**Logo

Description automatically generated**

**FACULTY OF COMPUTING**

**UNIVERSITI TEKNOLOGI MALAYSIA**

**DATA STRUCTRUE & ALGORITHM**  
(MECS0023)

SEMESTER 2 2023/2024

Mini Project Documentation

**Student Course Management System**

**By**

1. **Pang Siew La – MECS235008**
2. **Ng Chi Yuan – MECS253003**
3. **Ooi Soo Han – MECS235010**
4. **Tan Kai Siang – MECS232008**

**Section 52**

**Lecturer:**

**Pang Yee Yong**

**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 |  |

**MECS0023 DSA - MINI PROJECT SPECIFICATION**

**PART 1: INTRODUCTION**

* 1. **Synopsis Project**

The project aims to develop a text-based application in C++ that manages course and student registrations using Link List and queues.

**1.2 Objective of the project**

The objective is to implement a system that efficiently handles student enrolments and course registrations, while utilizing fundamental data structures such as linked lists and queues.

**PART 2: SYSTEM ANALYSIS AND DESIGN (USE CASE, FLOWCHART AND CLASS DIAGRAM)**

**2.1 System Requirements**

The system must allow users to:

1. Register a student.
2. Register a course (with a limited number of slots).
3. Enrol a student in a course (if slots are available, otherwise add to a waiting list).
4. De-enrol a student (if there are students on the waiting list, enrol the first student in the queue).
5. Display available courses.
6. Display registered students.

The system has 6 use cases: -

|  |  |
| --- | --- |
| **Use Case** | **Purpose** |
| Register a student | Register new student name |
| Register a course | Register new course |
| Enrol a student in a course | Enrol student in a course subject |
| De-enrol a student from a course | Remove a student from enrollment in a course |
| Display available courses | Checkthe courses that are available |
| Display registered students | Check the registered students’ information |

**2.2 System Design**

The system will be menu-driven, offering options for each functionality mentioned above.

The class diagram represents the structure of the system in terms of classes, their attributes, methods, and relationships. It illustrates how different entities such as students, courses, and registrations are modelled in the system.The block diagram belowdivide the flow of processes into register a student, register a course, enrol a student into a course, de-enrol a student from a course, display available courses, and display registered students.

1. Student course management system

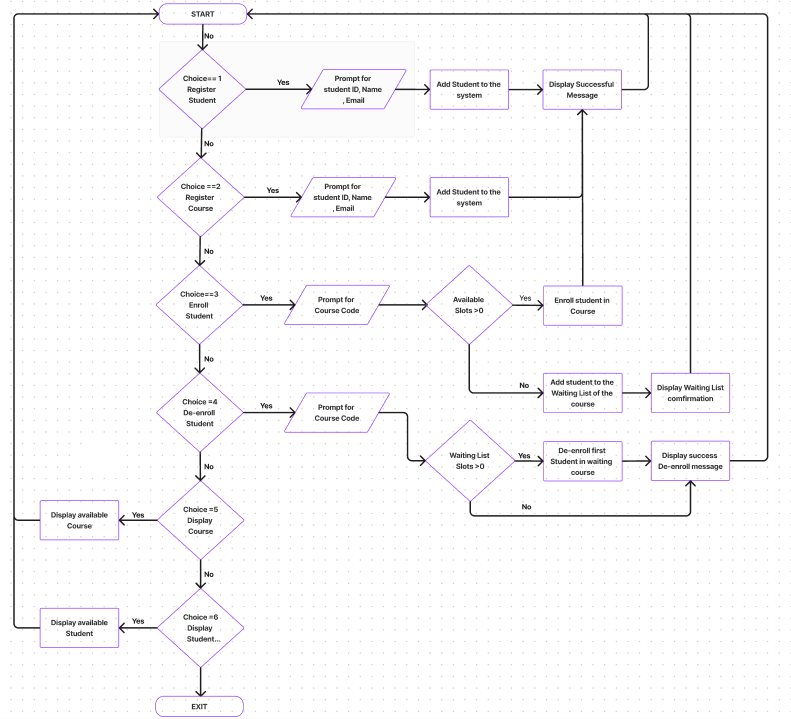


Figure 2.1: Flow chart: Student course management system

1. Register a student

Figure 2.2: Flow chart: Register a student

1. Register a course



Figure 2.3: Flow chart: Register a course

1. Enrol a student in a course

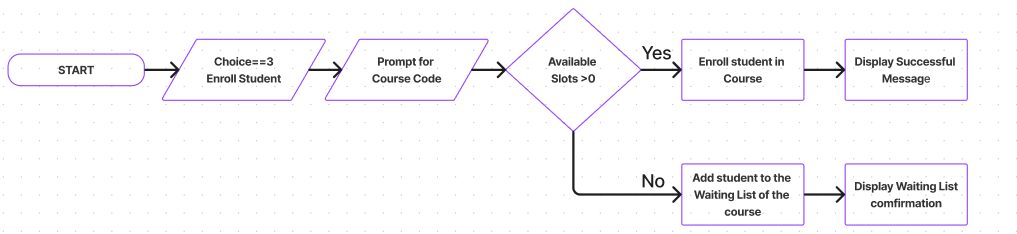


Figure 2.4: Flow chart: Enrol a student in a course

1. De-enrol a student from a course

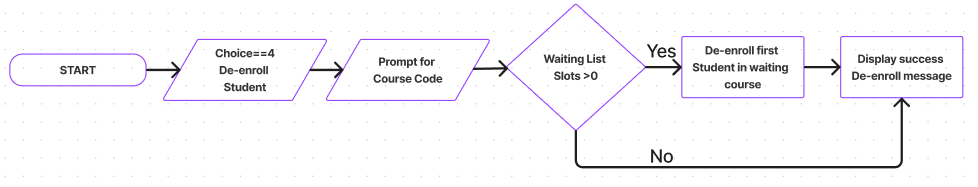


Figure 2.5: Flow chart: De-enrol a student from a course

1. Display available courses

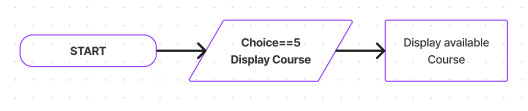


Figure 2.6: Flow chart: Display available course

1. Display registered students

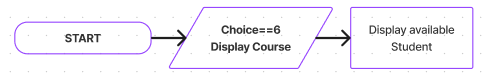


Figure 2.7: Flow chart: Display registered student

**PART 3: SYSTEM PROTOTYPE**

A prototype will be developed to demonstrate the core functionalities of the system.

**A screen shot of a computer

Description automatically generated**

**Screen 1: Menu Item**

Screen 1: The user must insert an integer value in the range 1-6. If the user enter other number, the system will prompt error message and the screen is displayed again

Prepared By: Siew La

A screen shot of a computer

Description automatically generated

**Screen 2: Register A Student**

Screen 2: The user will be prompted to enter 3 input, student ID (int), student email (string) and student name (string)

Prepared By: Siew La

A screenshot of a computer

Description automatically generated

**Screen 3: Register A Course**

Screen 3: The user will be prompted to enter 4 input, course name (string), course code (string), course slots (int), and credit hours (int).

Prepared By: Siew La

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Screen 4: Enrol or de-enrol student**

Screen 4: The list of courses will be shown to user, user to input the course code

Prepared By: Siew La

A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

**Screen 5: Enrol or de-enrol student Menu**

Screen 5: The user must insert an integer value in the range 1-4. If the user enter other number, the system will prompt error message and the screen is displayed again

Prepared By: Siew La

A screen shot of a computer

Description automatically generated

**Screen 6: List of Available Course**

A screenshot of a computer

Description automatically generated

**Screen 7: List of All Registered Students**

**PART 4: UML Diagram**

These are the data structure for the application

**A group of pink and white tags

Description automatically generated**

The UML diagram describes a system where courses and students are managed using linked lists (CourseList and StudentList). Each course (Course) and student (Student) is represented by objects with specific attributes, and linked list nodes (CourseNode and StudentNode) are used to organize and manipulate these objects. This structure supports operations typical of educational systems, such as enrollment, waitlisting, and dynamic management of course capacities.

Component Explanation:

Course

courseName: Name of the course (type: string).

courseCode: Code identifier for the course (type: string).

courseSlots: Number of available slots for students in the course (type: int).

courseCreditHours: Number of credit hours the course is worth (type: int).

students: List of students enrolled in the course (type: StudentList).

waitlist: List of students on the waitlist for the course (type: StudentList).

Student

matrixNumber: Unique identifier for the student (type: int).

name: Name of the student (type: string).

studentEmail: Email address of the student (type: string).

CourseNode

course: Reference to a Course object (type: Course).

next: Reference to the next CourseNode in a linked list structure (type: CourseNode\*).

StudentNode

student: Reference to a Student object (type: Student).

next: Reference to the next StudentNode in a linked list structure (type: StudentNode\*).

CourseList

head: Reference to the first node (CourseNode) in the linked list of courses.

tail: Reference to the last node (CourseNode) in the linked list of courses.

size: Number of courses in the list (type: int).

StudentList

head: Reference to the first node (StudentNode) in the linked list of students.

tail: Reference to the last node (StudentNode) in the linked list of students.

size: Number of students in the list (type: int).

Relationship Explanation:

Course and Student:

* Course represents a course offered at the educational institution. It has attributes describing its identity (courseName, courseCode), capacity (courseSlots), academic value (courseCreditHours), and lists (students and waitlist) for managing enrolled and waitlisted students.
* Student represents an individual enrolled at the institution, identified uniquely by matrixNumber. It includes basic personal information such as name and studentEmail.

CourseNode and StudentNode:

* CourseNode and StudentNode are helper classes used for implementing linked lists of courses and students, respectively. Each node (CourseNode or StudentNode) contains a reference to either a Course or a Student object (course or student), and a pointer (next) to the next node in the list.

CourseList and StudentList:

* CourseList (CourseLst) and StudentList (StudentList) are collections of courses and students, implemented as linked lists. They maintain references (head, tail) to the first and last nodes of their respective lists, facilitating operations such as insertion, deletion, and traversal. The size attribute keeps track of the number of elements in each list.

Course - Student Relationship:

* Each Course object has associations (students and waitlist) with StudentList objects. This allows for efficient management of enrolled students and those on the waitlist for each course.

Node Relationships:

* CourseNode and StudentNode facilitate the implementation of linked lists for courses and students respectively, allowing for dynamic management of collections without fixed size constraints.

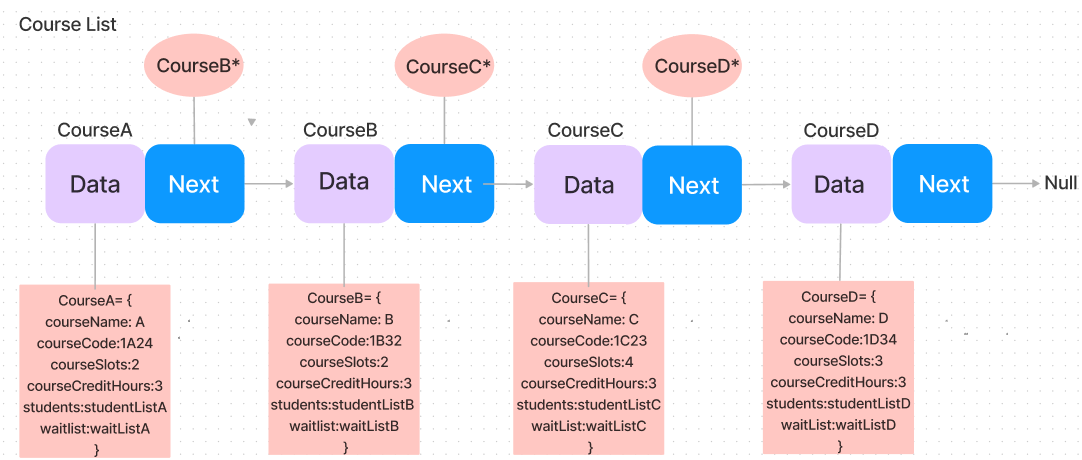
The linked list is used in the course list and student list. Whereas the queue is used in the course waitlist list.

1. Student List (Linked List)

**A diagram of a student

Description automatically generated**

1. Course List (Linked List)

****

**PART 5: DEVELOPMENT ACTIVITIES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Meeting Date | Members Participate in Meeting | Activity | Task for each member | Task Achieved (Yes/No) |
| 5 June 2024 | Siew La | Idea Generation | Get the idea approval from Prof | Yes |
| 6 June 2024 | Siew La | Do Flow Generation | Complete the overall workflow | Yes |
| 8 June 2024 | Siew La | Basic File Structure of the Code | Complete the basic file structure | Yes |
| 9 June 2024 | Siew La | Complete the MVP | Complete the MVP | Yes |
| 9 June 2024 | Siew La | Draft the initial report | Complete Initial Report Structure | Yes |
| 14 June 2024 | Chi Yuan | Refactor Code for V2 | Refactor Code to cater for error handling | Yes |
| 17 June 2024 | Soo Han | Add Individual Flow chart | Add flow chart to every flow | Yes |
| 18 June 2024 | Kai Siang | Refactor Code | Refactor Code to accept lower case for course | Yes |
| 23 June 2024 | Siew La | UML Diagram | Draw UML Diagram | Yes |
| 23 June 2024 | Siew La | Linked List for StudentList | Draw linked list for StudentList | Yes |
| 23 June 2024 | Soo Han | Linked List for CourseList | Draw linked list for CourseList | Yes |
| 24 Jun 2024 | Kai Siang | Explain UML Diagram | Elaborate on the UML Diagram | Yes |