FACULTY OF COMPUTING

UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITHM
(SECJ2013)

SEMESTER 1 2024/2025

Mini Project Documentation
E-Learning Assignment System

By

EVELYN ANG (050723-01-0452) - Leader
TEOH XIN YEE (050610-07-0512)
TOH SHEE THONG (051004-01-1242)

SECTION 02

Lecturer:
Dr. Pang Yee Yong

Date:
9 JANUARY 2026

For lecturer use:

Description	**Mark Distribution**	**Mark**
Project Report • System analysis • Design • Program code	 10 15 25	
Presentation & Demo	25	
System Prototype	25	
TOTAL	**100**	

Table of Contents

PART 1: INTRODUCTION

1.1 Synopsis Project

This mini project involves the development of an E-Learning Assignment System using C++ programming languages to manage assignment submissions. It allows students to submit the assignments digitally while the lecturers can create assignments, view submissions, mark assignments.

The system implements various concepts of data structures to manage and organize data efficiently. It uses **linked lists** to store assignment submissions dynamically and **queues** are used to keep track of the order of submissions based on the FIFO principle. Moreover, it can handle assignment creation undo operations using **stacks** and perform fast searches for assignments by their titles using **Binary Search Trees (BST**) which have a time complexity of O(log n). In addition, **sorting** algorithms are used to rank students based on their total scores.

The system begins with a simple login process, where users need to choose a role either student or lecturer to gain access to the system. Depending on that role, various features will be provided by the system. In this aspect, students will be able to view the available assignments, search for any assignment using BST algorithms, and submit their assignments. After grading, students will be able to view their marks once grading is done by lecturers. Additionally, students can view class rankings to see their performance compared to peers.

For lecturers, the system enables them to create new assignments by entering the details such as title, description, due date and maximum score. Then, assignments are automatically sorted by due date. They can check the history of assignment creation in LIFO order. Moreover, they can use the undo feature to remove the recently created assignment. They can also check the list of all assignment submissions in FIFO order then grade the assignments through the input of marks and comments. Finally, they can check the ranks of students depending on the total marks sorted in the highest first.

1.2 Objective of the project

The objectives of this projects are:

- To design and develop an E-Learning Assignment System for managing assignment submission.
- To allow students to submit assignments digitally and view their submission marks.
- To enable lecturers to create assignments, view student submissions, and grade assignments efficiently.
- To apply and demonstrate the use of data structures such as linked lists, queues, stacks, and binary search trees in the real world.
- To enhance understanding of how data structures improve data organization, searching, and sorting in real-world applications.

PART 2: SYSTEM ANALYSIS AND DESIGN

2.1 System Requirements

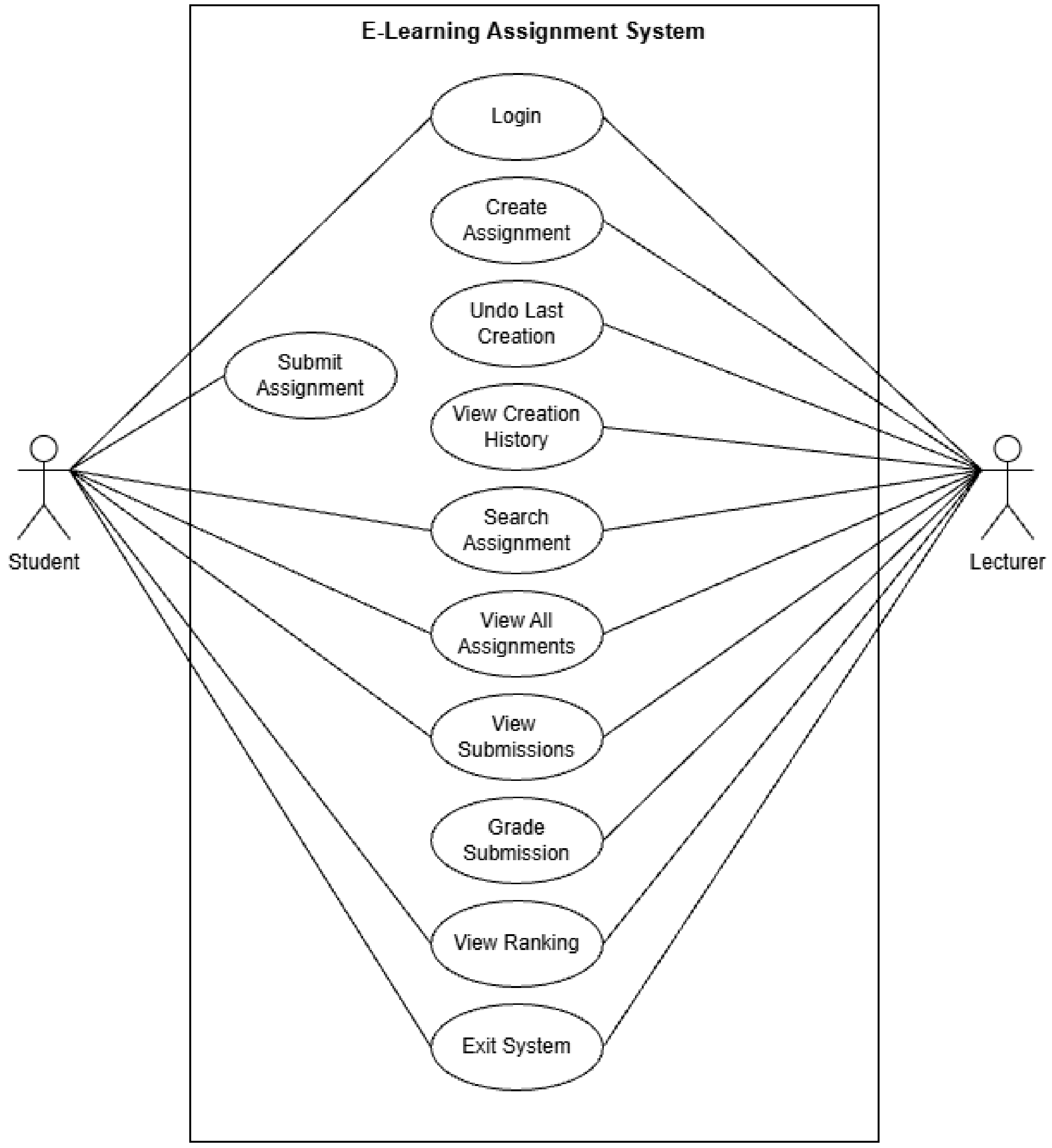


Figure 1: Use Case Diagram for E-Learning Assignment System

Use Case Description for E-Learning Assignment System

The primary users of the system are students and lecturers.

Table 2.1.1: Use Case Description for E-Learning Assignment System

Actor	**Task**
Student	The students can log in to the system to view all assignments, search specific assignment, submit assignments and view their submissions. Once the lecturer grades the assignment, they can check the marks and feedback via the system, view their rankings and then exit the system.
Lecturer	The lecturer is responsible for managing the assignments within the system. The lecturer can log in to the system and create assignments. They can view assignment creation history and undo the last created assignment if needed. Once the students submit the assignments, they can view all student submissions, and grade assignments. The lecturer can also view student performance rankings before exiting the system.

Detail Description for Each Use Cases

In our proposed system, there are 11 main use cases as shown in the table below:

Table 2.1.2: Detail Description for Each Use Case

Use Case	**Purpose**
Login	This use case allows students and lecturers to login into the system and access appropriate functions based on their role.
Create Assignment	This use case enables the lecturer to create new assignments by providing details such as ID, title, and deadline.
Undo Last Creation	This use case allows lecturers to undo the most recently created assignment.
View Creation History	This use case allows lecturers to view the history of assignment creations in the order they were added.
Search Assignment	This use case enables both students and lecturers to search assignments using specific criteria such as assignment ID or title.
View All Assignment	This use case allows users to view all assignments that are available in the system.
Submit Assignment	This use case allows students to submit their assignments using the system. The submission is stored in relation to the order of submission for lecturer grading.
View Submissions	This use case allows both students and lecturers to view submission of assignments. Students are able to see their own submission details, while lecturers can view all student submissions for a selected assignment.
Grade Submission	This use case allows lecturers to assign marks to student submissions and update grading records in the system.
View Ranking	This use case allows users to view rankings based on assignment marks.
Exit System	This use case allows lecturers and students to exit the system after completing their tasks.

2.2 System Design

2.2.1 Class Diagram

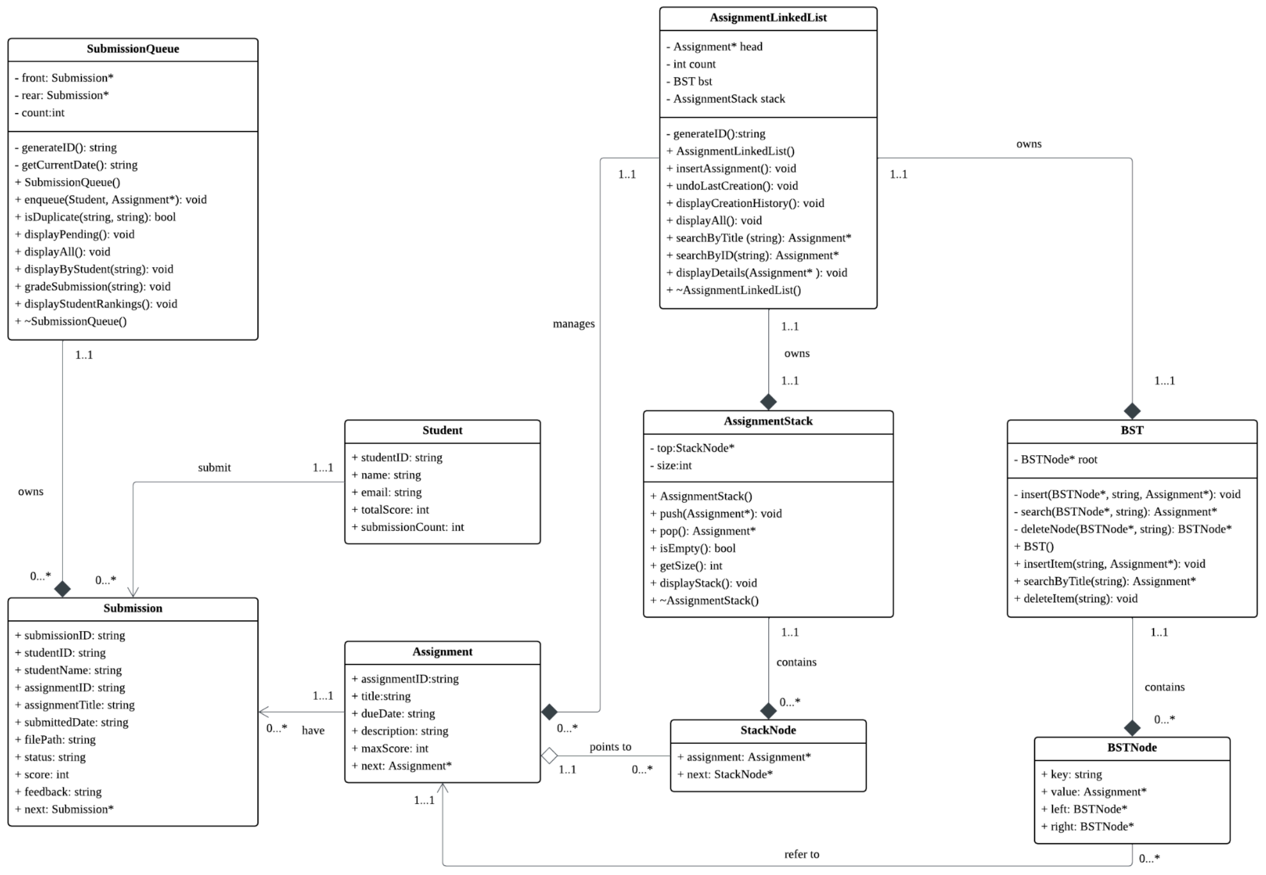


Figure 1: Class Diagram

2.2.2 Explanation

Feature	Data Structure	Reason	Time Complexity
Create Assignment	Linked list	Can store assignments dynamically	O(n)
	BST	Enables fast searching of assignment by title	O(log n)
	Stack	Support undo functionality which is Last In First Out	O(n)
Undo Assignment	Stack	Retrieve the most recently created assignment efficiently	O(1)
	Linked List	Removes the assignment from the stored list	O(n)
	BST	Removes the assignment from search structure	O(log n)
Search Assignment	BST	Search by title that allows faster search compared to linear search	O(log n)
	Linked List	Search by ID	O(n)
Submit Assignment	Queue	Submission processed in First-In-First-Out (FIFO) order.	O(n)
Grade Submission	Queue	Grading also follows the order of submissions.	O(n)
Display Assignments	Linked List	Assignments are stored dynamically.	O(n)
Student Ranking	Bubble Sort	Suitable for a small number of students and no additional memory required	$O(n^2)$

2.2.3 Pseudo Code

Pseudo Code 1: System Entry Point

```
1.    PRINT "Welcome to E-Learning Assignment System"
2.    REPEAT
      2.1.    DISPLAY main menu:
              - "1. Student Login"
              - "2. Lecturer Login"
              - "0. Exit System"
      2.2.    READ userChoice
      2.3.    IF userChoice == 1
          2.3.1.    CALL Student Menu
      2.4.    ELSE IF userChoice == 2
          2.4.1.    CALL Lecturer Menu
      2.5.    ELSE IF userChoice == 0
          2.5.1.    PRINT "Thank you! Goodbye!"
          2.5.2.    TERMINATE SYSTEM
      2.6.    ELSE
          2.6.1.    PRINT "Invalid choice. Please try again."
      2.7.    END IF
3.    UNTIL userChoice == 0
```

Prepared by: EVELYN ANG

Pseudo Code 2: Student Login and Registration

```
1. PRINT "Please enter your credentials:"
2. INPUT studentID
3. INPUT studentName
4. INPUT studentEmail
5. STORE into CurrentStudent
```

Prepared by: EVELYN ANG

Pseudo Code 3: Student Main Menu Interaction

```
1.      REPEAT
    1.1.    DISPLAY Student Menu
    1.2.    PRINT "1. View All Available Assignments"
    1.3.    PRINT "2. Search for Specific Assignment"
    1.4.    PRINT "3. Submit an Assignment"
    1.5.    PRINT "4. View My Submission History"
    1.6.    PRINT "5. View Class Rankings"
    1.7.    PRINT "6. Logout"
    1.8.    READ studentChoice
    1.9.    IF studentChoice ==1
        1.9.1.    DISPLAY assignments
    1.10.   ELSE IF studentChoice ==2
        1.10.1.   ASK search method (Title / ID)
        1.10.2.   IF Title THEN
            1.10.2.1.   SEARCH assignment using BST
        1.10.3.   ELSE
            1.10.3.1.   SEARCH assignment using Linked List
        1.10.4.   END IF
        1.10.5.   DISPLAY assignment details
    1.11.   ELSE IF studentChoice ==3
        1.11.1.   DISPLAY all assignments
        1.11.2.   INPUT assignment ID
        1.11.3.   IF assignment not found THEN
            1.11.3.1.   DISPLAY error
        1.11.4.   ELSE IF duplicate submission detected THEN
            1.11.4.1.   DISPLAY "Already submitted"
        1.11.5.   ELSE
            1.11.5.1.   ENQUEUE submission into Submission Queue
        1.11.6.   END IF
    1.12.   ELSE IF studentChoice ==4
        1.12.1.    DISPLAY submissions belonging to CurrentStudent
    1.13.   ELSE IF studentChoice ==5
        1.13.1.   CALCULATE total scores per student
        1.13.2.   SORT students by total score
        1.13.3.    DISPLAY ranking table
    1.14.   ELSE IF studentChoice ==6
        1.14.1.   LOGOUT student
        1.14.2.   RETURN to main menu
    1.15.   ELSE
```

```
            1.15.1.    PRINT "Invalid choice. Please enter 1-6."
        1.16.    END IF
    2.    UNTIL studentChoice == 6
```

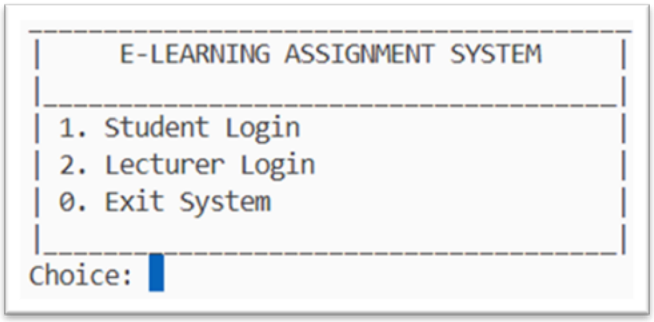
Prepared by: TOH SHEE THONG

Pseudo Code 4: Lecturer Main Menu

```
1.    REPEAT
    1.1.    DISPLAY Lecturer Menu
        1.1.1.    PRINT "What would you like to do?"
        1.1.2.    PRINT "1. Create New Assignment"
        1.1.3.    PRINT "2. Undo Last Assignment Creation (Undo feature)"
        1.1.4.    PRINT "3. View Assignment Creation History"
        1.1.5.    PRINT "4. View All Assignments"
        1.1.6.    PRINT "5. Search for Assignment"
        1.1.7.    PRINT "6. View Pending Submissions (Need Grading)"
        1.1.8.    PRINT "7. Grade a Submission"
        1.1.9.    PRINT "8. View Student Rankings"
        1.1.10.   PRINT "9. Logout"
    1.2.    READ lecturerChoice
    1.3.    IF lecturerChoice ==1
        1.3.1.    CREATE new assignment
        1.3.2.    GENERATE assignment ID
        1.3.3.    INSERT assignment into:
            1.3.3.1.    Linked List
            1.3.3.2.    BST
            1.3.3.3.    Stack
    1.4.    ELSE IF lecturerChoice==2
        1.4.1.    IF Stack is empty THEN
            1.4.1.1.    DISPLAY "No assignment to undo"
        1.4.2.    ELSE
            1.4.2.1.    POP assignment from Stack
            1.4.2.2.    REMOVE assignment from Linked List
            1.4.2.3.    REMOVE assignment from BST
        1.4.3.    END IF
    1.5.    ELSE IF lecturerChoice==3
        1.5.1.    DISPLAY assignment creation history (Stack LIFO)
    1.6.    ELSE IF lecturerChoice==4
        1.6.1.     DISPLAY all assignments (Linked List)
    1.7.  ELSE IF lecturerChoice==5
        1.7.1.  INPUT assignment title
        1.7.2.  SEARCH assignment using BST
        1.7.3.  DISPLAY assignment details
    1.8.  ELSE IF lecturerChoice==6
        1.8.1.  DISPLAY all pending submissions (Queue FIFO)
    1.9.  ELSE IF lecturerChoice==7
        1.9.1.  DISPLAY pending submissions
        1.9.2.  INPUT submission ID
        1.9.3.  IF submission found AND not graded THEN
            1.9.3.1.  INPUT score and feedback
            1.9.3.2.  UPDATE submission status to "Graded"
        1.9.4.  ELSE
            1.9.4.1.  DISPLAY error
        1.9.5.  END IF
    1.10.  ELSE IF lecturerChoice==8
        1.10.1.  CALCULATE student total scores
        1.10.2.  SORT students by total score
        1.10.3.  DISPLAY rankings
    1.11.  ELSE IF lecturerChoice==9
        1.11.1.  LOGOUT lecturer
        1.11.2.  RETURN to main manu
    1.12.  ELSE
        1.12.1.  DISPLAY "Invalid choice"
    1.13.  END IF
2.  UNTIL lecturerChoice == 9
```

Prepared by: TEOH XIN YEE

PART 3: SYSTEM PROTOTYPE

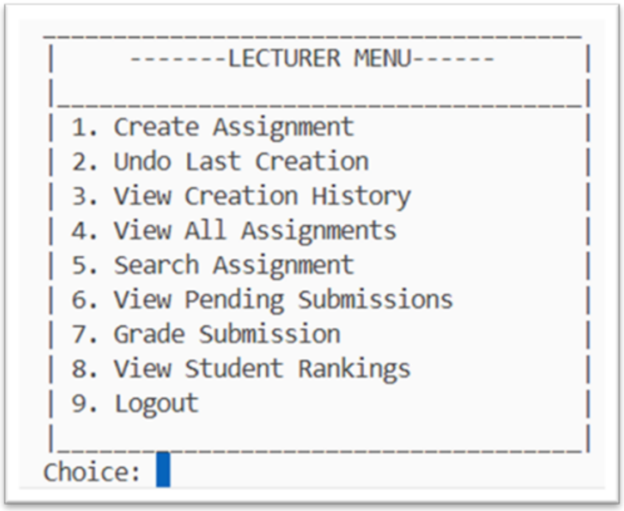


```
 _________________________________________
|      E-LEARNING ASSIGNMENT SYSTEM       |
|_________________________________________|
| 1. Student Login                        |
| 2. Lecturer Login                       |
| 0. Exit System                          |
|_________________________________________|
Choice: 
```

Screen 1: Main Menu

This screen is the main menu of the E-Learning Assignment System, where users select their role by entering a number. If the user selects 1, the system proceeds to Student Login, and if 2 is selected, it proceeds to Lecturer Login. If an invalid input enters, the system will prompt an error message and the screen will be displayed again.

Prepared By: EVELYN ANG

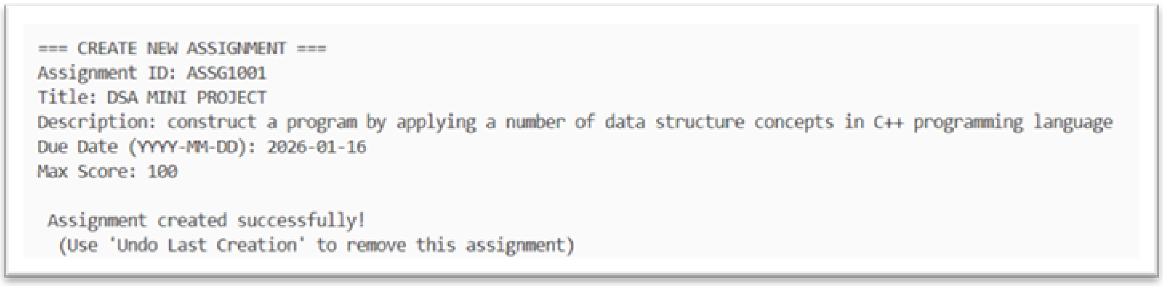


```
 ______________________________________
|      -------LECTURER MENU------      |
|______________________________________|
| 1. Create Assignment                 |
| 2. Undo Last Creation                |
| 3. View Creation History             |
| 4. View All Assignments              |
| 5. Search Assignment                 |
| 6. View Pending Submissions          |
| 7. Grade Submission                  |
| 8. View Student Rankings             |
| 9. Logout                            |
|______________________________________|
Choice: 
```

Screen 2: Lecturer Menu

Lecturer Menu allows the lecturer to manage the assignment workflow, including creating new assignments, undoing the last creation, viewing and searching existing assignments, and checking pending submissions. The lecturer can also grade student submissions and view student rankings.

Prepared By: EVELYN ANG

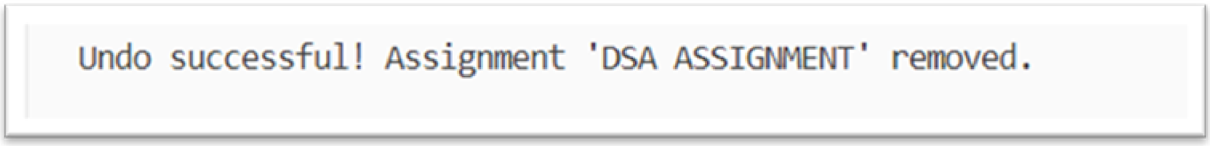


```
=== CREATE NEW ASSIGNMENT ===
Assignment ID: ASSG1001
Title: DSA MINI PROJECT
Description: construct a program by applying a number of data structure concepts in C++ programming language
Due Date (YYYY-MM-DD): 2026-01-16
Max Score: 100

 Assignment created successfully!
  (Use 'Undo Last Creation' to remove this assignment)
```

Screen 3: Create New Assignment System

Create New Assignment allows the lecturer to create a new assignment where the assignment ID is automatically generated by the system. The lecturer is required to enter the assignment title, description, due date, and maximum score. Once the assignment is created successfully, the system displays a confirmation message indicating that the assignment has been created.

Prepared By: EVELYN ANG

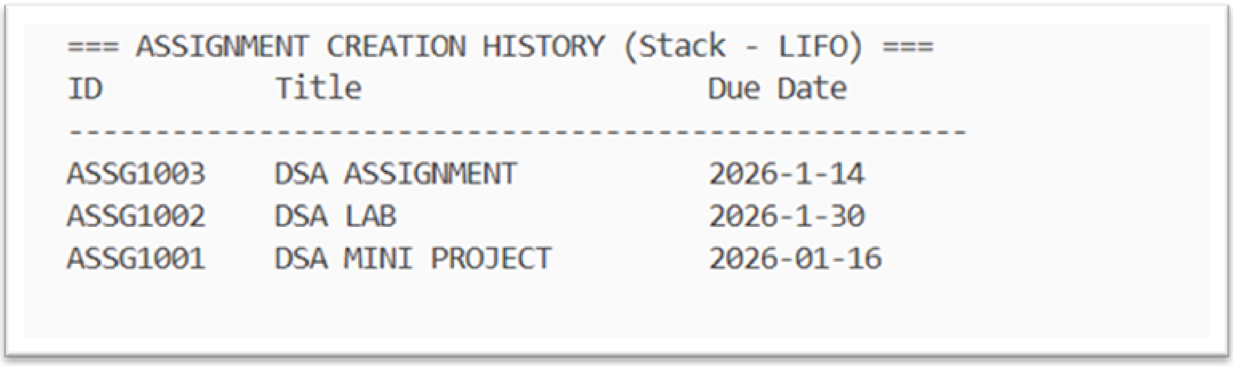


```
Undo successful! Assignment 'DSA ASSIGNMENT' removed.
```

Screen 4: Undo Last Assignment Creation System

Screen 4: Undo Last Assignment Creation demonstrates the use of the stack (Last In, First Out) concept, where the most recently created assignment is removed first. When the undo operation is successful, the system displays a confirmation message indicating that the latest assignment has been deleted.

Prepared By: EVELYN ANG

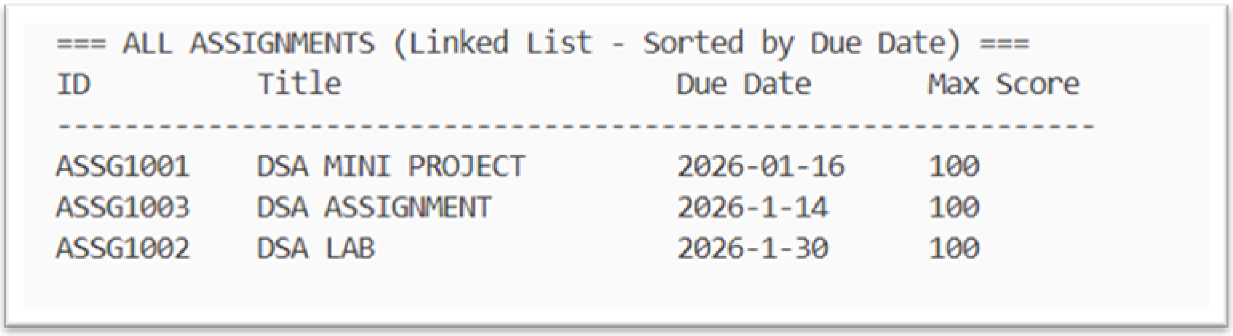


```
=== ASSIGNMENT CREATION HISTORY (Stack - LIFO) ===
ID          Title                       Due Date
----------------------------------------------------
ASSG1003    DSA ASSIGNMENT              2026-1-14
ASSG1002    DSA LAB                     2026-1-30
ASSG1001    DSA MINI PROJECT            2026-01-16
```

Screen 5: Assignment Creation History System

Assignment Creation History displays all previously created assignments stored in a stack using the Last In, First Out (LIFO) concept. The most recently created assignment appears at the top of the list. This design allows the lecturer to easily undo the latest assignment creation when needed.

Prepared By: TOH SHEE THONG

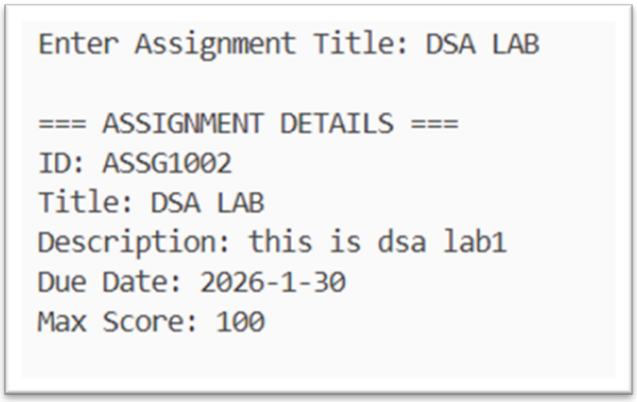


```
=== ALL ASSIGNMENTS (Linked List - Sorted by Due Date) ===
ID          Title                       Due Date      Max Score
-------------------------------------------------------------
ASSG1001    DSA MINI PROJECT            2026-01-16    100
ASSG1003    DSA ASSIGNMENT              2026-1-14     100
ASSG1002    DSA LAB                     2026-1-30     100
```

Screen 6: View All Assignments System

View All Assignments displays all assignments stored in a linked list and sorted by their due dates. Each assignment is shown with its ID, title, due date, and maximum score

Prepared By: TOH SHEE THONG

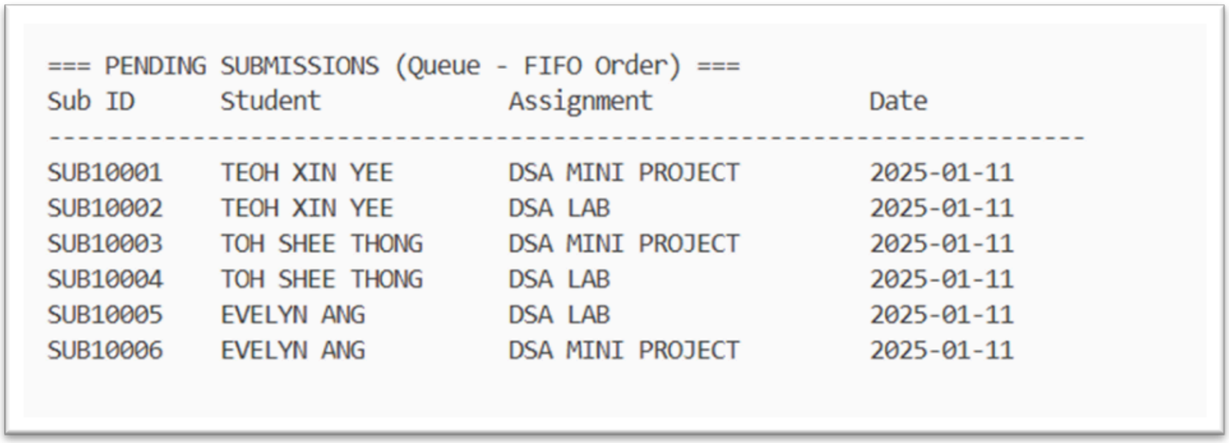


```
Enter Assignment Title: DSA LAB

=== ASSIGNMENT DETAILS ===
ID: ASSG1002
Title: DSA LAB
Description: this is dsa lab1
Due Date: 2026-1-30
Max Score: 100
```

Screen 7: Search Assignment System

Search Assignment (by Title) allows the user to search for a specific assignment by entering the assignment title. The system uses a Binary Search Tree (BST) to locate the matching assignment quickly, then displays the full assignment details such as ID, description, due date, and maximum score.

Prepared By: TOH SHEE THONG

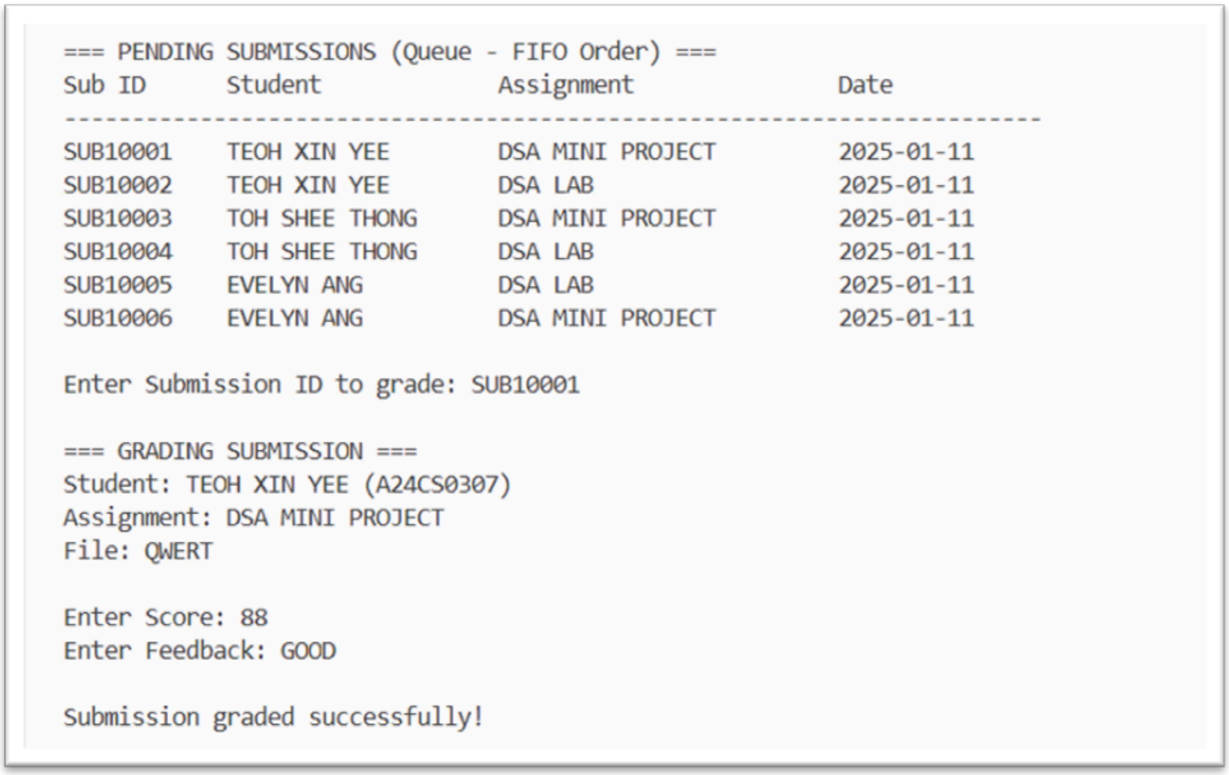


```
=== PENDING SUBMISSIONS (Queue - FIFO Order) ===
Sub ID      Student            Assignment              Date
------------------------------------------------------------------------
SUB10001    TEOH XIN YEE       DSA MINI PROJECT        2025-01-11
SUB10002    TEOH XIN YEE       DSA LAB                 2025-01-11
SUB10003    TOH SHEE THONG     DSA MINI PROJECT        2025-01-11
SUB10004    TOH SHEE THONG     DSA LAB                 2025-01-11
SUB10005    EVELYN ANG         DSA LAB                 2025-01-11
SUB10006    EVELYN ANG         DSA MINI PROJECT        2025-01-11
```

Screen 8: View Pending Submissions System

View Pending Submissions displays all student submissions that have not yet been graded, stored in a queue following the First In, First Out (FIFO) principle. The screen shows the submission ID, student name, assignment title, and submission date, allowing pending submissions to be reviewed in the order they were received.

Prepared By: TOH SHEE THONG

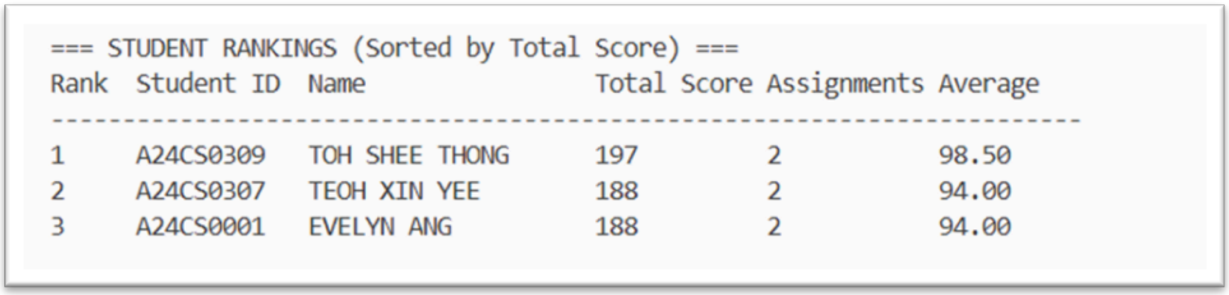


```
=== PENDING SUBMISSIONS (Queue - FIFO Order) ===
Sub ID      Student             Assignment               Date
------------------------------------------------------------------------
SUB10001    TEOH XIN YEE        DSA MINI PROJECT         2025-01-11
SUB10002    TEOH XIN YEE        DSA LAB                  2025-01-11
SUB10003    TOH SHEE THONG      DSA MINI PROJECT         2025-01-11
SUB10004    TOH SHEE THONG      DSA LAB                  2025-01-11
SUB10005    EVELYN ANG          DSA LAB                  2025-01-11
SUB10006    EVELYN ANG          DSA MINI PROJECT         2025-01-11

Enter Submission ID to grade: SUB10001

=== GRADING SUBMISSION ===
Student: TEOH XIN YEE (A24CS0307)
Assignment: DSA MINI PROJECT
File: QWERT

Enter Score: 88
Enter Feedback: GOOD

Submission graded successfully!
```

Screen 9: Grade Submission System

Grade Submission allows the lecturer to select a submission from the pending submissions queue (FIFO) by entering the submission ID. The system then displays the submission details and enables the lecturer to enter the score and feedback. Once completed, the submission status is updated to graded, and a confirmation message is shown.

Prepared By: TOH SHEE THONG

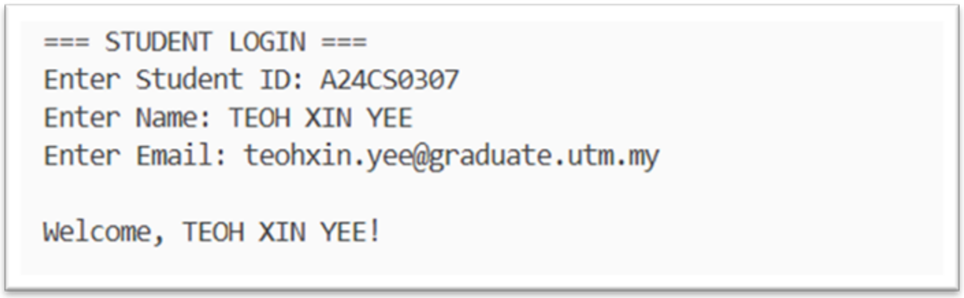


```
=== STUDENT RANKINGS (Sorted by Total Score) ===
Rank  Student ID  Name                Total Score Assignments Average
------------------------------------------------------------------------
1     A24CS0309   TOH SHEE THONG      197         2           98.50
2     A24CS0307   TEOH XIN YEE        188         2           94.00
3     A24CS0001   EVELYN ANG          188         2           94.00
```

Screen 10: View Student Rankings System

View Student Rankings displays the ranking of students based on their total scores, which are calculated from graded submissions. The results are sorted in descending order using a sorting algorithm, showing each student's rank, total score, number of assignments submitted, and average score.

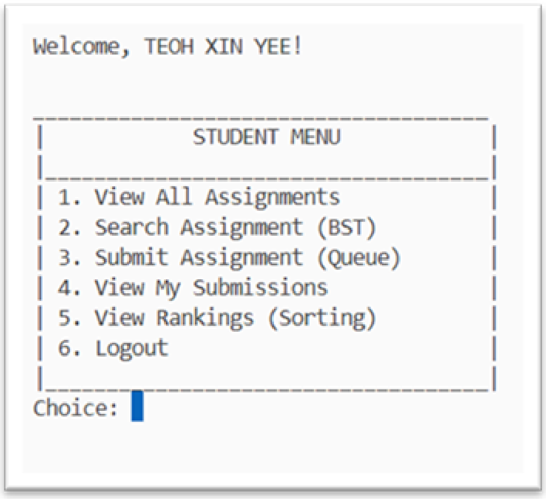
Prepared By: TOH SHEE THONG


```
=== STUDENT LOGIN ===
Enter Student ID: A24CS0307
Enter Name: TEOH XIN YEE
Enter Email: teohxin.yee@graduate.utm.my

Welcome, TEOH XIN YEE!
```

Screen 11: Student Login System

Student Login allows the student to enter their student ID, name, and email to access the system. After successful login, the system displays a welcome message.

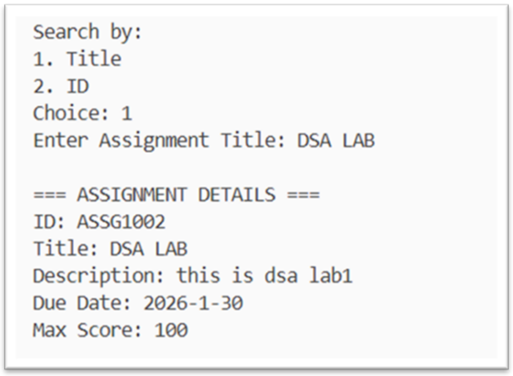
Prepared By: TEOH XIN YEE


```
Welcome, TEOH XIN YEE!

 ___________________________________
|           STUDENT MENU            |
|___________________________________|
| 1. View All Assignments           |
| 2. Search Assignment (BST)        |
| 3. Submit Assignment (Queue)      |
| 4. View My Submissions            |
| 5. View Rankings (Sorting)        |
| 6. Logout                         |
|___________________________________|
Choice:
```

Screen 12: Student Menu

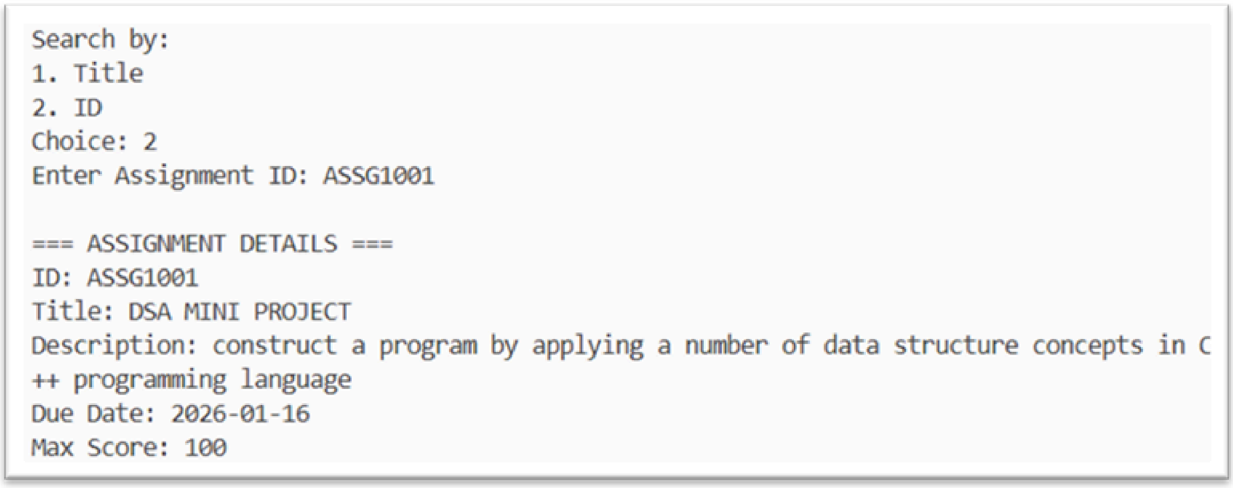
Student Menu displays the main set of functions available after a student successfully logs in. From this menu, the student can view and search assignments, submit assignments using a queue, view their submission history, check rankings based on sorting, or log out of the system.

Prepared By: TEOH XIN YEE



```
Search by:
1. Title
2. ID
Choice: 1
Enter Assignment Title: DSA LAB

=== ASSIGNMENT DETAILS ===
ID: ASSG1002
Title: DSA LAB
Description: this is dsa lab1
Due Date: 2026-1-30
Max Score: 100
```

Screen 13: Search Assignment by Title System

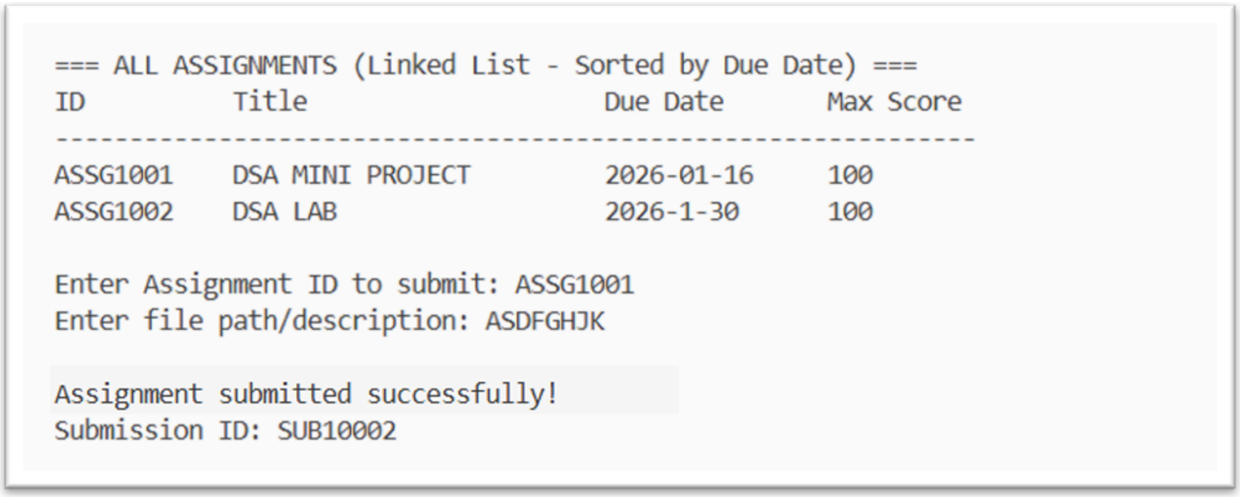


```
Search by:
1. Title
2. ID
Choice: 2
Enter Assignment ID: ASSG1001

=== ASSIGNMENT DETAILS ===
ID: ASSG1001
Title: DSA MINI PROJECT
Description: construct a program by applying a number of data structure concepts in C
++ programming language
Due Date: 2026-01-16
Max Score: 100
```

Screen 14: Search Assignment by Assignment Id System

The Search Assignment system allows students to choose a search method, either by title or by assignment ID. Searching by title uses the BST for faster retrieval, while searching by ID uses a linked list traversal, and the system then displays the selected assignment's full details.

Prepared By: TEOH XIN YEE

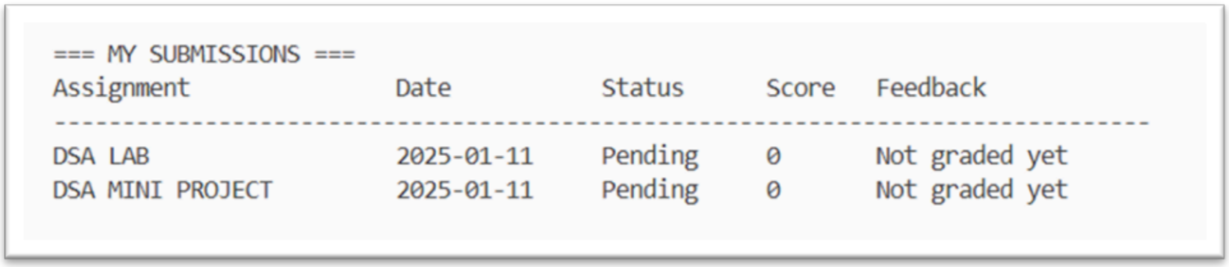


```
=== ALL ASSIGNMENTS (Linked List - Sorted by Due Date) ===
ID          Title                     Due Date       Max Score
--------------------------------------------------------------
ASSG1001    DSA MINI PROJECT          2026-01-16     100
ASSG1002    DSA LAB                   2026-1-30      100

Enter Assignment ID to submit: ASSG1001
Enter file path/description: ASDFGHJK

Assignment submitted successfully!
Submission ID: SUB10002
```

Screen 15: Submit Assignment System

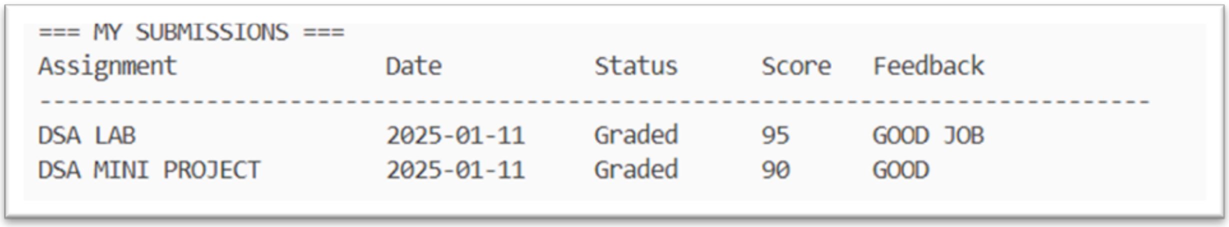
Submit Assignment allows the student to select an assignment from the list and submit it by entering the assignment ID and file path or description. The submission is stored in a queue (FIFO), and upon successful submission, the system displays a confirmation message along with the generated submission ID.

Prepared By: TEOH XIN YEE



```
=== MY SUBMISSIONS ===
Assignment               Date          Status      Score   Feedback
-------------------------------------------------------------------------------
DSA LAB                  2025-01-11    Pending     0       Not graded yet
DSA MINI PROJECT         2025-01-11    Pending     0       Not graded yet
```

Screen 17: View Submission System (before graded)



```
=== MY SUBMISSIONS ===
Assignment               Date          Status      Score   Feedback
-------------------------------------------------------------------------------
DSA LAB                  2025-01-11    Graded      95      GOOD JOB
DSA MINI PROJECT         2025-01-11    Graded      90      GOOD
```

Screen 18: View Submission System (after graded)

View My Submissions allows the student to view all assignments they have submitted. The screen displays the submission date, status (pending or graded), score, and feedback, and it updates automatically once the lecturer grades the submission.

Prepared By: EVELYN ANG

PART 4: DEVELOPMENT ACTIVITIES

Table 4.1: Meeting Log

Meeting Date	**Members Participate in the meeting**	**Activity**	**Task for each member**	**Task Achieved (Yes/No)**
10/1/2026	EVELYN ANG	- Identify system requirements - Discuss data structures needed	Stack for undo feature and creation history, sort for student ranking	Yes
	TEOH XIN YEE		Linked list, BST for search functionality and lecturer menu interface	Yes
	TOH SHEE THONG		Queue, grading functionality and student menu interface	Yes
13/1/2026	EVELYN ANG	- Distribute task in report	- Use case diagram and description - Pseudocode for system entry and student login - System prototype screen	Yes
	TEOH XIN YEE		- Synopsis and objective of project - Pseudocode for lecturer main menu - System prototype screen	Yes
	TOH SHEE THONG		- Class Diagram - Pseudocode for student main menu - System prototype screen	Yes
15/1/2026	EVELYN ANG	Finalize code and report	Check report formatting	Yes
	TEOH XIN YEE		Check and test all functionalities of system	Yes
	TOH SHEE THONG		Combine report	Yes

PART 5: APPENDIX

5.1 List of Data Files/File output

1. miniProject.cpp

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

// ============== STRUCTURES ==============

struct Assignment {
    string assignmentID;
    string title;
    string dueDate;
    string description;
    int maxScore;
    Assignment* next;
};

struct Submission {
    string submissionID;
    string studentID;
    string studentName;
    string assignmentID;
    string assignmentTitle;
    string submittedDate;
    string filePath;
    string status;
    int score;
    string feedback;
    Submission* next;
};

struct Student {
    string studentID;
    string name;
    string email;
    int totalScore;
    int submissionCount;
};

struct StackNode {
    Assignment* assignment;
    StackNode* next;
};

struct BSTNode {
    string key;
    Assignment* value;
```

```cpp
    BSTNode* left;
    BSTNode* right;
};

// ============== BST CLASS ==============
class BST {
private:
    BSTNode* root;

    void insert(BSTNode*& node, string key, Assignment* value) {
        if (!node)
            node = new BSTNode{key, value, nullptr, nullptr};
        else if (key < node->key)
            insert(node->left, key, value);
        else if (key > node->key)
            insert(node->right, key, value);
    }

    Assignment* search(BSTNode* node, string key) {
        if (!node) return nullptr;
        if (key == node->key) return node->value;
        else if (key < node->key) return search(node->left, key);
        else return search(node->right, key);
    }

    BSTNode* deleteNode(BSTNode* node, string key) {
        if (!node) return nullptr;

        if (key < node->key)
            node->left = deleteNode(node->left, key);
        else if (key > node->key)
            node->right = deleteNode(node->right, key);
        else {
            if (!node->left) {
                BSTNode* temp = node->right;
                delete node;
                return temp;
            } else if (!node->right) {
                BSTNode* temp = node->left;
                delete node;
                return temp;
            } else {
                BSTNode* succ = node->right;
                while (succ->left) succ = succ->left;
                node->key = succ->key;
                node->value = succ->value;
                node->right = deleteNode(node->right, succ->key);
            }
        }
        return node;
    }

public:
    BST() : root(nullptr) {}
    void insertItem(string key, Assignment* value) { insert(root, key, value); }
    Assignment* searchByTitle(string key) { return search(root, key); }
    void deleteItem(string key) { root = deleteNode(root, key); }
};

// ============== STACK CLASS (for Undo) ==============
class AssignmentStack {
private:
    StackNode* top;
    int size;

public:
    AssignmentStack() {
        top = nullptr;
        size = 0;
    }

    void push(Assignment* assignment) {
        StackNode* newNode = new StackNode();
        newNode->assignment = assignment;
        newNode->next = top;
        top = newNode;
        size++;
    }

    Assignment* pop() {
        if (isEmpty()) {
            return nullptr;
        }

        StackNode* temp = top;
        Assignment* assignment = top->assignment;
        top = top->next;
        delete temp;
        size--;
        return assignment;
    }

    bool isEmpty() {
        return top == nullptr;
    }

    int getSize() {
        return size;
    }

    void displayStack() {
        if (isEmpty()) {
            cout << "\nNo assignments in history.\n";
            return;
        }

        cout << "\n=== ASSIGNMENT CREATION HISTORY (Stack - LIFO) ===\n";
        cout << left
             << setw(12) << "ID"
             << setw(25) << "Title"
             << setw(15) << "Due Date" << endl;
        cout << string(52, '-') << endl;

        StackNode* temp = top;
        while (temp != nullptr) {
            cout << left
                 << setw(12) << temp->assignment->assignmentID
                 << setw(25) << temp->assignment->title
                 << setw(15) << temp->assignment->dueDate << endl;
            temp = temp->next;
        }
        cout << endl;
    }

    ~AssignmentStack() {
        while (!isEmpty()) {
            pop();
        }
    }
};

// ============== ASSIGNMENT LINKED LIST ==============
class AssignmentLinkedList {
private:
    Assignment* head;
    int count;
    BST bst;
    AssignmentStack stack;

    string generateID() {
        count++;
        return "ASSG" + to_string(1000 + count);
    }

public:
    AssignmentLinkedList() {
        head = nullptr;
        count = 0;
    }

    // Insert assignment
    void insertAssignment() {
        Assignment* newAssg = new Assignment();

        cout << "\n=== CREATE NEW ASSIGNMENT ===\n";
        newAssg->assignmentID = generateID();
        cout << "Assignment ID: " << newAssg->assignmentID << endl;

        cin.ignore();
        cout << "Title: ";
        getline(cin, newAssg->title);
        cout << "Description: ";
        getline(cin, newAssg->description);

        cout << "Due Date (YYYY-MM-DD): ";
        getline(cin, newAssg->dueDate);

        cout << "Max Score: ";
        cin >> newAssg->maxScore;

        newAssg->next = nullptr;


        bst.insertItem(newAssg->title, newAssg);

        stack.push(newAssg);

        if (head == nullptr || newAssg->dueDate < head->dueDate) {
            newAssg->next = head;
            head = newAssg;
        }
        else {
            Assignment* current = head;
            while (current->next != nullptr && current->next->dueDate <
newAssg->dueDate) {
                current = current->next;
            }
            newAssg->next = current->next;
            current->next = newAssg;
        }

        cout << "\n Assignment created successfully!\n";
        cout << "  (Use 'Undo Last Creation' to remove this assignment)\n";
    }

    // Stack operation
    void undoLastCreation() {
        if (stack.isEmpty()) {
            cout << "\nNo assignments to undo.\n";
            return;
        }

        Assignment* lastAssg = stack.pop();

        Assignment* temp = head;
        Assignment* prev = nullptr;

        while (temp != nullptr && temp != lastAssg) {
            prev = temp;
            temp = temp->next;
        }

        if (temp == nullptr) {
            cout << "Error: Assignment not found in list.\n";
            return;
        }

        if (prev == nullptr)
            head = temp->next;
        else
            prev->next = temp->next;

        bst.deleteItem(lastAssg->title);

        cout << "\n✓ Undo successful! Assignment '" << lastAssg->title << "'
removed.\n";

        delete lastAssg;
    }

    void displayCreationHistory() {
        stack.displayStack();
    }

    void displayAll() {
        if (head == nullptr) {
            cout << "\nNo assignments available.\n";
            return;
        }

        cout << "\n=== ALL ASSIGNMENTS (Linked List - Sorted by Due Date) ===\n";
        cout << left
             << setw(12) << "ID"
             << setw(25) << "Title"
             << setw(15) << "Due Date"
             << setw(10) << "Max Score" << endl;
        cout << string(62, '-') << endl;

        Assignment* temp = head;
        while (temp != nullptr) {
            cout << left
                 << setw(12) << temp->assignmentID
                 << setw(25) << temp->title
                 << setw(15) << temp->dueDate
                 << setw(10) << temp->maxScore << endl;
            temp = temp->next;
        }
        cout << endl;
    }

    // Search using BST
    Assignment* searchByTitle(string title) {
        return bst.searchByTitle(title);
    }

    // Search by ID
    Assignment* searchByID(string id) {
        Assignment* temp = head;
```

```cpp
        while (temp != nullptr) {
            if (temp->assignmentID == id) {
                return temp;
            }
            temp = temp->next;
        }
        return nullptr;
    }

    void displayDetails(Assignment* assg) {
        if (assg == nullptr) {
            cout << "Assignment not found.\n";
            return;
        }

        cout << "\n=== ASSIGNMENT DETAILS ===\n";
        cout << "ID: " << assg->assignmentID << endl;
        cout << "Title: " << assg->title << endl;
        cout << "Description: " << assg->description << endl;
        cout << "Due Date: " << assg->dueDate << endl;
        cout << "Max Score: " << assg->maxScore << endl;
    }

    ~AssignmentLinkedList() {
        Assignment* current = head;
        while (current != nullptr) {
            Assignment* temp = current;
            current = current->next;
            delete temp;
        }
    }
};

// ============== SUBMISSION QUEUE ==============
class SubmissionQueue {
private:
    Submission* front;
    Submission* rear;
    int count;

    string generateID() {
        count++;
        return "SUB" + to_string(10000 + count);
    }

    string getCurrentDate() {
        return "2025-01-11";
    }

public:
    SubmissionQueue() {
        front = nullptr;
        rear = nullptr;
        count = 0;
    }

    // Enqueue
    void enqueue(Student student, Assignment* assignment) {
        if (assignment == nullptr) {
            cout << "Invalid assignment.\n";
            return;
        }

        Submission* newSub = new Submission();

        newSub->submissionID = generateID();
        newSub->studentID = student.studentID;
        newSub->studentName = student.name;
        newSub->assignmentID = assignment->assignmentID;
        newSub->assignmentTitle = assignment->title;
        newSub->submittedDate = getCurrentDate();

        cout << "Enter file path/description: ";
        cin.ignore();
        getline(cin, newSub->filePath);

        newSub->status = "Pending";
        newSub->score = 0;
        newSub->feedback = "Not graded yet";
        newSub->next = nullptr;

        // Queue operation (FIFO)
        if (rear == nullptr) {
            front = rear = newSub;
        } else {
            rear->next = newSub;
            rear = newSub;
        }

        cout << "\n✓ Assignment submitted successfully!\n";
        cout << "Submission ID: " << newSub->submissionID << endl;
    }

    bool isDuplicate(string studentID, string assignmentID) {
        Submission* temp = front;
        while (temp != nullptr) {
            if (temp->studentID == studentID && temp->assignmentID == assignmentID)
{
                return true;
            }
            temp = temp->next;
        }
        return false;
    }

    void displayPending() {
        if (front == nullptr) {
            cout << "\nNo pending submissions.\n";
```

```cpp
            return;
        }

        cout << "\n=== PENDING SUBMISSIONS (Queue - FIFO Order) ===\n";
        cout << left
             << setw(12) << "Sub ID"
             << setw(20) << "Student"
             << setw(25) << "Assignment"
             << setw(15) << "Date" << endl;
        cout << string(72, '-') << endl;

        Submission* temp = front;
        while (temp != nullptr) {
            if (temp->status == "Pending") {
                cout << left
                     << setw(12) << temp->submissionID
                     << setw(20) << temp->studentName
                     << setw(25) << temp->assignmentTitle
                     << setw(15) << temp->submittedDate << endl;
            }
            temp = temp->next;
        }
        cout << endl;
    }

    void displayByStudent(string studentID) {
        Submission* temp = front;
        bool found = false;

        cout << "\n=== MY SUBMISSIONS ===\n";
        cout << left
             << setw(25) << "Assignment"
             << setw(15) << "Date"
             << setw(12) << "Status"
             << setw(8) << "Score"
             << setw(20) << "Feedback" << endl;
        cout << string(80, '-') << endl;

        while (temp != nullptr) {
            if (temp->studentID == studentID) {
                cout << left
                     << setw(25) << temp->assignmentTitle
                     << setw(15) << temp->submittedDate
                     << setw(12) << temp->status
                     << setw(8) << temp->score
                     << setw(20) << temp->feedback << endl;
                found = true;
            }
            temp = temp->next;
        }

        if (!found) {
            cout << "No submissions found.\n";
        }
        cout << endl;
    }

    void gradeSubmission(string submissionID) {
        Submission* temp = front;

        while (temp != nullptr) {
            if (temp->submissionID == submissionID) {
                if (temp->status == "Graded") {
                    cout << "This submission is already graded.\n";
                    return;
                }

                cout << "\n=== GRADING SUBMISSION ===\n";
                cout << "Student: " << temp->studentName << " (" << temp->studentID
<< ")\n";
                cout << "Assignment: " << temp->assignmentTitle << endl;
                cout << "File: " << temp->filePath << endl;

                cout << "\nEnter Score: ";
                cin >> temp->score;
                cin.ignore();

                cout << "Enter Feedback: ";
                getline(cin, temp->feedback);

                temp->status = "Graded";

                cout << "\n✓ Submission graded successfully!\n";
                return;
            }
            temp = temp->next;
        }

        cout << "Submission not found.\n";
    }

    //Sorting algorithm
    void displayStudentRankings() {
        if (front == nullptr) {
            cout << "\nNo submissions to rank.\n";
            return;
        }

        Student students[100];
        int studentCount = 0;

        Submission* temp = front;
        while (temp != nullptr) {
            if (temp->status == "Graded") {
                int index = -1;
                for (int i = 0; i < studentCount; i++) {
                    if (students[i].studentID == temp->studentID) {
                        index = i;
                        break;
                    }
                }

                if (index == -1) {
                    students[studentCount].studentID = temp->studentID;
                    students[studentCount].name = temp->studentName;
                    students[studentCount].totalScore = temp->score;
                    students[studentCount].submissionCount = 1;
                    studentCount++;
                } else {
                    students[index].totalScore += temp->score;
                    students[index].submissionCount++;
                }
            }
            temp = temp->next;
        }

        if (studentCount == 0) {
            cout << "\nNo graded submissions yet.\n";
            return;
        }

        // Bubble Sort
        for (int i = 0; i < studentCount - 1; i++) {
            for (int j = 0; j < studentCount - i - 1; j++) {
                if (students[j].totalScore < students[j + 1].totalScore) {
                    Student tempStudent = students[j];
                    students[j] = students[j + 1];
                    students[j + 1] = tempStudent;
                }
            }
        }

        cout << "\n=== STUDENT RANKINGS (Sorted by Total Score) ===\n";
        cout << left
             << setw(6) << "Rank"
             << setw(12) << "Student ID"
             << setw(20) << "Name"
             << setw(12) << "Total Score"
             << setw(12) << "Assignments"
             << setw(10) << "Average" << endl;
        cout << string(72, '-') << endl;

        for (int i = 0; i < studentCount; i++) {
            float average = (float)students[i].totalScore /
students[i].submissionCount;
            cout << left
                 << setw(6) << (i + 1)
                 << setw(12) << students[i].studentID
                 << setw(20) << students[i].name
                 << setw(12) << students[i].totalScore
                 << setw(12) << students[i].submissionCount
                 << fixed << setprecision(2) << average << endl;
        }
        cout << endl;
    }

    ~SubmissionQueue() {
        Submission* current = front;
        while (current != nullptr) {
            Submission* temp = current;
            current = current->next;
            delete temp;
        }
    }
};

// ============== GLOBAL OBJECTS ==============
AssignmentLinkedList assignmentList;
SubmissionQueue submissionQueue;
Student currentStudent;

// ============== MENU FUNCTIONS ==============

void studentLogin() {
    cout << "\n=== STUDENT LOGIN ===\n";
    cout << "Enter Student ID: ";
    cin >> currentStudent.studentID;
    cin.ignore();
    cout << "Enter Name: ";
    getline(cin, currentStudent.name);
    cout << "Enter Email: ";
    getline(cin, currentStudent.email);

    cout << "\nWelcome, " << currentStudent.name << "!\n";
}

void studentMenu() {
    studentLogin();

    int choice;
    string id, title;

    do {
        cout<<endl;
        cout << "_____________________________________\n";
        cout << "|            STUDENT MENU            | \n";
        cout << "|____________________________________|\n";
        cout << "| 1. View All Assignments            |\n";
        cout << "| 2. Search Assignment (BST)         |\n";
        cout << "| 3. Submit Assignment (Queue)       |\n";
        cout << "| 4. View My Submissions             |\n";
        cout << "| 5. View Rankings (Sorting)         |\n";
        cout << "| 6. Logout                          |\n";
        cout << "|____________________________________|\n";
        cout << "Choice: ";
        cin >> choice;
        system("cls");

        switch (choice) {
            case 1:
                assignmentList.displayAll();
                break;

            case 2:
                cout << "\nSearch by:\n1. Title\n2. ID\nChoice: ";
                {
                    int searchChoice;
                    cin >> searchChoice;
                    cin.ignore();

                    if (searchChoice == 1) {
                        cout << "Enter Assignment Title: ";
                        getline(cin, title);

assignmentList.displayDetails(assignmentList.searchByTitle(title));
                    } else {
                        cout << "Enter Assignment ID: ";
                        getline(cin, id);

assignmentList.displayDetails(assignmentList.searchByID(id));
                    }
                }
                break;

            case 3: {
                assignmentList.displayAll();
                cout << "Enter Assignment ID to submit: ";
                cin >> id;

                Assignment* assg = assignmentList.searchByID(id);
                if (assg == nullptr) {
                    cout << "Assignment not found.\n";
                    break;
                }

                if (submissionQueue.isDuplicate(currentStudent.studentID, id)) {
                    cout << "You already submitted this assignment.\n";
                    break;
                }

                submissionQueue.enqueue(currentStudent, assg);
                break;
            }

            case 4:
                submissionQueue.displayByStudent(currentStudent.studentID);
                break;

            case 5:
                submissionQueue.displayStudentRankings();
```

```cpp
                break;

            case 6:
                cout << "Logging out...\n";
                break;

            default:
                cout << "Invalid choice.\n";
        }
    } while (choice != 6);
}

void lecturerMenu() {
    int choice;
    string id;

    do {
        cout<<endl;
        cout << "______________________________________\n";
        cout << "|     -------LECTURER MENU------      |\n";
        cout << "|_____________________________________|\n";
        cout << "| 1. Create Assignment                |\n";
        cout << "| 2. Undo Last Creation               |\n";
        cout << "| 3. View Creation History            |\n";
        cout << "| 4. View All Assignments             |\n";
        cout << "| 5. Search Assignment                |\n";
        cout << "| 6. View Pending Submissions         |\n";
        cout << "| 7. Grade Submission                 |\n";
        cout << "| 8. View Student Rankings            |\n";
        cout << "| 9. Logout                           |\n";
        cout << "|_____________________________________|\n";
        cout << "Choice: ";
        cin >> choice;
        system("cls");

        switch (choice) {
            case 1:
                assignmentList.insertAssignment();
                break;

            case 2:
                assignmentList.undoLastCreation();
                break;

            case 3:
                assignmentList.displayCreationHistory();
                break;

            case 4:
                assignmentList.displayAll();
                break;

            case 5: {
                string title;
                cout << "Enter Assignment Title: ";
                cin.ignore();
                getline(cin, title);
                assignmentList.displayDetails(assignmentList.searchByTitle(title));
                break;
            }

            case 6:
                submissionQueue.displayPending();
                break;

            case 7:
                submissionQueue.displayPending();
                cout << "Enter Submission ID to grade: ";
                cin >> id;
                submissionQueue.gradeSubmission(id);
                break;

            case 8:
                submissionQueue.displayStudentRankings();
                break;

            case 9:
                cout << "Logging out...\n";
                break;

            default:
                cout << "Invalid choice.\n";
        }
    } while (choice != 9);
}

int main() {
    int choice;

    do {
        cout << "________________________________________\n";
        cout << "|     E-LEARNING ASSIGNMENT SYSTEM     |\n";
        cout << "|______________________________________|\n";
        cout << "| 1. Student Login                     |\n";
        cout << "| 2. Lecturer Login                    |\n";
        cout << "| 0. Exit System                       |\n";
        cout << "|______________________________________|\n";
        cout << "Choice: ";
        cin >> choice;
        system("cls");

        switch (choice) {
            case 1:
                studentMenu();
                break;

            case 2:
                lecturerMenu();
                break;

            case 0:
                cout << "\n=== Thank you for using the system! ===\n";
                break;

            default:
                cout << "Invalid choice.\n";
        }
    } while (choice != 0);

    return 0;
}
```