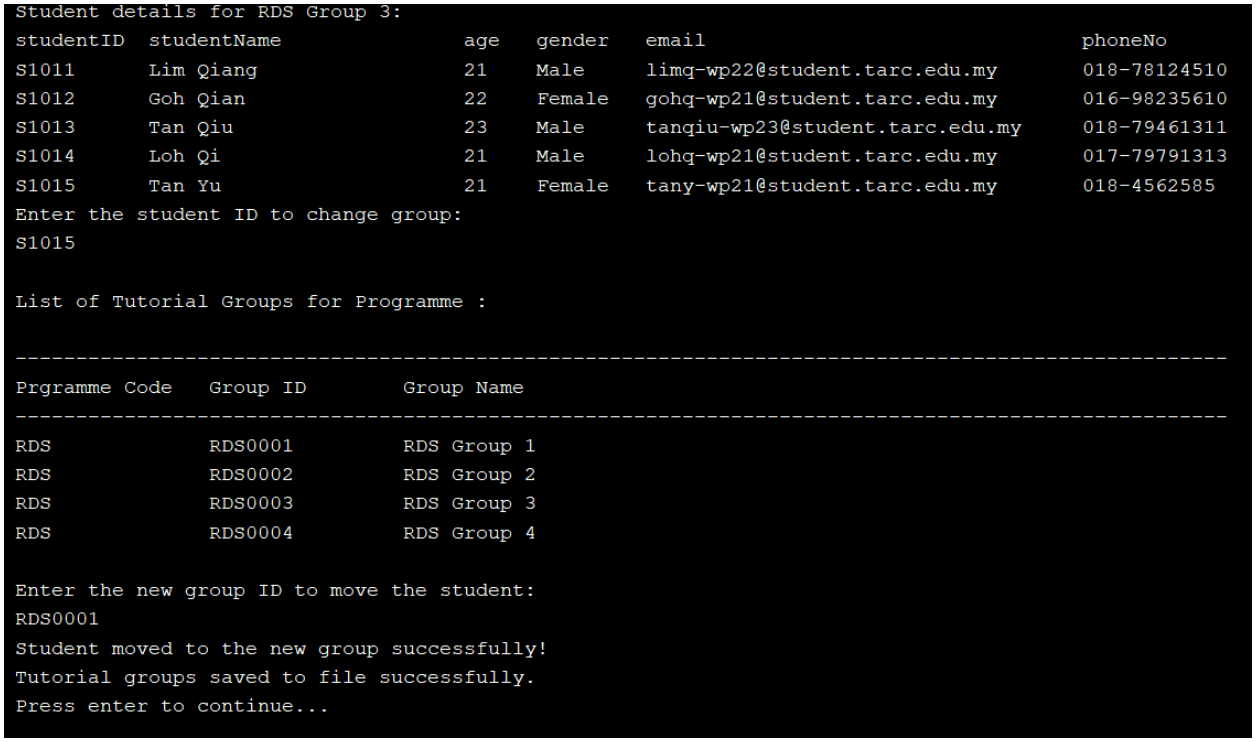


Figure 3.15: Student change tutorial group

Figure 3.15 shows that after the user input the current tutorial group id of the student it will show the student list of the tutorial group and then require the user to input the studentID of the student that changes to another tutorial group. Next, it will require the user to input the groupID that the student wants to change. Then it will prompt out the successful message and return to the tutorial group management subsystem menu.

```
Please indicate your selection to proceed(1-9): 7
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter programme code to view student or tutorial group:
Enter Programme Code: RDS
```

Figure 3.16 : The list of programme to view all student in programme and tutorial group

After the user enters option 7, it will show as figure 3.16 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that wants to view all students in that programme and also the tutorial group.

```
Please enter programme code to view student or tutorial group:
Enter Programme Code: RDS

Do you want to view all students under RDS? (Y/N)
Y
List of Student Under RDS:
StudentID  Student Name         Age  Gender   Email                              PhoneNo
S1000      Tan Lock Kwan        21   Female   tanlk-wp21@student.tarc.edu.my     012-11112222
S1002      Goh Boon Xiang       22   Male     gohbx-wp21@student.tarc.edu.my     016-34098124
S1005      Tan Ling Ling        21   Female   tanll-wp21@student.tarc.edu.my     012-12341234
S1010      Tan Qing             21   Female   tanq-wp21@student.tarc.edu.my      012-4561230
S1011      Lim Qiang            21   Male     limq-wp22@student.tarc.edu.my      018-78124510
S1012      Goh Qian             22   Female   gohq-wp21@student.tarc.edu.my      016-98235610
S1013      Tan Qiu              23   Male     tanqiu-wp23@student.tarc.edu.my    018-79461311
S1014      Loh Qi               21   Male     lohq-wp21@student.tarc.edu.my      017-79791313
S1015      Tan Yu               21   Female   tany-wp21@student.tarc.edu.my      018-4562585
S1016      Lim Yao              21   Male     limy-wp22@student.tarc.edu.my      017-75395120
S1019      Loh Ying             21   Female   lohy-wp21@student.tarc.edu.my      014-02175855

List of Tutorial Groups for Programme :

-------------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-------------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter the group ID to view students:
```

Figure 3.17: The list of the student under the programme

Figure 3.17 shows that after the user input the programme code he wants, it will ask whether the user wants to view all of the students that are under the programme. If yes, then it will list out all of the students. If no, then it will print out all of the tutorial groups that are under the programme.Then it will ask the user to input the groupID to view the student list of that tutorial group.

```
Enter the group ID to view students:
RDS0001
Student details for RDS Group 1:
studentID  studentName             age   gender   email                          phoneNo
S1000      Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my 012-11112222
S1002      Goh Boon Xiang          22    Male     gohbx-wp21@student.tarc.edu.my 016-34098124
S1015      Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my  018-4562585
```

Figure 3.18: The list of the student inside the tutorial group

Figure 3.18 shows the student list of the tutorial group.Inside the student list, it will show more detailed information about the student including the studentID , student name , age , gender, email and phone number.

```
Please indicate your selection to proceed(1-9): 8
Programme Details:

-----------------------------------------------------------------------------------------------------
Programme Code       Programme Name
-----------------------------------------------------------------------------------------------------
RDS                  Bachelor Of Computer Science (HONOURS) In Data Science
RSW                  Bachelor Of Software Engineering (HONOURS)
RSD                  Bachelor Of Information Technology (HONOURS) In Software Systems Development
RMM                  Bachelor Of Science (Honours) In Management Mathematics With Computing
RIS                  Bachelor Of Information Technology (Honours) In Information Security
RST                  Bachelor Of Computer Science (Honours) In Interactive Software Technology
REI                  Bachelor Of Information Systems (Honours) In Enterprise Information Systems
RBS                  Bachelor Of Science (Hons) In Bioscience With Chemistry
RAN                  Bachelor Of Science (Hons) In Analytical Chemistry
RFN                  Bachelor Of Science (Hons) In Food Science
RNR                  Bachelor Of Science (Hons) In Nutrition
RSE                  Bachelor Of Science (Hons) In Sports And Exercise Science
REE                  Bachelor Of Electrical And Electronics Engineering With Honours
RME                  Bachelor Of Mechanical Engineering With Honours
RMT                  Bachelor Of Engineering (Honours) Material

Please enter programme code to merge tutorial groups:
Enter Programme Code: RDS
```

Figure 3.19 :The list of the programme to merge two tutorial group

After the user enters option 7, it will show as figure 3.19 that the list of the programmes that are available to choose by the user . Users will be required to select the programme that wants to merge the tutorial group.

```
Please enter programme code to merge tutorial groups:
Enter Programme Code: RDS


List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter two tutorial group IDs to merge (separated by comma):
RDS0001,RDS0004
Tutorial groups saved to file successfully.
Tutorial groups merged successfully!
```

Figure 3.20 : The list of the tutorial group that can merge together

After the user input the programme code, it will show as figure 3.20 that all tutorial groups under the programme, then require the user to input two tutorial groups that he wants to merge together

and separate by the comma. At the same time, it will also have validation checking to check whether the tutorial group that wants to merge together has more than 5 students or not. If no, then it will be successful to merge two tutorial groups together. At the same time, the students inside the tutorial group will be merged together also.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3

Enter the group ID to view students:
RDS0001
Student details for RDS Group 1:
studentID  studentName              age   gender   email                           phoneNo
S1000      Tan Lock Kwan            21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222
S1002      Goh Boon Xiang           22    Male     gohbx-wp21@student.tarc.edu.my  016-34098124
S1015      Tan Yu                   21    Female   tany-wp21@student.tarc.edu.my   018-4562585
S1016      Lim Yao                  21    Male     limy-wp22@student.tarc.edu.my   017-75395120
```

Figure 3.21 : Student list after merge tutorial group

Figure 3.21 shows that after the tutorial group has successfully merged together, it will list out the tutorial group that currently have and require the user to input the tutorial groupID to view the students inside the tutorial group. At the time, users can check whether the students are successfully merged together or not. Then it will return to the tutorial group management subsystem menu.

```
Please indicate your selection to proceed(1-9): 9

Summary Report Menu
1. Programme Group Summary Report
2. Tutorial Group Summary Report
0. Back
Please select the report you would like to display(1-2): 0


Exiting system
```

Figure 3.22: Summary report menu

Figure 3.22 shows that after the user enters option 9, it will bring the user to the summary report menu and require the user to select which summary report he wants to view. If the user input 0 then will exit the summary report menu and return to the tutorial group management subsystem menu.

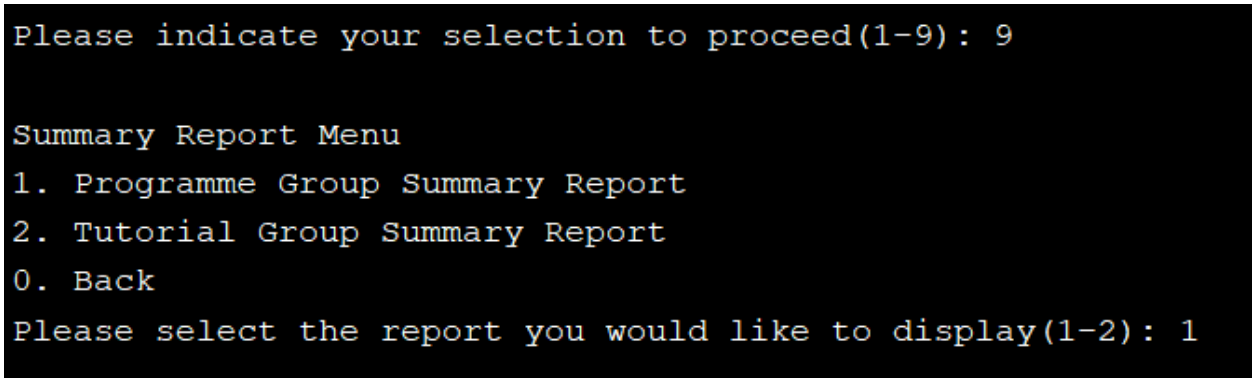


Figure 3.23 : Summary report menu of programme group summary report

Figure 3.23 shows the user input option 1 to view the programme group summary report.

```
================================================================================================================================================================

                                                TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY
================================================================================================================================================================

                                                        Tutorial Group Management Subsystem

                                                         Programme Group Summary Report

                                                       ----------------------------------------

Generated at: 2024-04-20 03:03:16

Programme_Code   Programme_Name                                                                  Tutorial_Group_Offer            Total_Student

RDS              Bachelor Of Computer Science (HONOURS) In Data Science                                3                          11

RSW              Bachelor Of Software Engineering (HONOURS)                                            2                          2

RSD              Bachelor Of Information Technology (HONOURS) In Software Systems Development          1                          1

RMM              Bachelor Of Science (Honours) In Management Mathematics With Computing                1                          1

RIS              Bachelor Of Information Technology (Honours) In Information Security                   1                          1

RST              Bachelor Of Computer Science (Honours) In Interactive Software Technology             1                          0

REI              Bachelor Of Information Systems (Honours) In Enterprise Information Systems           1                          0

RBS              Bachelor Of Science (Hons) In Bioscience With Chemistry                               1                          1

RAN              Bachelor Of Science (Hons) In Analytical Chemistry                                    2                          2

RFN              Bachelor Of Science (Hons) In Food Science                                            1                          1

RNR              Bachelor Of Science (Hons) In Nutrition                                               1                          0

RSE              Bachelor Of Science (Hons) In Sports And Exercise Science                             1                          6

REE              Bachelor Of Electrical And Electronics Engineering With Honours                       1                          3

RME              Bachelor Of Mechanical Engineering With Honours                                       1                          3

RMT              Bachelor Of Engineering (Honours) Material                                            3                          2

--------------------------------------------------------------------------------------------------------------------------------------------------------------
Total Tutorial Group Offers of All Programmes: 21
Programme Offer Most Tutorial Group  :  RDS [3 tutorial group(s)]
Programme Offer Fewest Tutorial Group :
RSD , RSD , RMM , RIS , RST , REI , RBS , RFN , RNR , RSE , REE [1 tutorial group(s)]
Note: [0] tutorial group offered is not counted
**************************************************************************************************************************************************************
Total Student of All Programme  :  34
Programme with the most student    :  RDS [11 student(s)]
Programme with the fewest student :
RSD , RSD , RMM , RIS , RST , REI , RBS , RFN , RNR , RSE , REE , RME [1 student(s)]
Note : [0] student(s) is not counted
**************************************************************************************************************************************************************

                                                        End of the Programme Group Summary Report

**************************************************************************************************************************************************************
Press <ENTER> Key To Continue ...
```

Figure 3.24 :Programme group summary report

Figure 3.24 shows the programme summary report. Inside the programme group summary report, it will show the generated time first and then list out all of the programme code, programme name, total number of the tutorial group that programme offers, and total number of the student that is under the programme.Then it will have the summary detail including total tutorial group offers of all programmes, programmes offer most tutorial tutorial group, programme offer fewest tutorial group, total student of all programme, programme with that having most student and fewest student. After the user input enter key, it will return to the summary report menu.

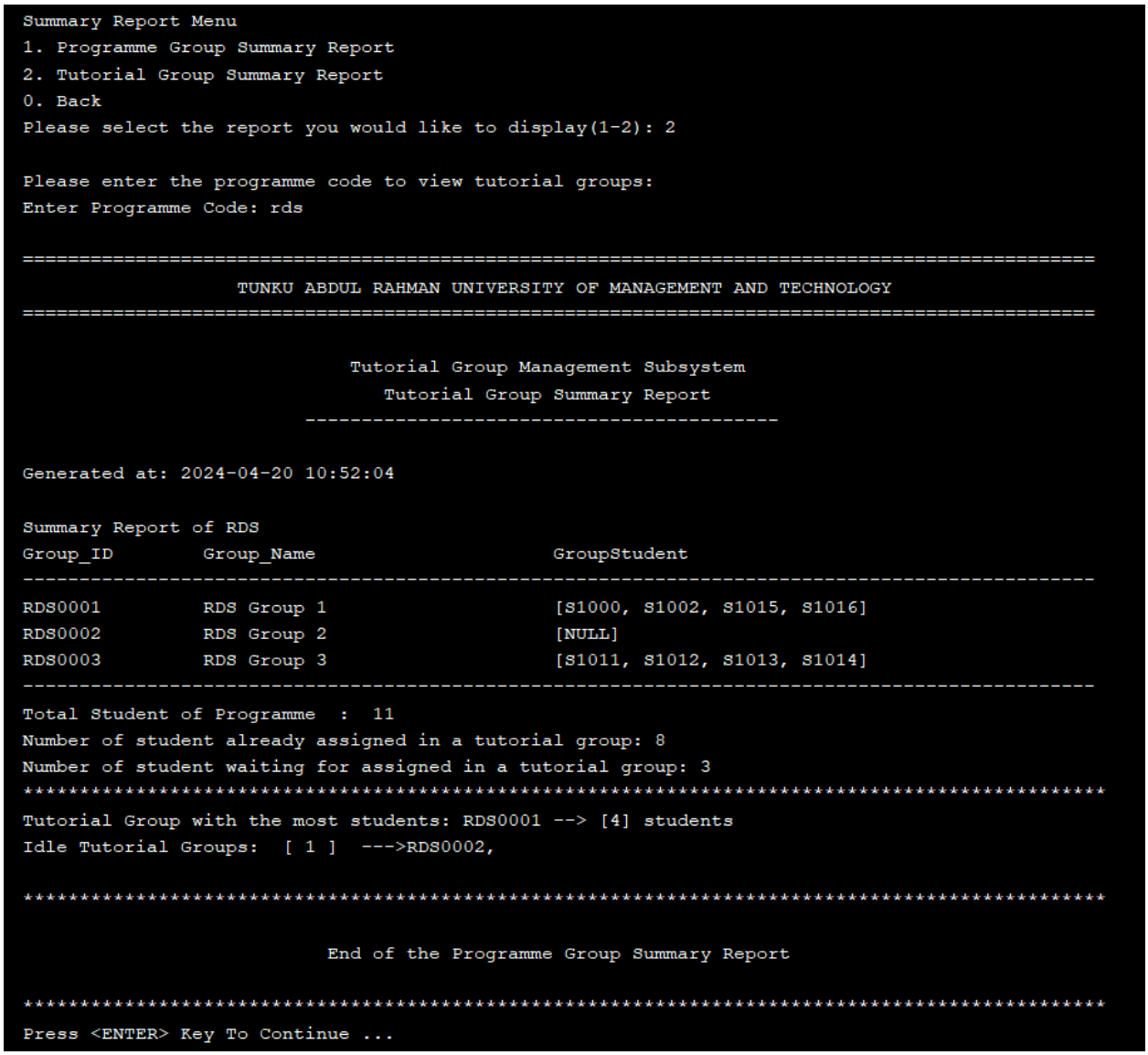


Figure 3.25 : Tutorial group summary report

Figure 3.25 shows the tutorial group summary report. First, it will ask the user to input the programme code that wants to summarize then generate the summary report. Inside the tutorial group summary report, it will show the generated time first and then list out all of the tutorials under the programme code, and also list the group student of each tutorial group. Then it will have the summary detail including total number of the students under the programme, number of students that are already assigned in a tutorial group, number of students that are still waiting to be assigned in a tutorial group, tutorial group with the most students, and the idle tutorial group. After the user input enter key, it will return to the summary report menu and then user can choose to input 0 to return to the tutorial group management subsystem menu.

Name	Student ID	Prog / Tut.Grp	Signature
Lim Hoi Yau	2214081	RDS2S2G3	*LHY*

Subsystem : Assignment Team Management Subsystem

1. **Source codes for Control classes**

```
package control;

import adt.*;
import boundary.*;
import dao.*;
import entity.AssignmentTeam;
import entity.Student;
import entity.TutorialGroup;
import entity.Programme;
import entity.courseProgramme;
import java.util.Scanner;
import utility.MessageUI;

/**
 *
 * @author Lim Hoi Yau
 */
public class AssignmentTeamManagement {

    Scanner scanner = new Scanner(System.in);

private ListInterface<AssignmentTeam> AssignmentTeamList = new ArrayList<>();
private AssignmentTeamDAO AssignmentTeamDAO = new AssignmentTeamDAO();
private AssignmentTeamInitializer AssignmentTeamInitializer = new
AssignmentTeamInitializer();
private AssignmentTeamManagementUI AssignmentTeamManagementUI = new
AssignmentTeamManagementUI();
```

```
private StudentDAO StudentDAO = new StudentDAO();
private StudentInitializer StudentInitializer = new StudentInitializer();
private ListInterface<Student> studentList = new ArrayList<>();

private ListInterface<TutorialGroup> TutorialGroupList = new ArrayList<>();
private TutorialGroupDAO TutorialGroupDAO = new TutorialGroupDAO();
private TutorialGroupInitializer TutorialGroupInitializer = new TutorialGroupInitializer();
private TutorialGroupManagement tutorialGroupManagement = new
TutorialGroupManagement();
private TutorialGroupManagementUI tutorialGroupManagementUI = new
TutorialGroupManagementUI();

private CourseManagement courseManagement = new CourseManagement();
private CourseManagementUI courseManagementUI = new CourseManagementUI();

private ProgrammeDAO ProgrammeDAO = new ProgrammeDAO();

ListInterface<Programme> ProgrammeList = new ArrayList<>();
private ListInterface<courseProgramme> CourseProgrammeList = new ArrayList<>();
ListInterface<Programme> programmeSortedList = new ArrayList<>();

public AssignmentTeamManagement() {
      studentList = StudentDAO.retrieveFromFile();
      TutorialGroupList = TutorialGroupDAO.retrieveFromFile();
      AssignmentTeamList = AssignmentTeamDAO.retrieveFromFile();
      ProgrammeList = ProgrammeDAO.retrieveFromFile();
   }

public static void main(String[] args) {
      AssignmentTeamManagement assignmentTeamManagement = new
AssignmentTeamManagement();
      assignmentTeamManagement.runAssignmentTeamManagement();
   }

public void runAssignmentTeamManagement() {
      int choice = 0;
      do {
         choice = AssignmentTeamManagementUI.getMenuChoice();
         switch (choice) {
            case 0:

               break;
            case 1:
               createAssignmentTeam(null, null, 'N');
               break;
            case 2:
```

```
                removeAssignmentTeam();
                break;
            case 3:
                editAssignmentTeam();
                break;
            case 4:
                addStudentToAssignmentTeam();
                break;
            case 5:
                removeStudentFromAssignmentTeam();
                break;
            case 6:
                mergeAsgTeams();
                break;
            case 7:
                listAssignmentTeamOfGroup();
                break;
            case 8:
                listStudentOfAsgTeam();
                break;
            case 9:
                Report();
                break;
        }
    } while (choice != 0);
  }

/*Create Assignment Team*/
public void createAssignmentTeam(String programmeCode, String tutorialGroupID, char check)
{
    courseManagement.displayAllProgramme(null);

          if (programmeCode == null || !courseManagement.findProgramme(programmeCode))
{
        do {
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');
    }

    System.out.println();
```

```
        tutorialGroupManagement.displayProgrammeGroup(programmeCode);
        check = 'N';

        if (tutorialGroupID == null || !courseManagement.findProgramme(programmeCode)) {
            do {
                tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
                if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                    System.out.println("Invalid Tutorial Group ID!");

                } else {
                    check = 'Y';

                }
                System.out.println();
            } while (check != 'Y');
        }

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        displayAssignmentTeam(group);
        int createTeamNumber = AssignmentTeamManagementUI.inputCreateTeamNumber();
        int asgTeamSize = 0;
        if (createTeamNumber != -1) {
            asgTeamSize = AssignmentTeamManagementUI.inputAssignmentTeamSize();
        }
        int currentAsgTeamIndex = getTutorialGroupAsgTeamNum(group);
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); ++i) {
                                        if ((tutorialGroupID + "T" + (currentAsgTeamIndex +
1)).equals(AssignmentTeamList.getEntry(i).getAssignmentTeamID())) {
                currentAsgTeamIndex++;
            }
        }
        if (createTeamNumber != -1 && asgTeamSize != -1) {
            for (int i = 1; i <= createTeamNumber && createTeamNumber != -1; ++i) {
                AssignmentTeam newAssignmentTeam = new AssignmentTeam();
                            newAssignmentTeam.setAssignmentTeamID(tutorialGroupID + "T" + (i +
currentAsgTeamIndex));
                    newAssignmentTeam.setAssignmentTeamName(tutorialGroupID + " Assignment
Team " + (i + currentAsgTeamIndex));
                newAssignmentTeam.setTutorialGroup(group);
                newAssignmentTeam.setAssignmentTeamSize(asgTeamSize);
                AssignmentTeamList.add(newAssignmentTeam);
            }
            AssignmentTeamDAO.saveToFile(AssignmentTeamList);
            displayAssignmentTeam(group);
```

```
            AssignmentTeamManagementUI.CreateAssignmentTeam();

            System.out.println("\nPress enter to continue...");
            scanner.nextLine();
        }
    }

/*Remove Assignment Team*/
    public void removeAssignmentTeam() {
        char check = 'N';
        String programmeCode;
        String assignmentTeamID = "";
        String tutorialGroupID;

            do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);

            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);
```

```
        check = 'N';

        while (check == 'N') {
                                                                    assignmentTeamID        =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
            if (checkAsgTeam(group, assignmentTeamID) || assignmentTeamID.equals("-1")) {
                check = 'Y';
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        if (!assignmentTeamID.equals("-1")) {
            int indexToRemove = -1;
            for (int i = 0; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                if (AssignmentTeamList.getEntry(i) != null) {
                    if (AssignmentTeamList.getEntry(i).getTutorialGroup().equals(group)
                                                                                        &&
AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(assignmentTeamID)) {
                        indexToRemove = i;
                        break;
                    }
                }
            }

            do {
                System.out.print("Continue removing assignment team from tutorial group (Y/N): ");
                check = Character.toUpperCase(scanner.next().charAt(0));
                if (check != 'Y' && check != 'N') {
                    System.out.println("Invalid Input, Please Try Again");
                }
                scanner.nextLine();
            } while (check != 'Y' && check != 'N');

            if (check == 'Y') {
                if (indexToRemove != -1) {
                    AssignmentTeamList.remove(indexToRemove);
                    AssignmentTeamDAO.saveToFile(AssignmentTeamList);
                            System.out.println(assignmentTeamID + " has been removed from " +
tutorialGroupID + " successfully!");
                } else {
                        System.out.println("Failed to remove " + assignmentTeamID + " from " +
tutorialGroupID + " !");
                }
```

```
            } else {
                System.out.println("Assignment Team remove canceled!");
            }

            System.out.println("\nPress enter to continue...");
            scanner.nextLine();

        }
    }

/* Amend Assignment Team Details */
    public void editAssignmentTeam() {
        char check = 'N';
        String assignmentTeam = "";
        String programmeCode;
        String tutorialGroupID;

        do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
```

```
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);

        while (check == 'N') {
                                                                    assignmentTeam =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
            if (checkAsgTeam(group, assignmentTeam) || assignmentTeam.equals("-1")) {
               check = 'Y';
            } else {
               System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        if (!assignmentTeam.equals("-1")) {
            int indexToEdit = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                                                        if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(assignmentTeam)) {
                   indexToEdit = i;
                   break;
               }
            }
            if (indexToEdit != -1) {
               int typeEditChoice = 0;
               AssignmentTeam editAsgTeam = AssignmentTeamList.getEntry(indexToEdit);
               do {
                  typeEditChoice = AssignmentTeamManagementUI.getTypeEditChoice();
                  switch (typeEditChoice) {
                     case 0:
                        MessageUI.displayExitMessage();
                        break;
                     case 1:
                        String oldTeamName = editAsgTeam.getAssignmentTeamName();
                                System.out.println("Old Team Name for " + assignmentTeam + ": " +
oldTeamName);
                                                                String newTeamName =
AssignmentTeamManagementUI.inputAssignmentTeamName();
                        if (newTeamName.equals("-1")) {
                           break;
                        }
                        if (newTeamName.equals(oldTeamName)) {
                           System.out.println("Invalid input, you have enter an old team name!");
                           System.out.println("\nPress enter to continue...");
```

```
                    scanner.nextLine();
                    break;
                }
                editAsgTeam.setAssignmentTeamName(newTeamName);
                                System.out.println("Team name had been changed to " +
editAsgTeam.getAssignmentTeamName());
                break;

            case 2:
                int oldTeamSize = editAsgTeam.getAssignmentTeamSize();
                        System.out.println("Old Team Size for " + assignmentTeam + ": " +
oldTeamSize);
                                                                int newTeamSize =
AssignmentTeamManagementUI.inputAssignmentTeamSize();
                if (newTeamSize == -1) {
                    break;
                }
                if (newTeamSize == oldTeamSize) {
                    System.out.println("Invalid input, you have enter an old team size!");
                    System.out.println("\nPress enter to continue...");
                    scanner.nextLine();
                    break;
                }
                if (newTeamSize < editAsgTeam.getTeamMember().getNumberOfEntries()) {
                    System.out.println("Current Member is Exceeding New Team Size, Please
Remove Member First");
                    System.out.println("\nPress enter to continue...");
                    scanner.nextLine();
                    break;
                }
                ListInterface<Student> newTeamMember = new ArrayList<>(newTeamSize);
                ListInterface<Student> teamMemberCopy = editAsgTeam.getTeamMember();
                for (int i = 1; i <= teamMemberCopy.getNumberOfEntries(); i++) {
                    newTeamMember.add(teamMemberCopy.getEntry(i));
                }
                editAsgTeam.setAssignmentTeamSize(newTeamSize);
                editAsgTeam.setTeamMember(newTeamMember);
                                System.out.println("Team size had been changed to " +
editAsgTeam.getAssignmentTeamSize());
                break;

            case 3:
                String oldTeamLeader;
                if (editAsgTeam.getTeamLeader() != null) {
                    oldTeamLeader = editAsgTeam.getTeamLeader().getName();
                        System.out.println("Old Team Leader for " + assignmentTeam + ": " +
```

```
oldTeamLeader);
                    }
                    displayStudentForAsgTeam(editAsgTeam);
                    if (editAsgTeam.getTeamMember().isEmpty()) {
                        System.out.println("No Student Can Choose to be the Team Leader");
                        System.out.println("\nPress enter to continue...");
                        scanner.nextLine();
                        break;
                    }
                                                                String newTeamLeaderID =
AssignmentTeamManagementUI.inputStudentID().toUpperCase();
                    if (newTeamLeaderID.equals("-1")) {
                        break;
                    }
                    Student newTeamLeader = retrieveStudent(newTeamLeaderID);
                    if (checkStudentExistsInTeam(editAsgTeam, newTeamLeader)) {
                        editAsgTeam.setTeamLeader(newTeamLeader);
                        if (editAsgTeam.getTeamLeader() != null) {
                                        System.out.println("Team Leader had been assigned to " +
editAsgTeam.getTeamLeader().getName());
                        }
                        break;
                    } else {
                        System.out.println("Target Student doesn't exists in this group");
                        break;
                    }

                }
            } while (typeEditChoice != 0);
            AssignmentTeamList.replace(indexToEdit, editAsgTeam);
            AssignmentTeamDAO.saveToFile(AssignmentTeamList);
        } else {
            System.out.println("Assignment Team with ID " + assignmentTeam + " is not found in
the list!\n");
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }
  }

/* Add Student To Assignment Team*/
  public void addStudentToAssignmentTeam() {
      char check = 'N';
      String assignmentTeam = "";
      String programmeCode;
      String tutorialGroupID;
```

```
        String studentID;

        do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);

        while (check == 'N') {
                                                                    assignmentTeam =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
            if (checkAsgTeam(group, assignmentTeam) || assignmentTeam.equals("-1")) {
                check = 'Y';
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }
```

```
        check = 'N';

        if (!assignmentTeam.equals("-1")) {
            int indexToAddStudent = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                                                                    if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(assignmentTeam)) {
                    indexToAddStudent = i;
                    break;
                }
            }

                                                    AssignmentTeam      addStudentAsgTeam      =
AssignmentTeamList.getEntry(indexToAddStudent);
             ListInterface<Student> asgTeamStudentsList = addStudentAsgTeam.getTeamMember();
            Programme programme = new Programme();
            programme.setProgrammeCode(programmeCode);
            displayStudentForAsgTeam(addStudentAsgTeam);
            while (check == 'N') {

                if (asgTeamStudentsList.isFull()) {

                        System.out.println("Cannot add more student to the group because the group is
full");
                    check = 'Y';
                    System.out.print("\nPress enter to continue...");
                    scanner.nextLine();
                } else {

                    System.out.println("Enter 0 to view student in the tutorial group\n");
                    studentID = AssignmentTeamManagementUI.inputStudentID().toUpperCase();
                    if (studentID.equals("0")) {
                        displayStudentForTutorialGroup(group);
                    } else if (studentID.equals("-1")) {
                        check = 'Y';
                    } else {
                        if (!checkStudentExists(group, studentID)) {
                            System.out.println("Student Does Not Exists, Please Try Again.");

                        } else {
                            Student student = retrieveStudent(studentID);
                              if (student != null && student.getProgrammeCode().equals(programmeCode))
{
                                 if (!checkStudentAsgTeamStatus(addStudentAsgTeam, student)) {
```

```
                        asgTeamStudentsList.add(student);

addStudentAsgTeam.setNumOfMember(asgTeamStudentsList.getNumberOfEntries());
                        AssignmentTeamList.replace(indexToAddStudent, addStudentAsgTeam);
                        AssignmentTeamDAO.saveToFile(AssignmentTeamList);
                        System.out.println("Students added to assignment team successfully!");
                        displayStudentForAsgTeam(addStudentAsgTeam);
                    } else {
                            System.out.print("Cannot assign student to this group since student
already assign in ");
                              System.out.println(findStudentCurrentTeam(addStudentAsgTeam,
student));
                        displayStudentForAsgTeam(addStudentAsgTeam);
                    }
                } else {
                    System.out.println("Student " + studentID + " not found in this group!");
                    displayStudentForAsgTeam(addStudentAsgTeam);
                }

            }
            System.out.println();
        }
    }
  }
  }

/* Remove Student From Assignment Team */
   public void removeStudentFromAssignmentTeam() {
      char check = 'N';
      String assignmentTeam = "";
      String programmeCode;
      String tutorialGroupID;
      String studentID;

      do {
         courseManagement.displayAllProgramme(null);
         programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
         if (!courseManagement.findProgramme(programmeCode)) {
            System.out.println("Invalid programme!");
         } else {
            check = 'Y';
         }
         System.out.println();
      } while (check == 'N');
```

```
        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);

        while (check == 'N') {
                                                                    assignmentTeam      =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
            if (checkAsgTeam(group, assignmentTeam) || assignmentTeam.equals("-1")) {
                check = 'Y';
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        check = 'N';

        if (!assignmentTeam.equals("-1")) {
            int indexOfAsgTeam = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                                                                    if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(assignmentTeam)) {
                    indexOfAsgTeam = i;
                    break;
                }
            }
```

```
                                    AssignmentTeam removeStudentAsgTeam =
AssignmentTeamList.getEntry(indexOfAsgTeam);
                                    ListInterface<Student> asgTeamStudentsList =
removeStudentAsgTeam.getTeamMember();
        Programme programme = new Programme();
        programme.setProgrammeCode(programmeCode);
        displayStudentForAsgTeam(removeStudentAsgTeam);
        while (check == 'N') {

          if (asgTeamStudentsList.isEmpty()) {

              System.out.println("Cannot remove student from the group because the group is
Empty");
            check = 'Y';
            System.out.print("\nPress enter to continue...");
            scanner.nextLine();
          } else {

            studentID = AssignmentTeamManagementUI.inputStudentID().toUpperCase();

            if (studentID.equals("-1")) {
              check = 'Y';
            } else {
              Student removeStudent = retrieveStudent(studentID);
              if (!checkStudentExistsInTeam(removeStudentAsgTeam, removeStudent)) {
                System.out.println("Member Does Not Exists, Please Try Again.");

              } else {
                int indexToRemove = -1;
                for (int i = 0; i < asgTeamStudentsList.getNumberOfEntries() + 1; i++) {
                  if (asgTeamStudentsList.getEntry(i) != null) {
                                                                    if
(asgTeamStudentsList.getEntry(i).getStudentID().equals(removeStudent.getStudentID())) {
                      indexToRemove = i;
                      break;
                    }
                  }
                }
                                    if (removeStudentAsgTeam.getTeamLeader() != null &&
asgTeamStudentsList.getEntry(indexToRemove).getStudentID().equals(removeStudentAsgTeam
.getTeamLeader().getStudentID())) {
                  removeStudentAsgTeam.setTeamLeader(null);
                }

removeStudentAsgTeam.setNumOfMember(asgTeamStudentsList.getNumberOfEntries() - 1);
                AssignmentTeamList.replace(indexOfAsgTeam, removeStudentAsgTeam);
```

```
                    asgTeamStudentsList.remove(indexToRemove);

                    AssignmentTeamDAO.saveToFile(AssignmentTeamList);
                    System.out.println("Students remove from assignment team successfully!");
                    displayStudentForAsgTeam(removeStudentAsgTeam);

                }
                System.out.println();
            }
        }
    }
}

/* Merge Assignment Team */
    public void mergeAsgTeams() {
        char check = 'N';
        String firstAssignmentTeam = "";
        String secondAssignmentTeam = "";
        String programmeCode;
        String tutorialGroupID;

        do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
```

```
        } while (check == 'N');

        check = 'N';

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);
        System.out.println("Enter assignment team ID you want to merge.\n");

        while (check == 'N') {
                                                                    firstAssignmentTeam =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
            if (checkAsgTeam(group, firstAssignmentTeam) || firstAssignmentTeam.equals("-1")) {
                check = 'Y';
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        check = 'N';

        if (!firstAssignmentTeam.equals("-1")) {
            int indexOfFirstAsgTeam = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                                                        if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(firstAssignmentTeam)) {
                    indexOfFirstAsgTeam = i;
                    break;
                }
            }

            AssignmentTeam firstAsgTeam = AssignmentTeamList.getEntry(indexOfFirstAsgTeam);
            ListInterface<Student> firstAsgTeamStudentsList = firstAsgTeam.getTeamMember();
                                                ListInterface<AssignmentTeam>  suitableTeam  =
retrieveSuitableMergeTeam(firstAsgTeam);

            System.out.println("Suitable Team For Merging:\n");
            if (!suitableTeam.isEmpty()) {
                System.out.println(suitableTeam);
                    System.out.println("Enter assignment team ID you want to merge with first team
selected");
            } else {
                System.out.println("No suitable team to merge.");
```

```
            System.out.print("\nPress enter to continue...");
            scanner.nextLine();
            check = 'Y';
            secondAssignmentTeam = "-1";
        }

        while (check == 'N') {
                                                                secondAssignmentTeam =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
                                if (checkAsgTeam(suitableTeam, secondAssignmentTeam) ||
secondAssignmentTeam.equals("-1")) {
                check = 'Y';
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        check = 'N';
        if (!secondAssignmentTeam.equals("-1")) {
            int indexOfSecondAsgTeam = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries(); i++) {
                                                                                    if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(secondAssignmentTeam)) {
                    indexOfSecondAsgTeam = i;
                    break;
                }
            }
                                                    AssignmentTeam secondAsgTeam =
AssignmentTeamList.getEntry(indexOfSecondAsgTeam);
                                    ListInterface<Student> secondAsgTeamStudentsList =
secondAsgTeam.getTeamMember();
                    firstAsgTeamStudentsList = mergeAssignmentTeam(firstAsgTeamStudentsList,
secondAsgTeamStudentsList);
                            firstAsgTeam.setNumOfMember(firstAsgTeam.getNumOfMember() +
secondAsgTeam.getNumOfMember());
            AssignmentTeamList.replace(indexOfFirstAsgTeam, firstAsgTeam);
            AssignmentTeamList.remove(indexOfSecondAsgTeam);

            AssignmentTeamDAO.saveToFile(AssignmentTeamList);
            System.out.println("Assignment Group Merge Successfully!");
            displayAssignmentTeam(group);
            System.out.print("\nPress enter to continue...");
            scanner.nextLine();
        }
```

```
        }
    }

/* List Assignment Team For A TutorialGroup */
    public void listAssignmentTeamOfGroup() {
        char check = 'N';
        String programmeCode;
        String tutorialGroupID;

        do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

        displayAssignmentTeam(group);
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();

    }
```

```
/* List Students Under An Assignment Team */
   public void listStudentOfAsgTeam() {
      char check = 'N';
      String assignmentTeam = "";
      String programmeCode;
      String tutorialGroupID;

      do {
         courseManagement.displayAllProgramme(null);
         programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
         if (!courseManagement.findProgramme(programmeCode)) {
            System.out.println("Invalid programme!");
         } else {
            check = 'Y';
         }
         System.out.println();
      } while (check == 'N');

      check = 'N';

      do {
         tutorialGroupManagement.displayProgrammeGroup(programmeCode);
         tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
         if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
            System.out.println("Invalid Tutorial Group ID!");

         } else {
            check = 'Y';

         }
         System.out.println();
      } while (check == 'N');

      check = 'N';

      int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
      TutorialGroup group = TutorialGroupList.getEntry(groupIndex);

      displayAssignmentTeam(group);

      while (check == 'N') {
                                                                  assignmentTeam        =
AssignmentTeamManagementUI.inputAssignmentTeamID().toUpperCase();
         if (checkAsgTeam(group, assignmentTeam) || assignmentTeam.equals("-1")) {
            check = 'Y';
```

```
            } else {
                System.out.println("Invalid assignment team ID!");

            }
            System.out.println();
        }

        if (!assignmentTeam.equals("-1")) {
            int indexToViewStudent = -1;
            for (int i = 1; i < AssignmentTeamList.getNumberOfEntries() ; i++) {
                                                                                    if
(AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(assignmentTeam)) {
                    indexToViewStudent = i;
                    break;
                }
            }

                                                AssignmentTeam      viewStudentAsgTeam      =
AssignmentTeamList.getEntry(indexToViewStudent);
            displayStudentForAsgTeam(viewStudentAsgTeam);
            System.out.print("\nPress enter to continue...");
            scanner.nextLine();

        }

    }

/* Summary Report */
    public void Report() {
        int choice = -1;
        do {
            choice = AssignmentTeamManagementUI.reportChoice();
            switch (choice) {
                case 0:
                    MessageUI.displayExitMessage();
                    break;
                case 1:
                    displayProgrammeAsgTeamReport();
                    break;
                case 2:
                    displayTutorialGroupAsgTeamReport();
                    break;
            }
        } while (choice != 0);
    }
```

```
/* Report Of Assignment Team For Each Programme */
   public String ProgrammeReport() {
      sortProgrammeForAsgTeam();
      ListInterface<String> outputStr = new ArrayList<>();
      ListInterface<AssignmentTeam> TEMPAssignmentTeam = new ArrayList<>();
      ListInterfacet<String> TEMPProgramme = new ArrayList<>();
      ListInterface<TutorialGroup> TEMPTutorialGroup = new ArrayList<>();
      ListInterface<Integer> programmeAssignmentTeamCount = new ArrayList<>();
      ListInterface<Integer> programmeTutorialGroupCount = new ArrayList<>();
      ListInterface<Integer> maxIndex = new ArrayList<>();
      ListInterface<Integer> minIndex = new ArrayList<>();
      int max = 0;
      int min = 999;

      for (int i = 1; i <= programmeSortedList.getNumberOfEntries(); i++) {
         TEMPProgramme.add(programmeSortedList.getEntry(i).getProgrammeCode());
      }

      for (int i = 1; i <= TEMPProgramme.getNumberOfEntries(); i++) {
         int count = 0;
         for (int j = 1; j <= TutorialGroupList.getNumberOfEntries(); j++) {
                                                                                  if
(TEMPProgramme.getEntry(i).equals(TutorialGroupList.getEntry(j).getProgrammeCode())) {
               TEMPTutorialGroup.add(TutorialGroupList.getEntry(j));
               count++;
            }
         }
         programmeTutorialGroupCount.add(count);
      }

      for (int i = 1; i <= TEMPTutorialGroup.getNumberOfEntries(); i++) {
         for (int j = 1; j <= AssignmentTeamList.getNumberOfEntries(); j++) {
                                                                                  if
(TEMPTutorialGroup.getEntry(i).equals(AssignmentTeamList.getEntry(j).getTutorialGroup())) {
               TEMPAssignmentTeam.add(AssignmentTeamList.getEntry(j));
            }
         }

      }

      for (int i = 1; i <= TEMPProgramme.getNumberOfEntries(); i++) {
         int count = 0;
         for (int j = 1; j <= TEMPAssignmentTeam.getNumberOfEntries(); j++) {
                                                                                  if
(TEMPAssignmentTeam.getEntry(j).getTutorialGroup().getProgrammeCode().equals(TEMPPro
gramme.getEntry(i))) {
```

```
                count++;
            }
        }
        programmeAssignmentTeamCount.add(count);
    }

    for (int i = 1; i <= programmeSortedList.getNumberOfEntries(); i++) {
        if (i >= 10) {
                    outputStr.add(i + ". " + programmeSortedList.getEntry(i) + " ".repeat(10) +
programmeTutorialGroupCount.getEntry(i) + " ".repeat(20) +
programmeAssignmentTeamCount.getEntry(i) + "\n");
        } else {
                outputStr.add(" " + i + ". " + programmeSortedList.getEntry(i) + " ".repeat(10) +
programmeTutorialGroupCount.getEntry(i) + " ".repeat(20) +
programmeAssignmentTeamCount.getEntry(i) + "\n");
        }
    }

    outputStr.add("-".repeat(150)
            + "\nTotal " + TEMPProgramme.getNumberOfEntries() + " Programme(s)");

    for (int i = 1; i <= programmeTutorialGroupCount.getNumberOfEntries(); i++) {
        if (programmeAssignmentTeamCount.getEntry(i) != 0) {
            if (programmeAssignmentTeamCount.getEntry(i) > max) {
                max = programmeAssignmentTeamCount.getEntry(i);
                maxIndex.clear();
                maxIndex.add(i);
            } else if (programmeAssignmentTeamCount.getEntry(i) == max) {
                maxIndex.add(i);
            }
            if (programmeAssignmentTeamCount.getEntry(i) < min) {
                min = programmeAssignmentTeamCount.getEntry(i);
                minIndex.clear();
                minIndex.add(i);
            } else if (programmeAssignmentTeamCount.getEntry(i) == min) {
                minIndex.add(i);
            }
        }
    }

    outputStr.add("-".repeat(150) + "\n" + "HIGHEST Number OF Assignment Teams: ");

    for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                        outputStr.add("-> [" + max + " Assignment Team] " + "<" +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(maxIndex.getEntry(i)).getProgrammeName());
```

```
        }

        outputStr.add("\nLOWEST Number OF Assignment Teams:");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                        outputStr.add("-> [" + min + " Assignment Team] " + "<" +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeCode() + "> " +
programmeSortedList.getEntry(minIndex.getEntry(i)).getProgrammeName());
        }

         outputStr.add("\n[NOTE: 0 Assignment Team IS NOT COUNTED]\n" + "-".repeat(150) +
"\n"
                    + " ".repeat(60) + "END OF PROGRAMME (Assignment Team) SUMMARY
REPORT\n"
                + "=".repeat(150));

        return outputStr.toString();
    }

/* Report Of Assignment Team Of A Tutorial Group */
    public String TutorialGroupReport() {
        char check = 'N';
        String programmeCode;
        String tutorialGroupID;
        ListInterface<String> outputStr = new ArrayList<>();
        ListInterface<AssignmentTeam> assignmentTeam = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int max = 0;
        int min = 999;

        do {
            courseManagement.displayAllProgramme(null);
            programmeCode = courseManagementUI.inputProgrammeCode().toUpperCase();
            if (!courseManagement.findProgramme(programmeCode)) {
                System.out.println("Invalid programme!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check == 'N');

        check = 'N';

        do {
            tutorialGroupManagement.displayProgrammeGroup(programmeCode);
```

```
            tutorialGroupID = tutorialGroupManagementUI.inputGroupID().toUpperCase();
            if (tutorialGroupManagement.findGroupIndex(tutorialGroupID) == -1) {
                System.out.println("Invalid Tutorial Group ID!");

            } else {
                check = 'Y';

            }
            System.out.println();
        } while (check == 'N');

        int groupIndex = tutorialGroupManagement.findGroupIndex(tutorialGroupID);
        TutorialGroup group = TutorialGroupList.getEntry(groupIndex);
        int count = 1;
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {

            if (AssignmentTeamList.getEntry(i).getTutorialGroup().equals(group)) {
                if (count >= 10) {
                                                                    outputStr.add(count + ". " +
AssignmentTeamList.getEntry(i).getAssignmentTeamID()        +        "        ".repeat(30)        +
AssignmentTeamList.getEntry(i).getAssignmentTeamName() + " ".repeat(32)
                                + AssignmentTeamList.getEntry(i).getNumOfMember() + " ".repeat(19) +
(AssignmentTeamList.getEntry(i).getAssignmentTeamSize()                                                -
AssignmentTeamList.getEntry(i).getNumOfMember()) + "\n");
                    assignmentTeam.add(AssignmentTeamList.getEntry(i));

                } else {
                                                            outputStr.add("  " + count + ". " +
AssignmentTeamList.getEntry(i).getAssignmentTeamID()        +        "        ".repeat(30)        +
AssignmentTeamList.getEntry(i).getAssignmentTeamName() + " ".repeat(32)
                                + AssignmentTeamList.getEntry(i).getNumOfMember() + " ".repeat(19) +
(AssignmentTeamList.getEntry(i).getAssignmentTeamSize()                                                -
AssignmentTeamList.getEntry(i).getNumOfMember()) + "\n");
                    assignmentTeam.add(AssignmentTeamList.getEntry(i));
                }
                count++;
            }

        }

        outputStr.add("-".repeat(150)
                + "\nTotal " + (count - 1) + " Assignment Team(s)");

        for (int i = 1; i <= assignmentTeam.getNumberOfEntries(); i++) {
            if (assignmentTeam.getEntry(i).getNumOfMember() != 0) {
                if (assignmentTeam.getEntry(i).getNumOfMember() > max) {
```

```
                max = assignmentTeam.getEntry(i).getNumOfMember();
                maxIndex.clear();
                maxIndex.add(i);
            } else if (assignmentTeam.getEntry(i).getNumOfMember() == max) {
                maxIndex.add(i);
            }
            if (assignmentTeam.getEntry(i).getNumOfMember() < min) {
                min = assignmentTeam.getEntry(i).getNumOfMember();
                minIndex.clear();
                minIndex.add(i);
            } else if (assignmentTeam.getEntry(i).getNumOfMember() == min) {
                minIndex.add(i);
            }
        }
    }

    outputStr.add("-".repeat(150) + "\n" + "HIGHEST Number OF Members: ");

    check = 'N';
    if (max == 0) {
        outputStr.add("None");
    } else {
        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + max + " Member(s)] " + "<" +
assignmentTeam.getEntry(maxIndex.getEntry(i)).getAssignmentTeamID() + "> ");
        }
    }

    outputStr.add("\nLOWEST Number OF Members:");

    if (min == 999) {
        outputStr.add("None");
    } else {
        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
                                outputStr.add("-> [" + min + " Member(s)] " + "<" +
assignmentTeam.getEntry(maxIndex.getEntry(i)).getAssignmentTeamID() + "> ");
        }
    }

    outputStr.add("\n[NOTE: 0 Member IS NOT COUNTED]" + "\n" + "-".repeat(150));

    outputStr.add("\nFull Of Member:");

    check = 'N';

    for (int i = 1; i <= assignmentTeam.getNumberOfEntries(); i++) {
```

```
                                    if (assignmentTeam.getEntry(i).getAssignmentTeamSize() ==
assignmentTeam.getEntry(i).getNumOfMember()) {
                outputStr.add("-> [" + assignmentTeam.getEntry(i).getNumOfMember() + " Member]
" + "<" + assignmentTeam.getEntry(i).getAssignmentTeamID() + "> ");
                check = 'Y';
            }
        }

        if (check == 'N') {
            outputStr.add("None");
        }
        check = 'N';

        outputStr.add("-".repeat(150) + "\n" + "Empty Assignment Team: ");
        for (int i = 1; i <= assignmentTeam.getNumberOfEntries(); i++) {
            if (assignmentTeam.getEntry(i).getNumOfMember() == 0) {
                outputStr.add("-> [" + assignmentTeam.getEntry(i).getNumOfMember() + " Member]
" + "<" + assignmentTeam.getEntry(i).getAssignmentTeamID() + "> ");
                check = 'Y';
            }
        }

        if (check == 'N') {
            outputStr.add("None");
        }

        outputStr.add("\n" + "-".repeat(150) + "\n"
                    + " ".repeat(60) + "END OF Tutorial Group( (Assignment Team) SUMMARY
REPORT\n"
                + "=".repeat(150));

        return outputStr.toString();
    }

    /* Check Assignment Team Exists in The Group */
    public boolean checkAsgTeam(TutorialGroup group, String asgTeamID) {
        for (int i = 0; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                        if (AssignmentTeamList.getEntry(i) != null &&
AssignmentTeamList.getEntry(i).getAssignmentTeamID().equals(asgTeamID) &&
AssignmentTeamList.getEntry(i).getTutorialGroup().equals(group)) {
                return true;
            }
        }
        return false;
    }
```

```
   /* Validate The Target Assignment Team ID is In the Assignment Team List*/
            public boolean checkAsgTeam(ListInterface<AssignmentTeam> asgTeam, String
targetAsgTeamID) {
      for (int i = 1; i <= asgTeam.getNumberOfEntries(); i++) {
                                                  if (targetAsgTeamID != null &&
asgTeam.getEntry(i).getAssignmentTeamID().equals(targetAsgTeamID)) {
            return true;
         }
      }
      return false;
   }

   /* Check Student Already Assign In A Assignment Team */
   public boolean checkStudentAsgTeamStatus(AssignmentTeam asgTeam, Student student) {
      for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
                                                  ListInterface<Student> teamMemberList =
AssignmentTeamList.getEntry(i).getTeamMember();
         if (teamMemberList != null && teamMemberList.contains(student)) {
            return true;
         }
      }
      return false;
   }

   /* Check Student Already In This Team */
   public boolean checkStudentExistsInTeam(AssignmentTeam asgTeam, Student student) {
      ListInterface<Student> teamMemberList = asgTeam.getTeamMember();
      return teamMemberList.contains(student);
   }

   /* Check Student Assign In This Tutorial Group */
   public boolean checkStudentExists(TutorialGroup group, String studentID) {
      for (int i = 0; i < studentList.getNumberOfEntries() + 1; i++) {
                                                  if (studentList.getEntry(i) != null &&
group.getGroupStudentsList().contains(studentList.getEntry(i))) {
            return true;
         }
      }
      return false;
   }

   /* Find Student Current Assignment Team */
   public String findStudentCurrentTeam(AssignmentTeam asgTeam, Student student) {
      for (int i = 0; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                          if (AssignmentTeamList.getEntry(i) != null &&
AssignmentTeamList.getEntry(i).getTeamMember().contains(student)) {
```

```
            if (AssignmentTeamList.getEntry(i).equals(asgTeam)) {
                return "This Group";
            }
                                                    return ":\nTutorial Group: " +
AssignmentTeamList.getEntry(i).getTutorialGroup().getGroupID() + " \nAssignment Team: " +
AssignmentTeamList.getEntry(i).getAssignmentTeamID() + "\n";
        }
    }
    return null;
}

/* Get The List Of Assignment Team That Can Merge With Target Team */
        public ListInterface<AssignmentTeam> retrieveSuitableMergeTeam(AssignmentTeam
asgTeam) {
    ListInterface<AssignmentTeam> suitableTeam = new ArrayList<>();
    for (int i = 0; i < AssignmentTeamList.getNumberOfEntries() + 1; i++) {
                                        if (AssignmentTeamList.getEntry(i) != null &&
AssignmentTeamList.getEntry(i).getTutorialGroup().equals(asgTeam.getTutorialGroup()) &&
AssignmentTeamList.getEntry(i).getNumOfMember() + asgTeam.getNumOfMember() <=
asgTeam.getAssignmentTeamSize() && !AssignmentTeamList.getEntry(i).equals(asgTeam)) {
            suitableTeam.add(AssignmentTeamList.getEntry(i));
        }
    }
    return suitableTeam;
}

/* Merge The Two Assignment Team*/
    public ListInterface<Student> mergeAssignmentTeam(ListInterface<Student> asgTeamA,
ListInterface<Student> asgTeamB) {
    for (int i = 0; i < asgTeamB.getNumberOfEntries() + 1; i++) {
        if (asgTeamB.getEntry(i) != null) {
            asgTeamA.add(asgTeamB.getEntry(i));
        }
    }
    return asgTeamA;
}

/* Get The Student Object By Its ID */
public Student retrieveStudent(String studentID) {
    if (studentID == null || studentID.isEmpty()) {
        return null;
    }

    for (int i = 0; i < studentList.getNumberOfEntries() + 1; i++) {
                                            if (studentList.getEntry(i) != null &&
studentList.getEntry(i).getStudentID().equals(studentID)) {
```

```
            Student student = studentList.getEntry(i);
            return student;
        }
    }
    return null;
}

/* Get The Number Of Assignment Team For A Group */
public int getTutorialGroupAsgTeamNum(TutorialGroup group) {
    int count = 0;

    for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
        AssignmentTeam team = AssignmentTeamList.getEntry(i);

        if (team.getTutorialGroup().equals(group)) {
            count++;
        }
    }

    return count;
}

/* Get String OF Assignment Team Member */
public String getAllTeamMember(AssignmentTeam asgTeam) {
    String outputStr = "";

    if (asgTeam == null) {
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
            outputStr += AssignmentTeamList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
            AssignmentTeam team = AssignmentTeamList.getEntry(i);
            if (team.getAssignmentTeamID().equals(asgTeam.getAssignmentTeamID())) {
                outputStr += team.getTeamMember() + "\n";
            }
        }
    }
    if (outputStr.isEmpty()) {
        return "There are no student in this assignment team\n";
    }
    return outputStr;

}

/* Get String Of Assignment Team */
```

```
public String getAllAssignmentTeam(TutorialGroup group) {
    String outputStr = "";

    if (group == null) {
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
            outputStr += AssignmentTeamList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= AssignmentTeamList.getNumberOfEntries(); i++) {
            AssignmentTeam team = AssignmentTeamList.getEntry(i);
            if (team.getTutorialGroup().equals(group)) {
                outputStr += team + "\n";
            }
        }
    }
    if (outputStr.isEmpty()) {
        return "There are no assignment team in this group\n";
    }
    return outputStr;

}

/* Get String Of Student In The Group */
public String getStudentInTutorialGroup(TutorialGroup group) {
    String outputStr = "";

    for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
        Student student = studentList.getEntry(i);
        if (group.getGroupStudentsList().contains(student)) {
            outputStr += student + "\n";
        }

    }
    if (outputStr.isEmpty()) {
        return "There are no student in this Tutorial Group\n";
    }
    return outputStr;

}

/* Sort The Programme Code */
public void sortProgrammeForAsgTeam() {
    programmeSortedList.clear();
    for (int i = 1; i <= ProgrammeList.getNumberOfEntries(); i++) {
        programmeSortedList.add(ProgrammeList.getEntry(i));
    }
```

```
        String criteria = null;
        String sortOrder = null;
        int sortType;
        int sortDetail = courseManagementUI.getProgrammeSortDetailsChoice();
        switch (sortDetail) {
            case 0:
                MessageUI.displayExitMessage();
                break;
            case 1:
                criteria = "ProgrammeCode";
                sortType = courseManagementUI.inputSortType();
                switch (sortType) {
                    case 1:
                        sortOrder = "Ascending";
                                                programmeSortedList.sortByAscending(programme ->
programme.getProgrammeCode());
                        break;
                    case 2:
                        sortOrder = "Descending";
                                                programmeSortedList.sortByDescending(programme ->
programme.getProgrammeCode());
                        break;
                }
                break;
            case 2:
                criteria = "Programme Name";
                sortType = courseManagementUI.inputSortType();
                switch (sortType) {
                    case 1:
                        sortOrder = "Ascending";
                                                programmeSortedList.sortByAscending(programme ->
programme.getProgrammeName());
                        break;
                    case 2:
                        sortOrder = "Descending";
                                                programmeSortedList.sortByDescending(programme ->
programme.getProgrammeName());
                        break;
                }
                break;

            default:
                MessageUI.displayInvalidChoiceMessage();
        }
    }
```

```
    /* Display Assignment Team*/
    public void displayAssignmentTeam(TutorialGroup group) {
        AssignmentTeamManagementUI.listAssignmentTeam(getAllAssignmentTeam(group));
    }

    /* Display Student For Assignment Team */
    public void displayStudentForAsgTeam(AssignmentTeam asgTeam) {

AssignmentTeamManagementUI.listAssignmentTeamStudent(getAllTeamMember(asgTeam));
    }

    /* Display Student For Tutorial Group */
    public void displayStudentForTutorialGroup(TutorialGroup group) {

AssignmentTeamManagementUI.listAssignmentTeamStudent(getStudentInTutorialGroup(group)
);
    }

    /* Display Programme Assignment Team Report */
    public void displayProgrammeAsgTeamReport() {
        AssignmentTeamManagementUI.asgTeamOfProgrammeReport(ProgrammeReport());
    }

    /* Display Tutorial Group Assignment Team Report */
    public void displayTutorialGroupAsgTeamReport() {
        AssignmentTeamManagementUI.asgTeamOfTutorialGroupReport(TutorialGroupReport());
    }

}
```

2. Screenshots

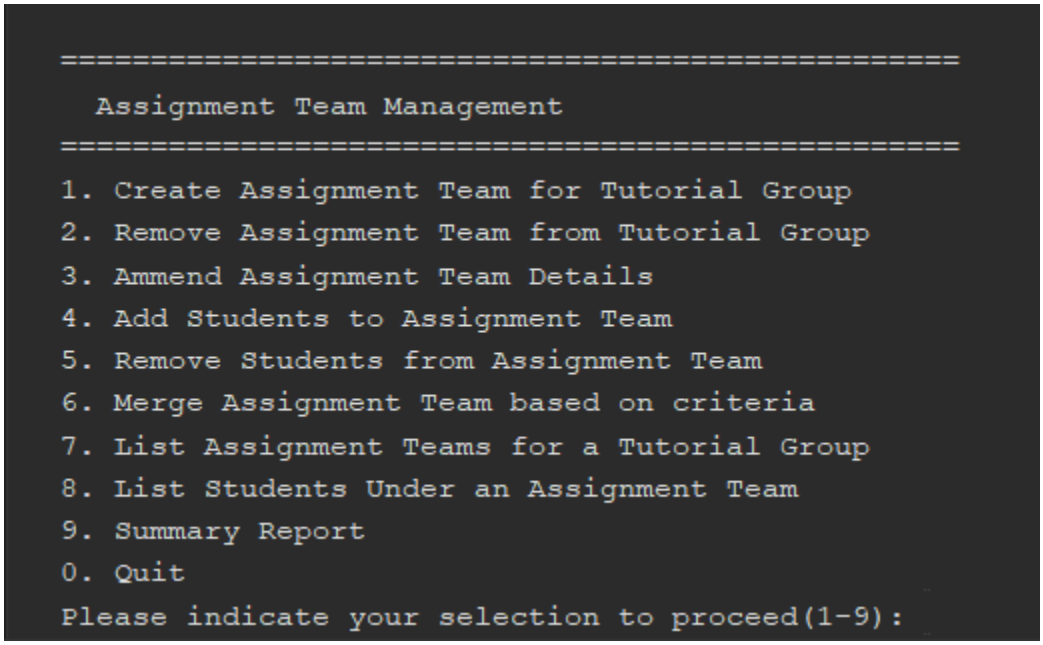


Figure 4.1 Assignment Team Subsystem Menu

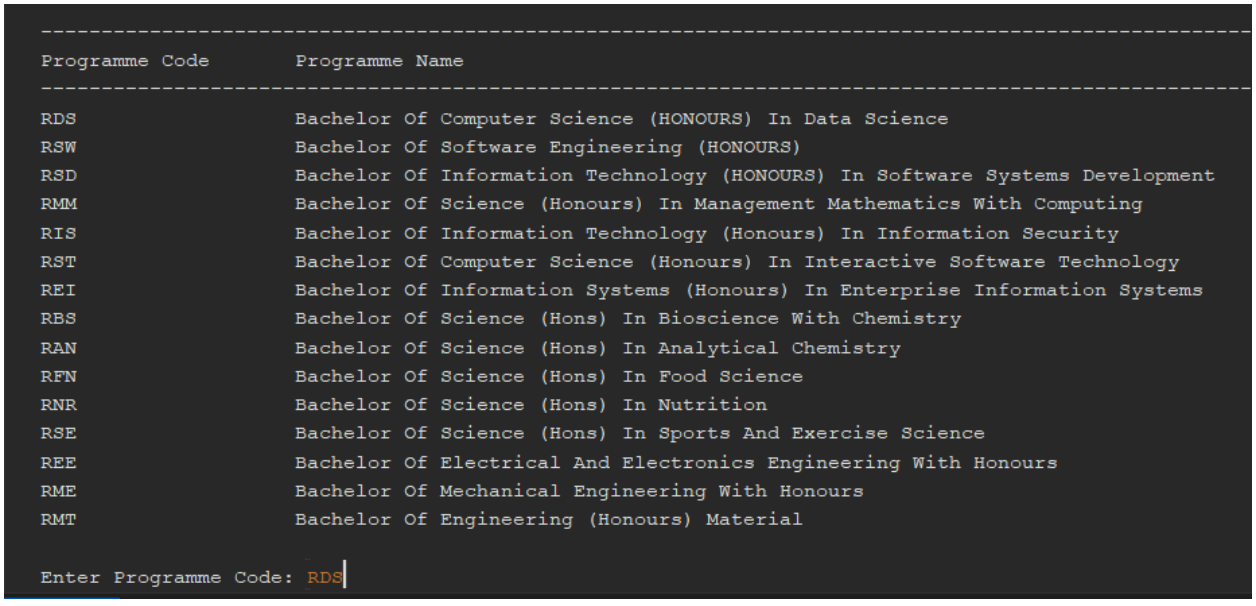


Figure 4.2 Choose The Programme To Add Assignment Team

After select “Create Assignment Team For Tutorial Group”, users will be asked to choose the programme of the tutorial group.

```
List of Tutorial Groups for Programme :

-------------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-------------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter Tutorial Group ID: RDS0001
```

Figure 4.3 Choose The Tutorial Group To Add Assignment Team

After inserting the programmed code, a list of tutorial groups will be shown to users, users are asked to select the tutorial group to create the assignment team.

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader    Number of Member    Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   RDS0001 Assignment Team 1     2              -              0                   RDS0001
RDS0001T2                   RDS0001 Assignment Team 2     4              -              0                   RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -              0                   RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -              0                   RDS0001

Enter Number of Assignment Team to Create (Enter -1 to exit): 2
Enter Assignment Team Size (Enter -1 to exit): 5
```

Figure 4.4 Decide the number of assignment team create and size of assignment team

After selecting the tutorial group, a list of existing assignment teams will be shown and the user is asked to enter the number of assignments to create and specify the size of assignment teams created.

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader    Number of Member    Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   RDS0001 Assignment Team 1     2              -              0                   RDS0001
RDS0001T2                   RDS0001 Assignment Team 2     4              -              0                   RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -              0                   RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -              0                   RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -              0                   RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -              0                   RDS0001

Assignment Team has created successfully!

Press enter to continue...
```

Figure 4.5 Assignment teams has been created by user successfully

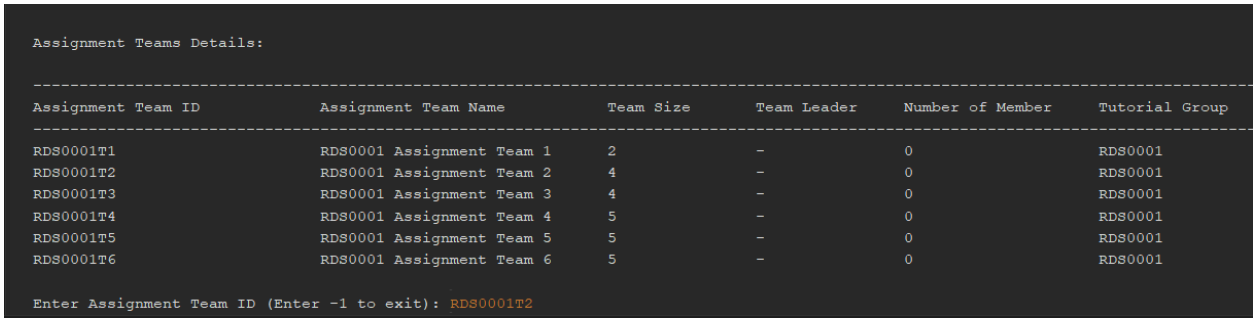


Figure 4.6 User is asked to insert the assignment team ID which the user wants to remove

After choosing the programme and tutorial group, a list of assignment teams will be shown and the user is asked to insert the assignment team ID where he wants to remove from the tutorial group.

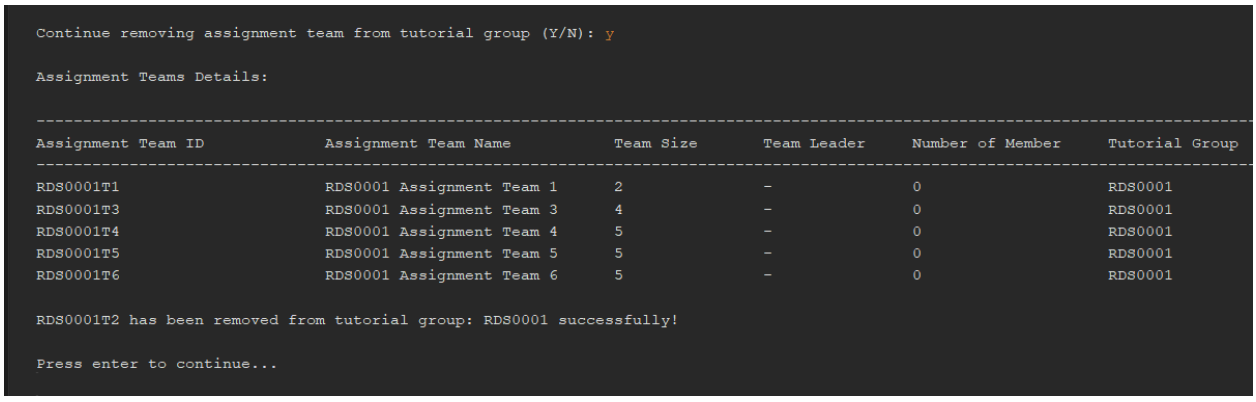


Figure 4.7 Assignment Team has been removed by user successfully

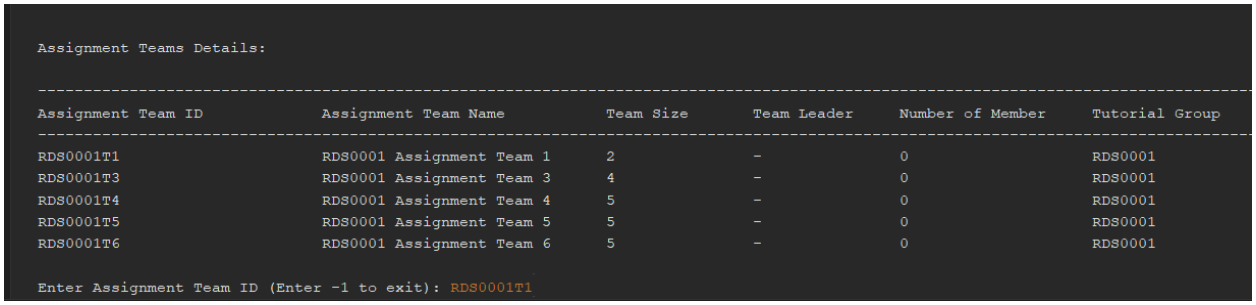


Figure 4.8 User is asked to insert the assignment team ID which the user wants to modify

After choosing the programme and tutorial group, a list of assignment teams will be shown and the user is asked to insert the assignment team ID where he wants to amend the assignment team details.

```
Edit Assignment Team Details Menu
1. Team Name
2. Team Size
3. Team Leader
0. Back
Please select the details you would like to modify(1-4):
```

Figure 4.9 Value that can modify by the user

After selecting the assignment team, a list of menu will be displayed.

```
Old Team Name for RDS0001T1: RDS0001 Assignment Team 1
Enter Assignment Team Name (Enter -1 to exit): DSA documentation report
Team name had been changed to DSA documentation report
```

Figure 4.10 Enter a new team name for selected assignment team

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      2              -               0                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               0                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               0                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001
```

Figure 4.11 Assignment team name has changed by user

```
Old Team Size for RDS0001T1: 2
Enter Assignment Team Size (Enter -1 to exit): 5
Team size had been changed to 5
```

Figure 4.12 Enter a new team size for selected assignment team

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      5              -               0                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               0                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               0                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001
```

Figure 4.12 Assignment team size has changed by user

```
-------------------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      2              -               2                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               0                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               0                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001

Enter Assignment Team ID (Enter -1 to exit): rds0001t1


Edit Assignment Team Details Menu
1. Team Name
2. Team Size
3. Team Leader
0. Back
Please select the details you would like to modify(1-4): 2

Old Team Size for RDS0001T1: 2
Enter Assignment Team Size (Enter -1 to exit): 1
Current Member is Exceeding New Team Size, Please Remove Member First

Press enter to continue...
```

Figure 4.13 Failed to change the assignment team size

Users unable to modify the team size if the value insert is smaller than the current member.

```
-------------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS


Enter Student ID (Enter -1 to exit): S1000
Team Leader had been assigned to Tan Lock Kwan
```

Figure 4.14 User enter the student ID to choose the team leader

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      2              Tan Lock Kwan   2                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               0                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               0                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001
```

Figure 4.15 Team leader has been assigned to student chosen

```
The assignment team is empty

No Student Can Choose to be the Team Leader

Press enter to continue...
```

Figure 4.16 No student can be choose when the assignment team is empty

Team Leader only available to the student in the assignment team.

```
Enter Student ID (Enter -1 to exit): 0
-------------------------------------------------------------------------------------------------------------------
Student ID   Name                  Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan         21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
S1019        Loh Ying              21    Female   lohy-wp21@student.tarc.edu.my   014-02175855    RDS

Enter 0 to view student in the tutorial group

Enter Student ID (Enter -1 to exit):
```

Figure 4.17 Available student to add into assignment team

After choosing the programme and tutorial group, users can view the student list of the tutorial group in the "Add Student To Assignment Team" Module.

```
-------------------------------------------------------------------------------------------------------------------
Student ID   Name                  Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan         21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
S1019        Loh Ying              21    Female   lohy-wp21@student.tarc.edu.my   014-02175855    RDS

Enter 0 to view student in the tutorial group

Enter Student ID (Enter -1 to exit): S1000
Students added to assignment team successfully!
-------------------------------------------------------------------------------------------------------------------
Student ID   Name                  Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan         21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS



Enter 0 to view student in the tutorial group

Enter Student ID (Enter -1 to exit): S1015
Students added to assignment team successfully!
-------------------------------------------------------------------------------------------------------------------
Student ID   Name                  Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan         21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
```

Figure 4.18 Enter Student ID to add to the assignment Team

```
Enter Assignment Team ID (Enter -1 to exit): rds0001t1

-------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS
```

Figure 4.19 Student add to the assignment team successfully

```
Enter Student ID (Enter -1 to exit): S1000
Cannot assign student to this group since student already assign in :
Tutorial Group: RDS0001
Assignment Team: RDS0001T1
```

Figure 4.20 User failed to add student which is already in other group

```
Students added to assignment team successfully!
-------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS


Cannot add more student to the group because the group is full

Press enter to continue...
```

Figure 4.21 User failed to add student when the team is full

```
Enter Assignment Team ID (Enter -1 to exit): rds0001t1

-------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1000        Tan Lock Kwan           21    Female   tanlk-wp21@student.tarc.edu.my  012-11112222    RDS
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS


Enter Student ID (Enter -1 to exit): s1000
```

Figure 4.22 Enter student which the user wants to remove from assignment team

After choosing the programme, tutorial group and assignment teams, students in the assignment team will be shown in the “Remove Student From Assignment Team” module.

```
Enter Student ID (Enter -1 to exit): s1000
Students remove from assignment team successfully!
-------------------------------------------------------------------------------------------------------------------------
Student ID   Name                    Age   Gender   E-mail                          Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------------
S1015        Tan Yu                  21    Female   tany-wp21@student.tarc.edu.my   018-4562585     RDS


Enter Student ID (Enter -1 to exit):
```

Figure 4.23 Student remove from the assignment team successfully

```
-------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      2              -               1                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               0                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               0                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001
```

Figure 4.24 The team leader become null again

If the removed student is the team leader, the team leader will become null after removing him

```
-------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size      Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report      2              -               1                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3     4              -               2                    RDS0001
RDS0001T4                   RDS0001 Assignment Team 4     5              -               1                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001

Enter assignment team ID you want to merge.

Enter Assignment Team ID (Enter -1 to exit): rds0001t1
```

Figure 4.25 Enter the assignment team to merge

```
Suitable Team For Merging:

RDS0001T4                   RDS0001 Assignment Team 4     5              -               1                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5     5              -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6     5              -               0                    RDS0001

Enter assignment team ID you want to merge with first team selected
Enter Assignment Team ID (Enter -1 to exit): rds0001t4
```

Figure 4.26 Show the suitable team suitable for merging and prompt user enter the team ID

Since RDS0001T1 team size is two and that has one member already, RDS0001T3 will not be suitable for merging.

```
Assignment Group Merge Successfully!

Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size     Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report     2             -               2                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3    4             -               2                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5    5             -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6    5             -               0                    RDS0001
```

Figure 4.27 Assignment teams has merge successfully

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size     Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report     2             -               2                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3    4             -               2                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5    5             -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6    5             -               0                    RDS0001
```

Figure 4.28 Assignment Teams details for a tutorial group

After choosing the programme and tutorial group, assignment teams details will be displayed in the “List Assignment Teams for a Tutorial Group” module.

```
Assignment Teams Details:

-------------------------------------------------------------------------------------------------------------------------------
Assignment Team ID          Assignment Team Name          Team Size     Team Leader     Number of Member     Tutorial Group
-------------------------------------------------------------------------------------------------------------------------------
RDS0001T1                   DSA documentation report     2             -               2                    RDS0001
RDS0001T3                   RDS0001 Assignment Team 3    4             -               2                    RDS0001
RDS0001T5                   RDS0001 Assignment Team 5    5             -               0                    RDS0001
RDS0001T6                   RDS0001 Assignment Team 6    5             -               0                    RDS0001

Enter Assignment Team ID (Enter -1 to exit): rds0001t1

-------------------------------------------------------------------------------------------------------------------
Student ID   Name              Age   Gender   E-mail                        Phone Number    Programme
-------------------------------------------------------------------------------------------------------------------
S1015        Tan Yu            21    Female   tany-wp21@student.tarc.edu.my 018-4562585     RDS
S1002        Goh Boon Xiang    22    Male     gohbx-wp21@student.tarc.edu.my 016-34098124    RDS
```

Figure 4.29 Students details for a assignment teams

After choosing the programme,tutorial group and assignment teams, students' details will be displayed in the “List Students Under an Assignment Team” module.

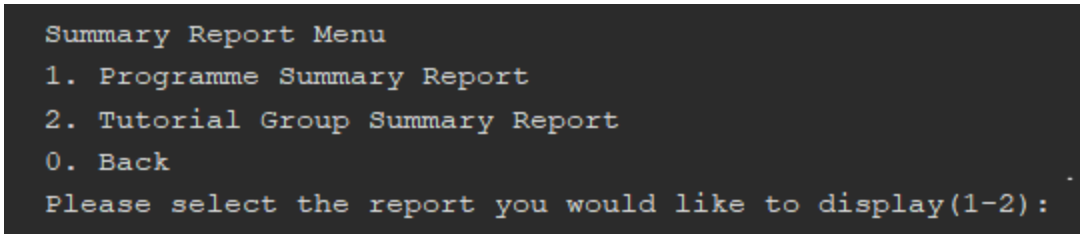


Figure 4.30 Generating Summary Report Menu

User can choose what type of summary report to view

PROGRAMME (Assignment Team) SUMMARY REPORT

Report generated at: Sunday 21-04-2024 01:36AM

	Programme Code	Programme Name	Tutorial Group	Assignment Team
1.	RAN	Bachelor Of Science (Hons) In Analytical Chemistry	2	4
2.	RBS	Bachelor Of Science (Hons) In Bioscience With Chemistry	1	2
3.	RDS	Bachelor Of Computer Science (HONOURS) In Data Science	4	10
4.	REE	Bachelor Of Electrical And Electronics Engineering With Honours	1	2
5.	REI	Bachelor Of Information Systems (Honours) In Enterprise Information Systems	1	2
6.	RFN	Bachelor Of Science (Hons) In Food Science	1	2
7.	RIS	Bachelor Of Information Technology (Honours) In Information Security	1	2
8.	RME	Bachelor Of Mechanical Engineering With Honours	1	2
9.	RMM	Bachelor Of Science (Honours) In Management Mathematics With Computing	1	2
10.	RMT	Bachelor Of Engineering (Honours) Material	3	6
11.	RNR	Bachelor Of Science (Hons) In Nutrition	1	2
12.	RSD	Bachelor Of Information Technology (HONOURS) In Software Systems Development	1	2
13.	RSE	Bachelor Of Science (Hons) In Sports And Exercise Science	1	2
14.	RST	Bachelor Of Computer Science (Honours) In Interactive Software Technology	1	2
15.	RSW	Bachelor Of Software Engineering (HONOURS)	1	2

Total 15 Programme(s)

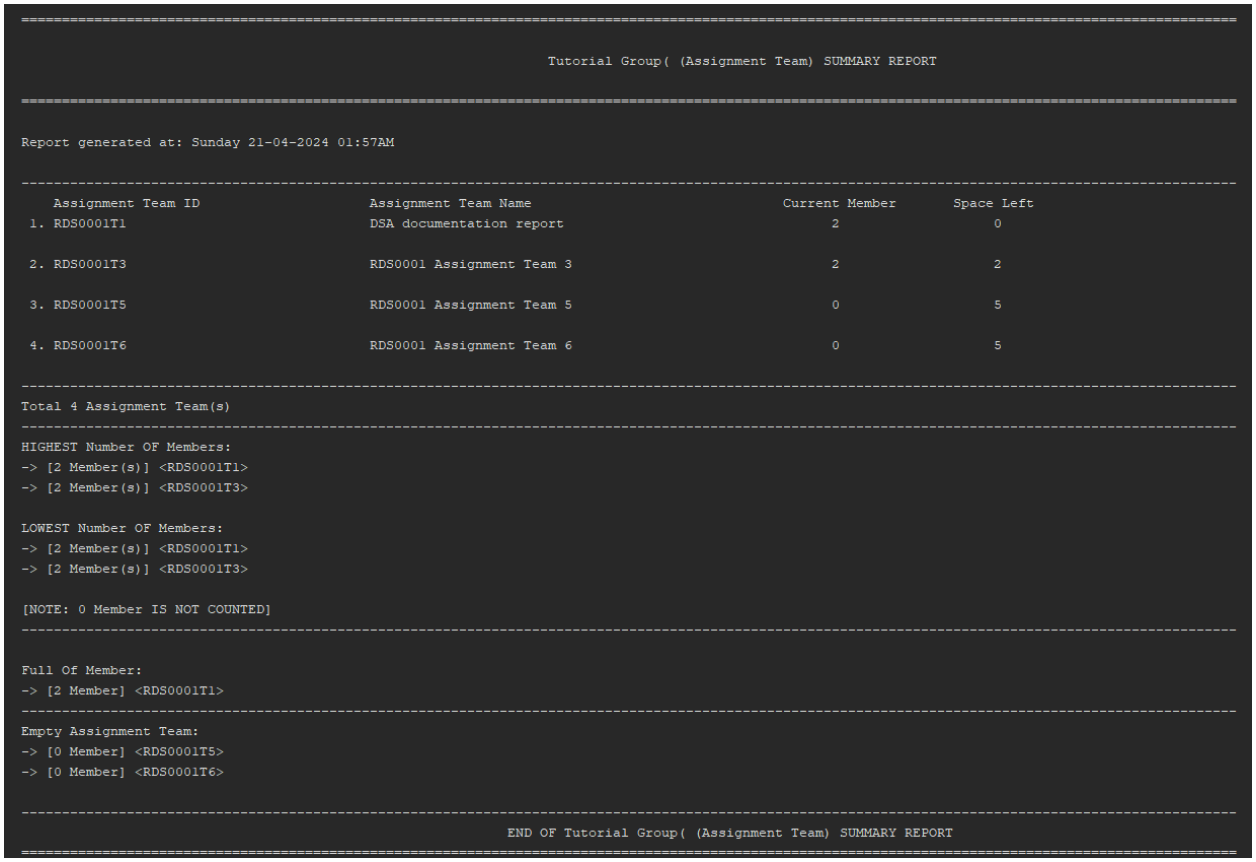
HIGHEST Number OF Assignment Teams:
-> [10 Assignment Team] <RDS> Bachelor Of Computer Science (HONOURS) In Data Science

LOWEST Number OF Assignment Teams:
-> [2 Assignment Team] <RBS> Bachelor Of Science (Hons) In Bioscience With Chemistry
-> [2 Assignment Team] <REE> Bachelor Of Electrical And Electronics Engineering With Honours
-> [2 Assignment Team] <REI> Bachelor Of Information Systems (Honours) In Enterprise Information Systems
-> [2 Assignment Team] <RFN> Bachelor Of Science (Hons) In Food Science
-> [2 Assignment Team] <RIS> Bachelor Of Information Technology (Honours) In Information Security
-> [2 Assignment Team] <RME> Bachelor Of Mechanical Engineering With Honours
-> [2 Assignment Team] <RMM> Bachelor Of Science (Honours) In Management Mathematics With Computing
-> [2 Assignment Team] <RNR> Bachelor Of Science (Hons) In Nutrition
-> [2 Assignment Team] <RSD> Bachelor Of Information Technology (HONOURS) In Software Systems Development
-> [2 Assignment Team] <RSE> Bachelor Of Science (Hons) In Sports And Exercise Science
-> [2 Assignment Team] <RST> Bachelor Of Computer Science (Honours) In Interactive Software Technology
-> [2 Assignment Team] <RSW> Bachelor Of Software Engineering (HONOURS)

[NOTE: 0 Assignment Team IS NOT COUNTED]

END OF PROGRAMME (Assignment Team) SUMMARY REPORT

Figure 4.31 Programme (Assignment Team) Summary Report



```
================================================================================================================================================

                                                    Tutorial Group( (Assignment Team) SUMMARY REPORT

================================================================================================================================================

Report generated at: Sunday 21-04-2024 01:57AM

------------------------------------------------------------------------------------------------------------------------------------------------
    Assignment Team ID                  Assignment Team Name                        Current Member       Space Left
 1. RDS0001T1                           DSA documentation report                         2                 0

 2. RDS0001T3                           RDS0001 Assignment Team 3                        2                 2

 3. RDS0001T5                           RDS0001 Assignment Team 5                        0                 5

 4. RDS0001T6                           RDS0001 Assignment Team 6                        0                 5

------------------------------------------------------------------------------------------------------------------------------------------------
Total 4 Assignment Team(s)
------------------------------------------------------------------------------------------------------------------------------------------------
HIGHEST Number OF Members:
-> [2 Member(s)] <RDS0001T1>
-> [2 Member(s)] <RDS0001T3>

LOWEST Number OF Members:
-> [2 Member(s)] <RDS0001T1>
-> [2 Member(s)] <RDS0001T3>

[NOTE: 0 Member IS NOT COUNTED]
------------------------------------------------------------------------------------------------------------------------------------------------

Full Of Member:
-> [2 Member] <RDS0001T1>
------------------------------------------------------------------------------------------------------------------------------------------------
Empty Assignment Team:
-> [0 Member] <RDS0001T5>
-> [0 Member] <RDS0001T6>

------------------------------------------------------------------------------------------------------------------------------------------------
                                                  END OF Tutorial Group( (Assignment Team) SUMMARY REPORT
================================================================================================================================================
```

Figure 4.32 Tutorial Group (Assignment Team) Summary Report

After choosing the programme and tutorial group, the summary report of the tutorial group chosen will be shown.

Name	Student ID	Prog / Tut.Grp	Signature
Tan Hoong Guan	2213632	RDS2S2G3	*THG*

Subsystem : Teaching Assignment Subsystem

3. **Source codes for Control classes**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package control;

import adt.*;
import boundary.CourseManagementUI;
import dao.*;
import boundary.TeachingAssignmentUI;
import boundary.TutorialGroupManagementUI;
import entity.Course;
import entity.CourseTutor;
import entity.GroupTutorCourse;
import entity.Tutor;
import entity.TutorialGroup;
import entity.courseProgramme;
import java.util.Scanner;
import utility.MessageUI;

/**
 *
 * @author hgtan
 */
public class TeachingAssignment {

    Scanner scanner = new Scanner(System.in);

    private ListInterface<Tutor> TutorList = new ArrayList<>();
```

```
    private ListInterface<Course> CourseList = new ArrayList<>();
    private ListInterface<courseProgramme> courseProgrammeList = new ArrayList<>();
    private ListInterface<TutorialGroup> TutorialGroupList = new ArrayList<>();
    private ListInterface<CourseTutor> CourseTutorList = new ArrayList<>();
    private ListInterface<GroupTutorCourse> GroupTutorCourseList = new ArrayList<>();

    private TutorDAO TutorDAO = new TutorDAO();
    private CourseDAO CourseDAO = new CourseDAO();
    private TutorialGroupDAO TutorialGroupDAO = new TutorialGroupDAO();
    private CourseTutorDAO CourseTutorDAO = new CourseTutorDAO();
    private GroupTutorCourseDAO GroupTutorCourseDAO = new GroupTutorCourseDAO();
    private courseProgrammeDAO CourseProgrammeDAO = new courseProgrammeDAO();

    private TutorInitializer TutorInitializer = new TutorInitializer();
    private CourseInitializer CourseInitializer = new CourseInitializer();
    private CourseTutorInitializer CourseTutorInitializer = new CourseTutorInitializer();
    private GroupTutorCourseInitializer GroupTutorCourseInitializer = new
GroupTutorCourseInitializer();

    private TeachingAssignmentUI TeachingAssignmentUI = new TeachingAssignmentUI();
    private CourseManagementUI CourseManagementUI = new CourseManagementUI();
    private TutorialGroupManagementUI GroupManagementUI = new
TutorialGroupManagementUI();
    private CourseManagement CourseManagement = new CourseManagement();

    public TeachingAssignment() {
        TutorList = TutorDAO.retrieveFromFile();
        CourseList = CourseDAO.retrieveFromFile();
        TutorialGroupList = TutorialGroupDAO.retrieveFromFile();
        courseProgrammeList = CourseProgrammeDAO.retrieveFromFile();
        CourseTutorList = CourseTutorDAO.retrieveFromFile();
        GroupTutorCourseList = GroupTutorCourseDAO.retrieveFromFile();
    }

    public static void main(String[] args) {
        TeachingAssignment teachingAssignment = new TeachingAssignment();
        teachingAssignment.runTeachingAssignment();
    }

    public void runTeachingAssignment() {
```

```
    int choice = 0;
    do {
        choice = TeachingAssignmentUI.getMenuChoice();
        switch (choice) {
            case 0:
                break;
            case 1:
                AssignTutorToCourse();
                break;
            case 2:
                AddTutorialGroupToTutor();
                break;
            case 3:
                AddTutorToTutorialGroupForCourse();
                break;
            case 4:
                SearchCourseUnderTutor();
                break;
            case 5:
                SearchTutorForCourse();
                break;
            case 6:
                ListTutorAndGroupForCourse();
                break;
            case 7:
                ListCourseForEachTutor();
                break;
            case 8:
                FilterTutor();
                break;
            case 9:
                SummaryReport();
                break;
        }
    } while (choice != 0);
}

public void AssignTutorToCourse(){
    String courseCode = "";
    String tutorID = "";
```

```
char check = 'N';
// Display course list and prompt user to select a course to assign tutor.
CourseManagement.displayAllCourse(null);
do {
    courseCode = CourseManagementUI.inputCourseCode();
    if (!CourseManagement.findCourse(courseCode)) {
        System.out.print("Invalid course!");
    } else {
        check = 'Y';
    }
    System.out.println();
} while (check != 'Y');

check = 'N';
// Display tutor list and prompt user to enter a tutor to be assign to the course selected.
displayTutorList(null);
do {
    tutorID = TeachingAssignmentUI.inputTutorID();
    if (!findTutor(tutorID)) {
        System.out.print("Invalid Tutor ID!");
    } else {
        check = 'Y';
    }
    System.out.println();
} while (check != 'Y');

char tutorType = 'N';
do {
    // Prompt user to select the tutor type to be assign to the tutor selected.
    tutorType = TeachingAssignmentUI.inputTutorType();
    // Check if Leturer already exists in the course
    if (tutorType == 'L') {
        for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
            if (CourseTutorList.getEntry(i).getCourseCode().equals(courseCode)) {
                char[] checkArr = CourseTutorList.getEntry(i).getTutorTypeArr();
                for (int j = 0; j < checkArr.length; j++) {
                    if (checkArr[j] == 'L') {
                        tutorType = 'N';
                        System.out.println("Lecturer Already Exists in This Course!");
                        break;
```

```
                    }
                }
            }
        }
    }
} while (tutorType == 'N');

// Check if record already exists.
check = 'N';
int checkPhase = 1;
int position = -1;
char[] tempArr = new char[1];
char[] tutorTypeArr = new char[3];
for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
    if (tutorID.equals(CourseTutorList.getEntry(i).getTutorID())) {
        if (CourseTutorList.getEntry(i).getCourseCode().equals(courseCode)) {
            checkPhase++;
            position = i;
            tempArr = CourseTutorList.getEntry(i).getTutorTypeArr();
            for (int j = 0; j < tempArr.length; j++) {
                tutorTypeArr[j] = tempArr[j];
                if (tutorTypeArr[j] == tutorType) {
                    check = 'Y';
                    break;
                }
            }
        }
    }
}
System.out.println();
if (check == 'Y') {
    System.out.println("Record already exist!");
}

// Add new record and save into file
else {
    if (checkPhase == 1) {
        tutorTypeArr[0] = tutorType;
        CourseTutor newCourseTutor = (CourseTutor)
TeachingAssignmentUI.assignTutorToCourse(tutorID, courseCode, tutorTypeArr);
```

```
            CourseTutorList.add(newCourseTutor);
            CourseTutorDAO.saveToFile(CourseTutorList);
        } else {
            for (int i = 0; i < tutorTypeArr.length; i++) {
                if (tutorTypeArr[i] == '\0') {
                    tutorTypeArr[i] = tutorType;
                    break;
                }
            }
            CourseTutor newCourseTutor = (CourseTutor)
TeachingAssignmentUI.assignTutorToCourse(tutorID, courseCode, tutorTypeArr);
            CourseTutorList.getEntry(position).setTutorTypeArr(tutorTypeArr);
            CourseTutorDAO.saveToFile(CourseTutorList);
        }
    }
    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
  }

  public void AddTutorialGroupToTutor() {
    String tutorID = "";
    String groupID = "";
    char check = 'N';
    // Display Tutor list and prompt user to select a tutor to add Tutorial Group to
    displayTutorList(null);
    do {
      tutorID = TeachingAssignmentUI.inputTutorID();
      if (!findTutor(tutorID)) {
        System.out.print("Invalid Tutor ID!");
      } else {
        check = 'Y';
      }
      System.out.println();
    } while (check != 'Y');

    check = 'N';

    // Display tutor list and prompt user to enter a tutor to be assign to the course selected.
    displayProgrammeGroup(null);
    do {
```

```
            groupID = TeachingAssignmentUI.inputTutorialGroup();
            if (!findTutorialGroup(groupID)) {
                System.out.print("Invalid Group ID!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');

        check = 'N';

        // Check if the record already exists
        for (int i = 1; i <= GroupTutorCourseList.getNumberOfEntries(); i++) {
            if (tutorID.equals(GroupTutorCourseList.getEntry(i).getTutorID())) {
                if (GroupTutorCourseList.getEntry(i).getGroupID().equals(groupID)) {
                    check = 'Y';
                    break;
                }
            }
        }
        System.out.println();
        if (check == 'Y') {
            System.out.println("Record already exist!");
        } else {
            GroupTutorCourse newGroupTutorCourse = (GroupTutorCourse)
TeachingAssignmentUI.addTutorialGroupToTutor(groupID, tutorID, "");
            GroupTutorCourseList.add(newGroupTutorCourse);
            GroupTutorCourseDAO.saveToFile(GroupTutorCourseList);
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void AddTutorToTutorialGroupForCourse() {
        String courseCode = "";
        String groupID = "";
        String tutorID = "";
        char check = 'N';

        // Display course list and prompt user to selsect course
```

```
CourseManagement.displayAllCourse(null);
do {
   courseCode = CourseManagementUI.inputCourseCode();
   if (!CourseManagement.findCourse(courseCode)) {
      System.out.print("Invalid course!");
   } else {
      check = 'Y';
   }
   System.out.println();
} while (check != 'Y');

check = 'N';

// Display tutorial group list under the course
displayProgrammeGroupUnderCourse(courseCode);

// Prompt user to input tutorial group id to add tutor
do {
   groupID = TeachingAssignmentUI.inputTutorialGroup();
   if (!findTutorialGroup(groupID)) {
      System.out.print("Invalid Group ID!");
   }
   else if (!findTutorialGroupUnderCourse(courseCode, groupID)) {
      System.out.print("The Tutorial Group Is Not Under the Course!");
   }
   else {
      check = 'Y';
   }
   System.out.println();
} while (check != 'Y');

check = 'N';

// Display tutor list under the course
displayTutorsUnderCourse(courseCode);

// Prompt user to input tutor id to be added to the tutorial group
do {
   tutorID = TeachingAssignmentUI.inputTutorID();
   if (!findTutor(tutorID)) {
```

```
            System.out.print("Invalid Tutor ID!");
        }
        else if (!findTutorUnderCourse(courseCode, tutorID)) {
            System.out.print("The Tutor Is Not Under The Course!");
        }
        else {
            check = 'Y';
        }
        System.out.println();
    } while (check != 'Y');

    check = 'N';

    // Check if there is an CourseCode empty version of the record exist, if yes, remove it
    for (int i = 1; i <= GroupTutorCourseList.getNumberOfEntries(); i++) {
        if (GroupTutorCourseList.getEntry(i).getGroupID().equals(groupID)) {
            if (GroupTutorCourseList.getEntry(i).getTutorID().equals(tutorID)) {
                if (GroupTutorCourseList.getEntry(i).getCourseCode().equals("")) {
                    GroupTutorCourseList.remove(i);
                    break;
                }
            }
        }
    }

    // Check if the record already exists
    for (int i = 1; i <= GroupTutorCourseList.getNumberOfEntries(); i++) {
        if (GroupTutorCourseList.getEntry(i).getGroupID().equals(groupID)) {
            if (GroupTutorCourseList.getEntry(i).getTutorID().equals(tutorID)) {
                if (GroupTutorCourseList.getEntry(i).getCourseCode().equals(courseCode)) {
                    check = 'Y';
                    break;
                }
            }
        }
    }
    System.out.println();
    if (check == 'Y') {
        System.out.println("Record already exist!");
    } else {
```

```
            GroupTutorCourse newGroupTutorCourse = (GroupTutorCourse)
TeachingAssignmentUI.addTutorialGroupToTutor(groupID, tutorID, courseCode);
            GroupTutorCourseList.add(newGroupTutorCourse);
            GroupTutorCourseDAO.saveToFile(GroupTutorCourseList);
        }
        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void SearchCourseUnderTutor() {
        String tutorID = "";
        char check = 'N';

        // Display tutor list and prompt user to input tutor id
        displayTutorList(null);
        do {
            tutorID = TeachingAssignmentUI.inputTutorID();
            if (!findTutor(tutorID)) {
                System.out.print("Invalid Tutor ID!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');

        // List courses under the tutor
        displayCoursesUnderTutor(tutorID);

        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void SearchTutorForCourse() {
        String courseCode = "";
        String tutorID = "";
        char check = 'N';

        // Display course list and prompt user to input course
        CourseManagement.displayAllCourse(null);
        do {
```

```
            courseCode = CourseManagementUI.inputCourseCode();
            if (!CourseManagement.findCourse(courseCode)) {
                System.out.print("Invalid course!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');

        check = 'N';

        // Prompt user to enter tutor type
        char tutorType = TeachingAssignmentUI.inputTutorType();

        // Display tutor list with the type selected under the course
        displayTutorsUnderCourseWithType(courseCode, tutorType);

        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void ListTutorAndGroupForCourse() {
        String courseCode = "";
        char check = 'N';

        // Display course list and prompt user to input course
        CourseManagement.displayAllCourse(null);
        do {
            courseCode = CourseManagementUI.inputCourseCode();
            if (!CourseManagement.findCourse(courseCode)) {
                System.out.print("Invalid course!");
            } else {
                check = 'Y';
            }
            System.out.println();
        } while (check != 'Y');

        check = 'N';

        // Display tutor and tutorial group for the course
```

```
    displayTutorAndGroupForCourse(courseCode);

    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

public void ListCourseForEachTutor() {
    // Display all tutors and their courses
    displayTutorCourse();

    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

public void FilterTutor() {
    // Display tutor list in full
    displayTutorList(null);

    // Get user criteria choice
    int criteriaChoice = 0;
    criteriaChoice = TeachingAssignmentUI.getCriteriaChoice();

    // Filter and display tutor list based on the criteria
    int age = 0;
    switch (criteriaChoice) {
        case 0:
            MessageUI.displayExitMessage();
            break;
        case 1:
            // filter by age above
            age = TeachingAssignmentUI.inputTutorAgeAbove();
            ageFilter(criteriaChoice, age);
            break;
        case 2:
            // filter by age below
            age = TeachingAssignmentUI.inputTutorAgeBelow();
            ageFilter(criteriaChoice, age);
            break;
        case 3:
            // filter by gender
```

```
            String gender = TeachingAssignmentUI.inputTutorGender();
            genderFilter(gender);
            break;
        }

        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void SummaryReport() {
        int choice = -1;
        do {
            choice = TeachingAssignmentUI.inputReportChoice();
            switch (choice) {
                case 0:
                    MessageUI.displayExitMessage();
                    break;
                case 1:
                    TutorReport();
                    break;
                case 2:
                    CourseReport();
            }
        } while (choice != 0);
    }

    public void TutorReport() {
        ListInterface<Tutor> TutorDetails = new ArrayList<>();
        ListInterface<Integer> CourseCount = new ArrayList<>();
        ListInterface<Integer> TutorialGroupCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();

        int max = 0;
        int min = 999;

        for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
            TutorDetails.add(TutorList.getEntry(i));
        }
```

```
        // Sort list by tutor id
        TutorDetails.sortByAscending(tutor -> tutor.getTutorID());

        // Count the number of courses teached by each tutor
        for (int i = 1; i <= TutorDetails.getNumberOfEntries(); i++) {
            int totalCourse = 0;
            String tutorID = TutorDetails.getEntry(i).getTutorID();
            for (int j = 1; j <= CourseTutorList.getNumberOfEntries(); j++) {
                if (CourseTutorList.getEntry(j).getTutorID().equals(tutorID)) {
                    totalCourse++;
                }
            }
            CourseCount.add(totalCourse);
        }

        // Count the number of tutorial groups handled by each tutor
        for (int i = 1; i <= TutorDetails.getNumberOfEntries(); i++) {
            int totalGroup = 0;
            String tutorID = TutorDetails.getEntry(i).getTutorID();
            for (int j = 1; j <= GroupTutorCourseList.getNumberOfEntries(); j++) {
                if (GroupTutorCourseList.getEntry(j).getTutorID().equals(tutorID)) {
                    String groupID = GroupTutorCourseList.getEntry(j).getGroupID();
                    if (j > 1) {
                        if (!groupID.equals(GroupTutorCourseList.getEntry(j-1).getGroupID())) {
                            totalGroup++;
                        }
                    } else {
                        totalGroup++;
                    }
                }
            }
            TutorialGroupCount.add(totalGroup);
        }

        // Display Report
        TeachingAssignmentUI.TutorReport();
        for (int i = 1; i <= TutorDetails.getNumberOfEntries(); i++) {
            System.out.printf("%2d. %-20s %-78s %4d %24d\n", i,
TutorDetails.getEntry(i).getTutorID(),
                    TutorDetails.getEntry(i).getName(), CourseCount.getEntry(i),
```

```
            TutorialGroupCount.getEntry(i));
        }

        System.out.print("-".repeat(150));
        System.out.print("\nTotal " + TutorDetails.getNumberOfEntries() + " Tutors\n");

        // Calculate min max CourseCount for tutor
        for (int i = 1; i <= CourseCount.getNumberOfEntries(); i++) {
            if (CourseCount.getEntry(i) != 0) {
                if (CourseCount.getEntry(i) > max) {
                    max = CourseCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (CourseCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (CourseCount.getEntry(i) < min) {
                    min = CourseCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (CourseCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("-".repeat(150) + "\n" + "HIGHEST COURSES HANDLED: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Courses] " + "<" +
TutorDetails.getEntry(maxIndex.getEntry(i)).getTutorID() + "> " +
TutorDetails.getEntry(maxIndex.getEntry(i)).getName());
        }

        System.out.print("\n\nLOWEST COURSES HANDLED : ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Courses] " + "<" +
TutorDetails.getEntry(minIndex.getEntry(i)).getTutorID() + "> " +
TutorDetails.getEntry(minIndex.getEntry(i)).getName());
```

```
        }

        System.out.print("\n[NOTE: 0 COURSES IS NOT COUNTED]" + "\n" + "-".repeat(150));

        max = 0;
        min = 999;

        // Calculate min max TutorialGroupCount for tutor
        for (int i = 1; i <= TutorialGroupCount.getNumberOfEntries(); i++) {
           if (TutorialGroupCount.getEntry(i) != 0) {
              if (TutorialGroupCount.getEntry(i) > max) {
                 max = TutorialGroupCount.getEntry(i);
                 maxIndex.clear();
                 maxIndex.add(i);
              } else if (TutorialGroupCount.getEntry(i) == max) {
                 maxIndex.add(i);
              }
              if (TutorialGroupCount.getEntry(i) < min) {
                 min = TutorialGroupCount.getEntry(i);
                 minIndex.clear();
                 minIndex.add(i);
              } else if (TutorialGroupCount.getEntry(i) == min) {
                 minIndex.add(i);
              }
           }
        }

        System.out.print("\n" + "MOST TUTORIAL GROUPS HANDLED TUTOR(S): ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
           System.out.print("\n-> [" + max + " Tutorial Groups] " + "<" +
TutorDetails.getEntry(maxIndex.getEntry(i)).getTutorID() + "> " +
TutorDetails.getEntry(maxIndex.getEntry(i)).getName());
        }

        System.out.print("\n\nLEAST TUTORIAL GROUPS HANDLED TUTOR(S): ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
```

```
            System.out.print("\n-> [" + min + " Tutorial Groups] " + "<" +
TutorDetails.getEntry(minIndex.getEntry(i)).getTutorID() + "> " +
TutorDetails.getEntry(minIndex.getEntry(i)).getName());
        }

        System.out.print(
                "\n[NOTE: 0 TUTORIAL GROUPS IS NOT COUNTED]\n"
                + "-".repeat(150) + "\n"
                + " ".repeat(60) + "END OF TUTOR SUMMARY REPORT\n"
                + "=".repeat(150));

        System.out.print("\nPress enter to continue...");
        scanner.nextLine();
    }

    public void CourseReport() {
        ListInterface<Course> CourseDetailList = new ArrayList<>();
        ListInterface<Integer> TutorCount = new ArrayList<>();
        ListInterface<Integer> LectureCount = new ArrayList<>();
        ListInterface<Integer> TutorialCount = new ArrayList<>();
        ListInterface<Integer> PracticalCount = new ArrayList<>();
        ListInterface<Integer> maxIndex = new ArrayList<>();
        ListInterface<Integer> minIndex = new ArrayList<>();
        int max = 0;
        int min = 999;

        for (int i = 1; i <= CourseList.getNumberOfEntries(); i++) {
            CourseDetailList.add(CourseList.getEntry(i));
        }

        CourseDetailList.sortByAscending(course -> course.getCourseCode());

        // Count the number of tutors for each course
        for (int i = 1; i <= CourseDetailList.getNumberOfEntries(); i++) {
            int totalTutor = 0;
            int LCount = 0;
            int TCount = 0;
            int PCount = 0;
            String courseCode = CourseDetailList.getEntry(i).getCourseCode();
            for (int j = 1; j <= CourseTutorList.getNumberOfEntries(); j++) {
```

```
            if (CourseTutorList.getEntry(j).getCourseCode().equals(courseCode)) {
                totalTutor++;

                // Count the number of L, T, P for each course
                char[] typeArr = CourseTutorList.getEntry(j).getTutorTypeArr();
                for (int k = 0; k < typeArr.length; k++) {
                    switch (typeArr[k]) {
                        case 'L':
                            LCount++;
                            break;
                        case 'T':
                            TCount++;
                            break;
                        case 'P':
                            PCount++;
                            break;
                        default:
                            break;
                    }
                }
            }
        }
        TutorCount.add(totalTutor);
        LectureCount.add(LCount);
        TutorialCount.add(TCount);
        PracticalCount.add(PCount);
    }

    // Display Report
    TeachingAssignmentUI.CourseReport();
    for (int i = 1; i <= CourseDetailList.getNumberOfEntries(); i++) {
        System.out.printf("%2d. %-20s %-78s %6d %15d %5d %5d\n", i,
                CourseDetailList.getEntry(i).getCourseCode(),
                CourseDetailList.getEntry(i).getCourseName(),
                TutorCount.getEntry(i), LectureCount.getEntry(i),
                TutorialCount.getEntry(i), PracticalCount.getEntry(i));
    }

    System.out.print("-".repeat(150));
    System.out.print("\nTotal " + CourseDetailList.getNumberOfEntries() + " Courses\n");
```

```
        // Calculate min max TutorCount for course
        for (int i = 1; i <= TutorCount.getNumberOfEntries(); i++) {
            if (TutorCount.getEntry(i) != 0) {
                if (TutorCount.getEntry(i) > max) {
                    max = TutorCount.getEntry(i);
                    maxIndex.clear();
                    maxIndex.add(i);
                } else if (TutorCount.getEntry(i) == max) {
                    maxIndex.add(i);
                }
                if (TutorCount.getEntry(i) < min) {
                    min = TutorCount.getEntry(i);
                    minIndex.clear();
                    minIndex.add(i);
                } else if (TutorCount.getEntry(i) == min) {
                    minIndex.add(i);
                }
            }
        }

        System.out.print("-".repeat(150) + "\n" + "MOST TUTORS COURSE: ");

        for (int i = 1; i <= maxIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + max + " Tutors] " + "<" +
CourseDetailList.getEntry(maxIndex.getEntry(i)).getCourseCode()+ "> " +
CourseDetailList.getEntry(maxIndex.getEntry(i)).getCourseName());
        }

        System.out.print("\n\nLEAST TUTORS COURSE : ");

        for (int i = 1; i <= minIndex.getNumberOfEntries(); i++) {
            System.out.print("\n-> [" + min + " Tutors] " + "<" +
CourseDetailList.getEntry(minIndex.getEntry(i)).getCourseCode() + "> " +
CourseDetailList.getEntry(minIndex.getEntry(i)).getCourseName());
        }

        System.out.print(
                "\n[NOTE: 0 TUTOR IS NOT COUNTED]\n"
                + "-".repeat(150) + "\n"
```

```
            + " ".repeat(60) + "END OF COURSE TUTOR SUMMARY REPORT\n"
            + "=".repeat(150));

    System.out.print("\nPress enter to continue...");
    scanner.nextLine();
}

public String getAllTutor(String tutorID) {
    String outputStr = "";
    if (tutorID == null) {
        for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
            outputStr += TutorList.getEntry(i) + "\n";

        }
    } else {
        for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
            if (tutorID.equals(TutorList.getEntry(i).getTutorID())) {
                outputStr += TutorList.getEntry(i) + "\n";
            }
        }
    }
    return outputStr;
}

public String getAllProgrammeGroup(String programmeCode) {
    String outputStr = "";

    if (programmeCode == null) {
        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
            outputStr += TutorialGroupList.getEntry(i) + "\n";
        }
    } else {
        for (int i = 1; i <= TutorialGroupList.getNumberOfEntries(); i++) {
         TutorialGroup group = TutorialGroupList.getEntry(i);
         if (group.getProgrammeCode().equals(programmeCode)) {
             outputStr += group + "\n";
            }
        }
    }
    if (outputStr.isEmpty()) {
```

```
        return "There are no group taking this programme!\n";
    }
    return outputStr;

}

public String getAllTutorCourse() {
    String outputStr = "";
    for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
        outputStr += CourseTutorList.getEntry(i) + "\n";
    }
    return outputStr;
}

public String getProgrammeGroupUnderCourse(String courseCode) {
    String outputStr = "";
    for (int i = 1; i <= courseProgrammeList.getNumberOfEntries(); i++) {
        if (courseProgrammeList.getEntry(i).getCourseCode().equals(courseCode)) {
            String programmeCode = courseProgrammeList.getEntry(i).getProgrammeCode();
            for (int j = 1; j <= TutorialGroupList.getNumberOfEntries(); j++) {
                if (TutorialGroupList.getEntry(j).getProgrammeCode().equals(programmeCode)) {
                    outputStr += TutorialGroupList.getEntry(j) + "\n";
                }
            }
        }
    }
    return outputStr;
}

public String getTutorsUnderCourse(String courseCode) {
    String outputStr = "";
    for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
        if (CourseTutorList.getEntry(i).getCourseCode().equals(courseCode)) {
            String tutorID = CourseTutorList.getEntry(i).getTutorID();
            for (int k = 1; k <= TutorList.getNumberOfEntries(); k++) {
                if (TutorList.getEntry(k).getTutorID().equals(tutorID)) {
                    outputStr += TutorList.getEntry(k) + "\n";
                }
            }
        }
```

```
    }
    return outputStr;
}

public String getCoursesUnderTutor(String tutorID) {
    String outputStr = "";
    for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
        if (CourseTutorList.getEntry(i).getTutorID().equals(tutorID)) {
            String courseCode = CourseTutorList.getEntry(i).getCourseCode();
            for (int j = 1; j <= CourseList.getNumberOfEntries(); j++) {
                if (CourseList.getEntry(j).getCourseCode().equals(courseCode)) {
                    outputStr += CourseList.getEntry(j) + "\n";
                }
            }
        }
    }
    return outputStr;
}

public String getTutorsUnderCourseWithType(String courseCode, char tutorType) {
    String outputStr = "";
    for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
        if (CourseTutorList.getEntry(i).getCourseCode().equals(courseCode)) {
            char[] checkArr = CourseTutorList.getEntry(i).getTutorTypeArr();
            for (int j = 0; j < checkArr.length; j++) {
                if (checkArr[j] == tutorType) {
                    String tutorID = CourseTutorList.getEntry(i).getTutorID();
                    for (int k = 1; k <= TutorList.getNumberOfEntries(); k++) {
                        if (TutorList.getEntry(k).getTutorID().equals(tutorID)) {
                            outputStr += TutorList.getEntry(k) + "\n";
                        }
                    }
                }
            }
        }
    }
    return outputStr;
}

public String getTutorAndGroupForCourse(String courseCode) {
```

```
    String outputStr = "";
    for (int i = 1; i <= GroupTutorCourseList.getNumberOfEntries(); i++) {
        if (GroupTutorCourseList.getEntry(i).getCourseCode().equals(courseCode)) {
            outputStr += GroupTutorCourseList.getEntry(i) + "\n";
        }
    }
    return outputStr;
}

public String getTutorForGender(String gender) {
    String outputStr = "";
    for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
        if (TutorList.getEntry(i).getGender().equals(gender)) {
            outputStr += TutorList.getEntry(i) + "\n";
        }
    }
    return outputStr;
}

public String getTutorForAge(int criteriaChoice, int age) {
    String outputStr = "";
    switch (criteriaChoice) {
        case 1:
            for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
                if (TutorList.getEntry(i).getAge() > age) {
                    outputStr += TutorList.getEntry(i) + "\n";
                }
            }
            break;
        case 2:
            for (int i = 1; i <= TutorList.getNumberOfEntries(); i++) {
                if (TutorList.getEntry(i).getAge() < age) {
                    outputStr += TutorList.getEntry(i) + "\n";
                }
            }
            break;
    }
    return outputStr;
}
```

```
public boolean findTutor(String tutorID) {
    for (int i = 0; i < TutorList.getNumberOfEntries() + 1; i++) {
        if (TutorList.getEntry(i) != null) {
            if (TutorList.getEntry(i).getTutorID().equals(tutorID)) {
                return true;
            }
        }
    }
    return false;
}

public boolean findTutorialGroup(String groupID) {
    for (int i = 0; i < TutorialGroupList.getNumberOfEntries() + 1; i++) {
        if (TutorialGroupList.getEntry(i) != null) {
            if (TutorialGroupList.getEntry(i).getGroupID().equals(groupID)) {
                return true;
            }
        }
    }
    return false;
}

public boolean findTutorialGroupUnderCourse(String courseCode, String groupID) {
    for (int i = 1; i <= courseProgrammeList.getNumberOfEntries(); i++) {
        if (courseProgrammeList.getEntry(i).getCourseCode().equals(courseCode)) {
            String programme = courseProgrammeList.getEntry(i).getProgrammeCode();
            for (int j = 1; j <= TutorialGroupList.getNumberOfEntries(); j++) {
                if (TutorialGroupList.getEntry(j).getProgrammeCode().equals(programme)) {
                    if (TutorialGroupList.getEntry(j).getGroupID().equals(groupID)) {
                        return true;
                    }
                }
            }
        }
    }
    return false;
}

public boolean findTutorUnderCourse(String courseCode, String tutorID) {
    for (int i = 1; i <= CourseTutorList.getNumberOfEntries(); i++) {
```

```
            if (CourseTutorList.getEntry(i).getCourseCode().equals(courseCode)) {
                if (CourseTutorList.getEntry(i).getTutorID().equals(tutorID)) {
                    return true;
                }
            }
        }
        return false;
    }

    public void displayTutorList(String tutorID) {
        TeachingAssignmentUI.listAllTutor(getAllTutor(tutorID));
    }

    public void displayProgrammeGroup(String programmeCode) {
        GroupManagementUI.listAllTutorialGroup(getAllProgrammeGroup(programmeCode));
    }

    public void displayProgrammeGroupUnderCourse(String courseCode) {

TeachingAssignmentUI.listProgrammeGroupUnderCourse(getProgrammeGroupUnderCourse(co
urseCode), courseCode);
    }

    public void displayTutorsUnderCourse(String courseCode) {
        TeachingAssignmentUI.listTutorsUnderCourse(getTutorsUnderCourse(courseCode),
courseCode);
    }

    public void displayTutorCourse() {
        TeachingAssignmentUI.listTutorCourse(getAllTutorCourse());
    }

    public void displayCoursesUnderTutor(String tutorID) {
        TeachingAssignmentUI.listCoursesUnderTutor(getCoursesUnderTutor(tutorID), tutorID);
    }

    public void displayTutorsUnderCourseWithType(String courseCode, char tutorType) {

TeachingAssignmentUI.listTutorsUnderCourseWithType(getTutorsUnderCourseWithType(cours
eCode, tutorType), courseCode, tutorType);
```

```
    }

    public void displayTutorAndGroupForCourse(String courseCode) {

TeachingAssignmentUI.listTutorAndGroupForCourse(getTutorAndGroupForCourse(courseCode
), courseCode);
    }

    public void genderFilter(String gender) {
        TeachingAssignmentUI.listAllTutor(getTutorForGender(gender));
    }

    public void ageFilter(int criteriaChoice, int age) {
        TeachingAssignmentUI.listAllTutor(getTutorForAge(criteriaChoice, age));
    }

}
```

4. Screenshots

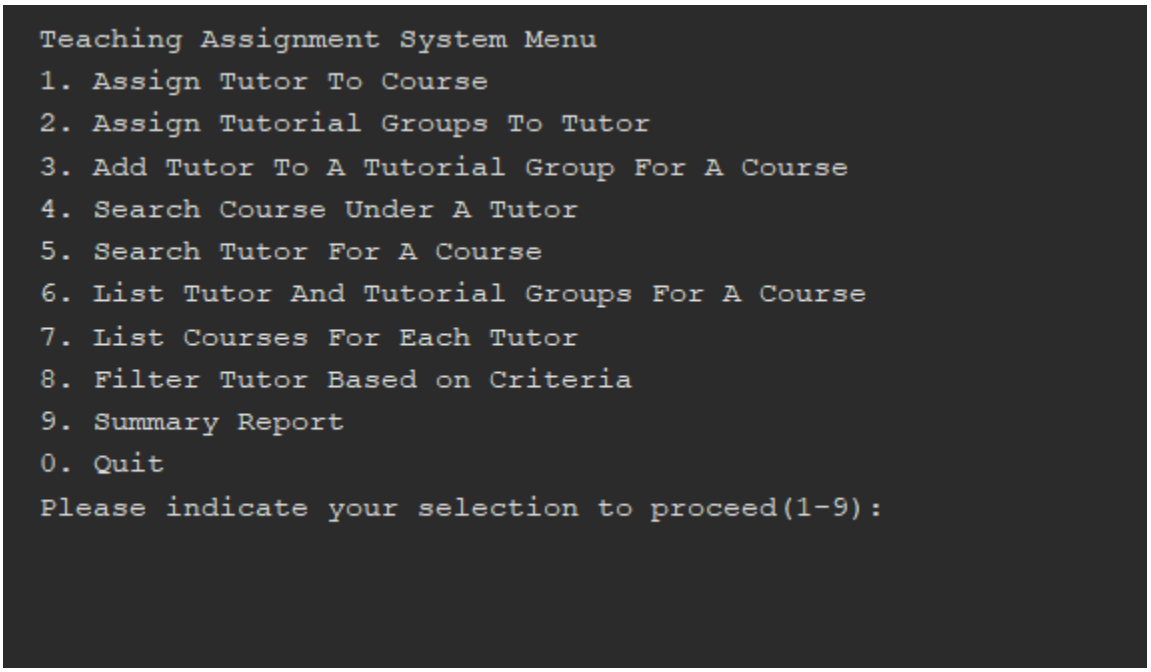


Figure 5.1 Menu of the Teaching Assignment Subsystem

Figure 2.1 shows the menu of the Teaching Assignment Subsystem and prompts the user for input options to proceed.

```
Please indicate your selection to proceed(1-9): 1


Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00            202401
BACS2026        Data Structure And Algorithms           4.00            202401
BAIT1023        Web Design And Development              2.00            202401
BAIT1043        System Analysis And Design              3.00            202401
BABS1233        Microbiology                            2.00            202401
BACH2233        Food Chemistry And Analysis             4.00            202401
BACH1613        Physical Chemistry                      3.00            202401
BTME4263        Heat Transfer                           3.00            202401
BGEE2614        Electrical Power Systems                2.00            202401
BTEE1513        Electrical Technology                   4.00            202401
BTEE4033        Integrated Circuit Technology           3.00            202401
BJEL1023        Academic English                        3.00            202401
BJEL1013        English For Tertiary Studies            3.00            202401
BAMS1413        Calculus And Algebra                    3.00            202401
BAMS2014        Linear Algebra                          3.00            202401

Enter Course Code:
```

Figure 5.2 Use case 1: Assign Tutor To Course

A course list is displayed to the user and prompts the user for input (choosing the course to be assigned to tutor).

```
Course Details:

-------------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00            202401
BACS2026        Data Structure And Algorithms           4.00            202401
BAIT1023        Web Design And Development              2.00            202401
BAIT1043        System Analysis And Design              3.00            202401
BABS1233        Microbiology                            2.00            202401
BACH2233        Food Chemistry And Analysis             4.00            202401
BACH1613        Physical Chemistry                      3.00            202401
BTME4263        Heat Transfer                           3.00            202401
BGEE2614        Electrical Power Systems                2.00            202401
BTEE1513        Electrical Technology                   4.00            202401
BTEE4033        Integrated Circuit Technology           3.00            202401
BJEL1023        Academic English                        3.00            202401
BJEL1013        English For Tertiary Studies            3.00            202401
BAMS1413        Calculus And Algebra                    3.00            202401
BAMS2014        Linear Algebra                          3.00            202401

Enter Course Code: BACS3074
Invalid course!
Enter Course Code:
```

Figure 5.3 Validate Course Code

User input is validated, if the user entered a wrong course code (which does not exist in the course list), an error message is displayed and prompt for user input again.

```
Enter Course Code: BACS3074
Invalid course!
Enter Course Code: BACS1053

Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my              010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my              010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my             018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my                014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my              012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my        019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my             012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my           017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my              012-3478838

Enter tutor id to proceed:
```

Figure 5.4 Display Tutor List

After a valid course code is entered, The system will display the tutor list and prompt the user to input tutor id to assign the tutor to the selected course.

```
Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my              010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my              010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my             018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my                014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my              012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my        019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my             012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my           017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my              012-3478838

Enter tutor id to assign to the course: T1020
Invalid Tutor ID!
Enter tutor id to assign to the course: T1000

Enter tutor type (L/T/P):
```

Figure 5.5 Validate Tutor ID

User input is validated, if the user entered a wrong tutor id (which does not exist in the tutor list), an error message is displayed and prompt for user input again. After a valid tutor id is entered, the system will proceed to specifying tutor type.

```
Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                                Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                           40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                  45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                              36    Female     Tanml@tarc.edu.my              010-5274012
T1003           Lim Wei Jie                               29    Male       Limwj@tarc.edu.my              010-0156282
T1004           Wong Kok Keong                            37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                              42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                             42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                                40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                              43    Female     Chanhm@tarc.edu.my             018-8212160
T1009           Teh Yee Ling                              29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                             29    Male       Ych@tarc.edu.my                014-1078771
T1011           Ong Chong Wei                             38    Male       Ongcw@tarc.edu.my              012-2592962
T1012           Nurul Azizah binti Abdullah               32    Female     Nurulazizah@tarc.edu.my        019-0208637
T1013           Siti Aishah binti Mohd Yusuf              38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                                37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                          42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                             55    Male       Chewkh@tarc.edu.my             012-7886162
T1017           Siti Sara Binti Mohd Ariff                35    Female     Sitisara@tarc.edu.my           017-3538834
T1018           Vijayamalini A/P Sathasivam               58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                              62    Male       Laiwk@tarc.edu.my              012-3478838

Enter tutor id to assign to the course: T1020
Invalid Tutor ID!
Enter tutor id to assign to the course: T1000

Enter tutor type (L/T/P): L
Lecturer Already Exists in This Course!
Enter tutor type (L/T/P): P

Record already exist!

Press enter to continue...
```

Figure 5.6 Validate Tutor Type And Check If The Record Has Already Existed

If the user input ‘L’ (Lecture) but the course has a lecturer already, the system will raise an error and prompt the user to reenter the tutor type. The system will also check if the record has already existed.

```
Enter Course Code: BACS1053

Tutor Details:

-------------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my              010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my              010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my             018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my                014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my              012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my        019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my             012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my           017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my              012-3478838

Enter tutor id to assign to the course: T1006

Enter tutor type (L/T/P): P

Assign tutor to course successfully!

Press enter to continue...
```

Figure 5.7 Assign Tutor To Course Successfully

```
Please indicate your selection to proceed(1-9): 2

Tutor Details:

-------------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my              010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my              010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my             018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my                014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my              012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my        019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my             012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my           017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my              012-3478838

Enter tutor id to proceed:
```

Figure 5.8 Use Case 2: Add Tutorial Group To Tutor

The system displays the tutor list and prompts the user to input tutor id to add a tutorial group to the tutor.

```
Enter tutor id to proceed: T1000


List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4
REE             REE0001         REE Group 1
RBS             RBS0001         RBS Group 1
RSW             RSW0001         RSW Group 1
RAN             RAN0001         RAN Group 1
RSD             RSD0001         RSD Group 1
RMM             RMM0001         RMM Group 1
RIS             RIS0001         RIS Group 1
RST             RST0001         RST Group 1
REI             REI0001         REI Group 1
RFN             RFN0001         RFN Group 1
RNR             RNR0001         RNR Group 1
RSE             RSE0001         RSE Group 1
RME             RME0001         RME Group 1
RAN             RAN0002         RAN Group 2
RMT             RMT0001         RMT Group 1
RMT             RMT0002         RMT Group 2
RMT             RMT0003         RMT Group 3

Enter the tutorial group id:
```

Figure 5.9 Display Programme/Tutorial Group List

The system displays the programme/tutorial group list and prompts the user to input tutorial group id to add the group to the selected tutor.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4
REE             REE0001         REE Group 1
RBS             RBS0001         RBS Group 1
RSW             RSW0001         RSW Group 1
RAN             RAN0001         RAN Group 1
RSD             RSD0001         RSD Group 1
RMM             RMM0001         RMM Group 1
RIS             RIS0001         RIS Group 1
RST             RST0001         RST Group 1
REI             REI0001         REI Group 1
RFN             RFN0001         RFN Group 1
RNR             RNR0001         RNR Group 1
RSE             RSE0001         RSE Group 1
RME             RME0001         RME Group 1
RAN             RAN0002         RAN Group 2
RMT             RMT0001         RMT Group 1
RMT             RMT0002         RMT Group 2
RMT             RMT0003         RMT Group 3

Enter the tutorial group id: RDS0005
Invalid Group ID!
Enter the tutorial group id: RDS0001


Record already exist!

Press enter to continue...
```

Figure 5.10 Validate Tutorial Group ID

User input is validated and redundancy of record is avoided. Corresponding error messages will be displayed to the user.

```
List of Tutorial Groups for Programme :

-----------------------------------------------------------------------------------------------------
Prgramme Code   Group ID        Group Name
-----------------------------------------------------------------------------------------------------
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4
REE             REE0001         REE Group 1
RBS             RBS0001         RBS Group 1
RSW             RSW0001         RSW Group 1
RAN             RAN0001         RAN Group 1
RSD             RSD0001         RSD Group 1
RMM             RMM0001         RMM Group 1
RIS             RIS0001         RIS Group 1
RST             RST0001         RST Group 1
REI             REI0001         REI Group 1
RFN             RFN0001         RFN Group 1
RNR             RNR0001         RNR Group 1
RSE             RSE0001         RSE Group 1
RME             RME0001         RME Group 1
RAN             RAN0002         RAN Group 2
RMT             RMT0001         RMT Group 1
RMT             RMT0002         RMT Group 2
RMT             RMT0003         RMT Group 3

Enter the tutorial group id: RDS0002


Tutorial group added successfully!

Press enter to continue...
```

Figure 5.11 Tutorial Group Added Successfully

```
Please indicate your selection to proceed(1-9): 3


Course Details:

-------------------------------------------------------------------------------------------------
Course Code     Course Name                          Credit Hour    Semester
-------------------------------------------------------------------------------------------------
BACS1053        Database Management                  3.00           202401
BACS2026        Data Structure And Algorithms        4.00           202401
BAIT1023        Web Design And Development           2.00           202401
BAIT1043        System Analysis And Design           3.00           202401
BABS1233        Microbiology                         2.00           202401
BACH2233        Food Chemistry And Analysis          4.00           202401
BACH1613        Physical Chemistry                   3.00           202401
BTME4263        Heat Transfer                        3.00           202401
BGEE2614        Electrical Power Systems             2.00           202401
BTEE1513        Electrical Technology                4.00           202401
BTEE4033        Integrated Circuit Technology        3.00           202401
BJEL1023        Academic English                     3.00           202401
BJEL1013        English For Tertiary Studies         3.00           202401
BAMS1413        Calculus And Algebra                 3.00           202401
BAMS2014        Linear Algebra                       3.00           202401

Enter Course Code:
```

Figure 5.12 Use Case 3: Add Tutor To Tutorial Group For A Course

The system displays the course list and prompts the user to input course code.

```
Enter Course Code: BACS2026


Programme Groups Under: BACS2026

-----------------------------------------------
Programme       Group ID        Group Name
-----------------------------------------------
RSW             RSW0001         RSW Group 1
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter the tutorial group id:
```

Figure 5.13 Display Programme/Tutorial Groups Under The Course

The system displays the list of programme/tutorial groups under the course and prompts the user to input tutorial group id.

```
Programme Groups Under: BACS2026

-----------------------------------------------
Programme       Group ID        Group Name
-----------------------------------------------
RSW             RSW0001         RSW Group 1
RDS             RDS0001         RDS Group 1
RDS             RDS0002         RDS Group 2
RDS             RDS0003         RDS Group 3
RDS             RDS0004         RDS Group 4

Enter the tutorial group id: RSW0002
Invalid Group ID!
```

Figure 5.14 Validate Tutorial Group ID

User input is validated, an error message is displayed when the user entered an invalid tutorial group id.

```
Enter the tutorial group id: RSW0001

Tutor Under: BACS2026

-------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my        017-4875233

Enter tutor id to proceed:
```

Figure 5.15 Display Tutor(s) Under The Course

After a valid group id is entered, the system displays the list of tutors who are assigned to the selected course. The system prompts the user to input tutor id to add the tutor to the selected tutorial group.

```
Tutor Under: BACS2026

-------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my        017-4875233

Enter tutor id to proceed: T1001


Record already exist!

Press enter to continue...
```

Figure 5.16 Check If The Record Has Already Existed

```
Enter the tutorial group id: RDS0004

Tutor Under: BACS2026

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my        017-4875233

Enter tutor id to proceed: T1001


Tutorial group added successfully!

Press enter to continue...
```

Figure 5.17 Tutorial Group Added Successfully

```
Please indicate your selection to proceed(1-9): 4

Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my          016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my        017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my           010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my           010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my          017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my           012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my           015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my           017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my          018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my           018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my             014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my           012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my     019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my      014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my            017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my         010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my          012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my        017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my    016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my           012-3478838

Enter tutor id to proceed:
```

Figure 5.18 Use Case 4: Search Course Under Tutor

The system displays the tutor list and prompts the user to input tutor id to view the courses under the tutor.

```
Enter tutor id to proceed: T1000


Courses Under Tutor ID: T1000

-------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401


Press enter to continue...
```

Figure 5.19 Display Course Taught By The Tutor

```
Please indicate your selection to proceed(1-9): 5


Course Details:

-------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401

Enter Course Code: BACS1053

Enter tutor type (L/T/P):
```

Figure 5.20 Use Case 5: Search Tutor For A Course (L/T/P)

The system displays the course list and prompts the user to input course code and tutor type to view the tutors (with the type) under that course.

```
Enter Course Code: BACS1053

Enter tutor type (L/T/P): P
Tutor Under: BACS1053
Tutor Type: P

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my          016-3115453
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my           015-9795730


Press enter to continue...
```

Figure 5.21 Display Tutors (P) Under The Course

```
Course Details:

-----------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-----------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401

Enter Course Code: BACS1053

Enter tutor type (L/T/P): T
Tutor Under: BACS1053
Tutor Type: T

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my          016-3115453


Press enter to continue...
```

Figure 5.22 Use Case 5 With Tutor Type = T

```
Please indicate your selection to proceed(1-9): 6


Course Details:

-------------------------------------------------------------------------------------------------
Course Code     Course Name                             Credit Hour    Semester
-------------------------------------------------------------------------------------------------
BACS1053        Database Management                     3.00           202401
BACS2026        Data Structure And Algorithms           4.00           202401
BAIT1023        Web Design And Development              2.00           202401
BAIT1043        System Analysis And Design              3.00           202401
BABS1233        Microbiology                            2.00           202401
BACH2233        Food Chemistry And Analysis             4.00           202401
BACH1613        Physical Chemistry                      3.00           202401
BTME4263        Heat Transfer                           3.00           202401
BGEE2614        Electrical Power Systems                2.00           202401
BTEE1513        Electrical Technology                   4.00           202401
BTEE4033        Integrated Circuit Technology           3.00           202401
BJEL1023        Academic English                        3.00           202401
BJEL1013        English For Tertiary Studies            3.00           202401
BAMS1413        Calculus And Algebra                    3.00           202401
BAMS2014        Linear Algebra                          3.00           202401

Enter Course Code:
```

Figure 5.23 Use Case 6: List Tutor And Tutorial Group For A Course

The system displays course list and prompts the user to input course code to view the tutors and tutorial groups in that course.

```
Enter Course Code: BACS1053

Tutors and Tutorial Groups List For: BACS1053

------------------------------------------------------------
GroupID             TutorID             Course
------------------------------------------------------------
RDS0001             T1000               BACS1053
RDS0003             T1000               BACS1053


Press enter to continue...
```

Figure 5.24 Display GroupID, TutorID and CourseCode For Selected Course

```
Please indicate your selection to proceed(1-9): 7

Tutor Course Details:

------------------------------------------------------
TutorID                 Course Code             Tutor Type
------------------------------------------------------
T1000                   BACS1053                [L, P]
T1001                   BACS2026                [L, P]
T1003                   BAIT1023                [P]
T1015                   BAMS1413                [L, T]


Press enter to continue...
```

Figure 5.25 Use Case 7: List Courses For Each Tutor

```
Please indicate your selection to proceed(1-9): 8

Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my          016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my        017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my           010-5274012
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my           010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my          017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my           012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my           015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my           017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my          018-8212160
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my           018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my             014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my           012-2592962
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my     019-0208637
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my      014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my            017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my         010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my          012-7886162
T1017           Siti Sara Binti Mohd Ariff              35    Female     Sitisara@tarc.edu.my        017-3538834
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my    016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my           012-3478838


Filter Tutor Criteria:
1. Age Above
2. Age Below
3. Gender
0. Back
Enter your choice:
```

Figure 5.26 Use Case 8: Filter Tutor Based On Criteria

The system displays the tutor list and prompts the user to input a selection of criteria.

```
Filter Tutor Criteria:
1. Age Above
2. Age Below
3. Gender
0. Back
Enter your choice: 1

Filter tutor by age above: 35
Tutor Details:

-------------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my             016-3115453
T1001           Thamarai A/P Subramaniam                45    Female     Thamarai@tarc.edu.my           017-4875233
T1002           Tan Mei Ling                            36    Female     Tanml@tarc.edu.my              010-5274012
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my             017-9950755
T1005           Loh Siew Hui                            42    Female     Lohsh@tarc.edu.my              012-5220749
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my              015-9795730
T1007           Ng Mei Yee                              40    Female     Ngwy6@tarc.edu.my              017-7022759
T1008           Chan Hui Min                            43    Female     Chanhm@tarc.edu.my             018-8212160
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my              012-2592962
T1013           Siti Aishah binti Mohd Yusuf            38    Female     Sitiaishah@tarc.edu.my         014-4799824
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my               017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my            010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my             012-7886162
T1018           Vijayamalini A/P Sathasivam             58    Female     Vijayamalini@tarc.edu.my       016-6547133
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my              012-3478838


Press enter to continue...
```

Figure 5.27 Filter By Age Above

If the user selects Age Above as the criteria to filter, the system will prompt the user to input age and display tutors who are above the age.

```
Filter Tutor Criteria:
1. Age Above
2. Age Below
3. Gender
0. Back
Enter your choice: 2

Filter tutor by age below: 35
Tutor Details:

-------------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                          Phone No
-------------------------------------------------------------------------------------------------------------------------
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my              010-0156282
T1009           Teh Yee Ling                            29    Female     Tehyl@tarc.edu.my              018-0690467
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my                014-1078771
T1012           Nurul Azizah binti Abdullah             32    Female     Nurulazizah@tarc.edu.my        019-0208637


Press enter to continue...
```

Figure 5.28 Filter By Age Below

If the user selects Age Below as the criteria to filter, the system will prompt the user to input age and display tutors who are below the age.

```
Filter Tutor Criteria:
1. Age Above
2. Age Below
3. Gender
0. Back
Enter your choice: 3

Enter tutor gender (Male/Female): Male
Tutor Details:

-------------------------------------------------------------------------------------------------------------------
TutorID         Tutor Name                              Age   Gender     Email                       Phone No
-------------------------------------------------------------------------------------------------------------------
T1000           Heng Jooi Huang                         40    Male       Hengjh@tarc.edu.my          016-3115453
T1003           Lim Wei Jie                             29    Male       Limwj@tarc.edu.my           010-0156282
T1004           Wong Kok Keong                          37    Male       Wongkk@tarc.edu.my          017-9950755
T1006           Lee Chee Seng                           42    Male       Leecs@tarc.edu.my           015-9795730
T1010           Yap Chin Hock                           29    Male       Ych@tarc.edu.my             014-1078771
T1011           Ong Chong Wei                           38    Male       Ongcw@tarc.edu.my           012-2592962
T1014           Ravi Kumar                              37    Male       Ravi@tarc.edu.my            017-0436496
T1015           Ngeoh Boon Heong                        42    Male       Ngeohbh@tarc.edu.my         010-3728188
T1016           Chew Kah Heng                           55    Male       Chewkh@tarc.edu.my          012-7886162
T1019           Lai Weng Kin                            62    Male       Laiwk@tarc.edu.my           012-3478838


Press enter to continue...
```

Figure 5.29 Filter By Gender

If the user selects Gender as the criteria to filter, the system will prompt the user to input gender, either Male/Female and display tutors who belong to that gender.

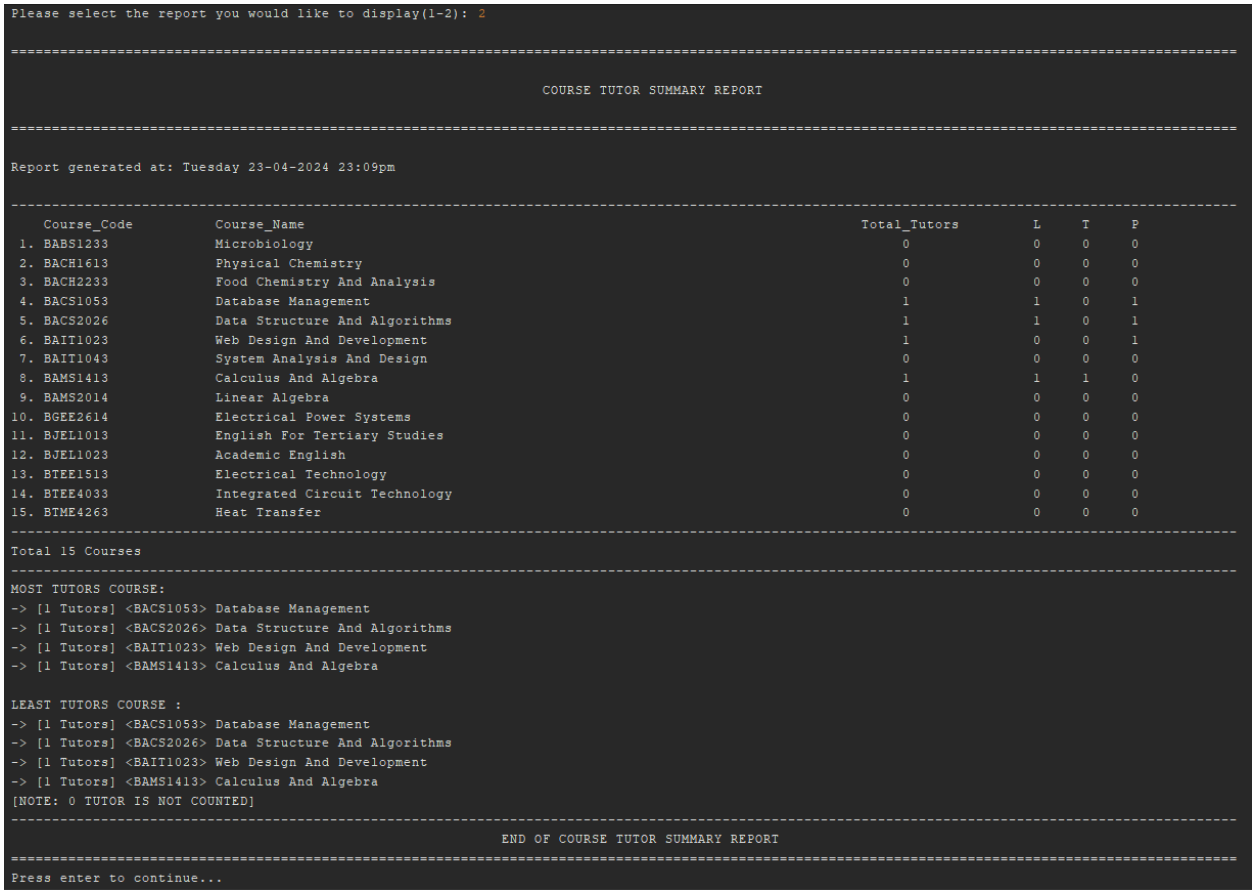
```
Teaching Assignment System Menu
1. Assign Tutor To Course
2. Assign Tutorial Groups To Tutor
3. Add Tutor To A Tutorial Group For A Course
4. Search Course Under A Tutor
5. Search Tutor For A Course
6. List Tutor And Tutorial Groups For A Course
7. List Courses For Each Tutor
8. Filter Tutor Based on Criteria
9. Summary Report
0. Quit
Please indicate your selection to proceed(1-9): 9


Summary Report Menu
1. Tutor Summary Report
2. Course Tutor Summary Report
0. Back
Please select the report you would like to display(1-2):
```

Figure 5.30 Use Case 9: Summary Report

The system displays the summary report menu and prompts the user to input report choice.

```
Please select the report you would like to display(1-2): 1

================================================================================================================================================================

                                                                TUTOR SUMMARY REPORT

================================================================================================================================================================

Report generated at: Tuesday 23-04-2024 23:09pm

----------------------------------------------------------------------------------------------------------------------------------------------------------------
    Tutor_ID            Tutor_Name                                                          Courses             Tutorial_Groups
 1. T1000               Heng Jooi Huang                                                       1                    3
 2. T1001               Thamarai A/P Subramaniam                                              1                    2
 3. T1002               Tan Mei Ling                                                          0                    0
 4. T1003               Lim Wei Jie                                                           1                    1
 5. T1004               Wong Kok Keong                                                        0                    0
 6. T1005               Loh Siew Hui                                                          0                    0
 7. T1006               Lee Chee Seng                                                         0                    0
 8. T1007               Ng Mei Yee                                                            0                    0
 9. T1008               Chan Hui Min                                                          0                    0
10. T1009               Teh Yee Ling                                                          0                    0
11. T1010               Yap Chin Hock                                                         0                    0
12. T1011               Ong Chong Wei                                                         0                    0
13. T1012               Nurul Azizah binti Abdullah                                           0                    0
14. T1013               Siti Aishah binti Mohd Yusuf                                          0                    0
15. T1014               Ravi Kumar                                                            0                    0
16. T1015               Ngeoh Boon Heong                                                      1                    0
17. T1016               Chew Kah Heng                                                         0                    0
18. T1017               Siti Sara Binti Mohd Ariff                                            0                    0
19. T1018               Vijayamalini A/P Sathasivam                                           0                    0
20. T1019               Lai Weng Kin                                                          0                    0
----------------------------------------------------------------------------------------------------------------------------------------------------------------
Total 20 Tutors
----------------------------------------------------------------------------------------------------------------------------------------------------------------
HIGHEST COURSES HANDLED:
-> [1 Courses] <T1000> Heng Jooi Huang
-> [1 Courses] <T1001> Thamarai A/P Subramaniam
-> [1 Courses] <T1003> Lim Wei Jie
-> [1 Courses] <T1015> Ngeoh Boon Heong

LOWEST COURSES HANDLED :
-> [1 Courses] <T1000> Heng Jooi Huang
-> [1 Courses] <T1001> Thamarai A/P Subramaniam
-> [1 Courses] <T1003> Lim Wei Jie
-> [1 Courses] <T1015> Ngeoh Boon Heong
[NOTE: 0 COURSES IS NOT COUNTED]
----------------------------------------------------------------------------------------------------------------------------------------------------------------
MOST TUTORIAL GROUPS HANDLED TUTOR(S):
-> [3 Tutorial Groups] <T1000> Heng Jooi Huang

LEAST TUTORIAL GROUPS HANDLED TUTOR(S):
-> [1 Tutorial Groups] <T1003> Lim Wei Jie
[NOTE: 0 TUTORIAL GROUPS IS NOT COUNTED]
----------------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                END OF TUTOR SUMMARY REPORT
================================================================================================================================================================
Press enter to continue...
```

Figure 5.31 Display Tutor Summary Report



```
Please select the report you would like to display(1-2): 2

================================================================================================================================================================

                                                                COURSE TUTOR SUMMARY REPORT

================================================================================================================================================================

Report generated at: Tuesday 23-04-2024 23:09pm

----------------------------------------------------------------------------------------------------------------------------------------------------------------
    Course_Code          Course_Name                                                                Total_Tutors         L     T     P
 1. BABS1233             Microbiology                                                                  0              0     0     0
 2. BACH1613             Physical Chemistry                                                            0              0     0     0
 3. BACH2233             Food Chemistry And Analysis                                                   0              0     0     0
 4. BACS1053             Database Management                                                           1              1     0     1
 5. BACS2026             Data Structure And Algorithms                                                 1              1     0     1
 6. BAIT1023             Web Design And Development                                                    1              0     0     1
 7. BAIT1043             System Analysis And Design                                                    0              0     0     0
 8. BAMS1413             Calculus And Algebra                                                          1              1     1     0
 9. BAMS2014             Linear Algebra                                                                0              0     0     0
10. BGEE2614             Electrical Power Systems                                                      0              0     0     0
11. BJEL1013             English For Tertiary Studies                                                  0              0     0     0
12. BJEL1023             Academic English                                                              0              0     0     0
13. BTEE1513             Electrical Technology                                                         0              0     0     0
14. BTEE4033             Integrated Circuit Technology                                                 0              0     0     0
15. BTME4263             Heat Transfer                                                                 0              0     0     0
----------------------------------------------------------------------------------------------------------------------------------------------------------------
Total 15 Courses
----------------------------------------------------------------------------------------------------------------------------------------------------------------
MOST TUTORS COURSE:
-> [1 Tutors] <BACS1053> Database Management
-> [1 Tutors] <BACS2026> Data Structure And Algorithms
-> [1 Tutors] <BAIT1023> Web Design And Development
-> [1 Tutors] <BAMS1413> Calculus And Algebra

LEAST TUTORS COURSE :
-> [1 Tutors] <BACS1053> Database Management
-> [1 Tutors] <BACS2026> Data Structure And Algorithms
-> [1 Tutors] <BAIT1023> Web Design And Development
-> [1 Tutors] <BAMS1413> Calculus And Algebra
[NOTE: 0 TUTOR IS NOT COUNTED]
----------------------------------------------------------------------------------------------------------------------------------------------------------------
                                                              END OF COURSE TUTOR SUMMARY REPORT
================================================================================================================================================================
Press enter to continue...
```

Figure 5.32 Display Course Summary Report