# CAFETERIA MANAGEMENT SYSTEM

## MINI – PROJECT

1. **Title of project:**
   Cafeteria Management System

2. **Group Details:**
   A2 Batch

   1426 – Srushti Jadhav
   1436 – Rajnandini Kokadwar
   1438 – Utkarsha Kumbhar
   1444 – Janhavi Mishra

3. **Explain in brief about the project.**
   As the name suggests, 'Cafeteria management system' it is a digitalized system where we are implementing Circular queue using Linked List as our main data structure.
   In the beginning, we have given 2 choices – Customer and Manager.

   **In Customer:**

   **a.** First, we display menu
   **b.** Place orders of customers
   **c.** Display order of customer
   d. Give extra discount is given for revisiting and if total cost is above certain limit.
   **e.** Deliver the order and display the bill.
   f. We have also added features like Reading Zone, Gaming Zone and Musical Zone with their additional charges.
   g. Cancel order only if order is not ready
   h. Check review about our café.

**In Manager:**

a. Create a new combo pack of dishes based on frequency of orders.
b. Ask customer's feedback about the café.
c. Update prices and dishes in the menu.

**Problem Statement:**
Implement Cafeteria Management System to place, display, deliver and cancel orders of customers using appropriate data structure. Also, update and display menu efficiently.

**4. List all data structures used. Justify why did you select theses data structure?**

Data Structures used:
a. Circular Queue
b. ArrayList
c. HashMap

**a. Circular Queue:**
**THEORY:**
Circular Queue is similar to linear queue i.e. FIFO(First In First Out) order is followed except that last element of queue is connected to first element of queue forming a circle.
In, Circular queue elements can be inserted easily if there are vacant locations until it is not fully occupied. Whereas in linear queue insertion is not possible once the rear reaches the last index, even if there are empty locations present in the queue.
Circular Queue can be implemented in  2 ways – Array and Linked List.

**IMPLEMENTATION IN OUR CODE:**
In this code, we have used circular queue to maintain records of customers which includes: name, address and phone number. Additionally, order of customer. Because of circular queue enqueue and dequeue operation of customer data takes place easily.

We have implemented circular queue using linked list because Linked list has dynamic memory allocation. Therefore, overflow can never occur unless memory is actually full.

**b. ArrayList:**
**THEORY:**
ArrayList implements the List interface so we can use all the methods of the List interface here.

**IMPLEMENTATION IN OUR CODE:**

In this code, we have used ArrayList to store orders of each customer, ID of customer, ratings etc. Using ArrayList we can directly get items. Also, we can calculate frequency of every item in the list.

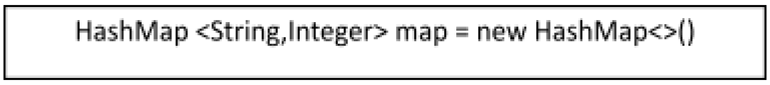
ArrayList uses a dynamic array for storing the elements. It is like an array, but there is no size limit. We can add or remove elements anytime. So, it is much more flexible than the traditional array.

c. **HashMap:**

**THEORY:**

HashMap allows us to store information in the form of key and pair

Syntax of HashMap:



```
HashMap <String,Integer> map = new HashMap<>()
```

All keys in HashMap are unique. Map is useful if you want to search, update or delete elements on the basis of key.

**IMPLEMENTATION IN OUR CODE:**

In this code, we have used HashMap for storing menu information where keys are dish name and values are price of that particular dish.

Using HashMap we can update and print menu easily. It is also useful to get prices of each dish effortlessly. As well as, we can iterate over menu.So, this is appropriate data structure to store menu.

**TIME COMPLEXITY:**

a. **CIRCULAR QUEUE:**

| | |
|---|---|
| display() | O(n) |
| enqueue() | O(1) |
| dequeue() | O(1) |
| cancel() | O(n) |

b. **HASHMAP:**

| | |
|---|---|
| put() | O(1) |
| get() | O(n) |

c. **ArrayList:**

| | |
|---|---|
| add() | O(1) |

| | |
|---|---|
| get() | O(1) |
| remove() | O(n) |