



School of Computing

Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITMA

(SECJ2013-06)

Lecturer: Dr. Johanna Ahmad

Mini Project Report

Food Ordering System

By,

ALAA ALRHMAN MOHAMMED RAWEH AL-SHAIBANI (A18CS4037)

OMAR HAMED ABDELLATIF IBRAHIM (A18CS4061)

VIMALRAJ A/L SIVARAJOO (A19EC0213)

HABIBA IBRAHIM ABDELRAHIM ELGAMMAL (A18CS0303)

Table of Contents

Introduction	3
Objective	4
System Analysis & Design	6
System Prototype	13
Development Activities	20
Source Code	23

1.0 Introduction

Consistent with the Ministry of Health Malaysia (MOH), there have been over 38 thousand active cases of the Coronavirus currently in Malaysia and has resulted in 605 deaths as of 18 January 2021. The number of new cases has been snowballing at an exponential rate since October of 2020 and as of early November 2020, the number of new cases has consistently been over 1000 every single day. Needless to say, the greatest number of cases has reached 4029 on 16 January 2021 which is deeply worrying.

Due to this fact, online ordering has become one of the most important and useful services that have been increasingly used, especially in the field of ordering food as each restaurant has only a limited number of customers to dine in. If you need to get a meal, you have to go through a time-consuming process especially when you go to a place without having a reservation or the restaurant cannot serve all your family members in one table. Facing such difficulties made online food ordering a good choice for customers to get their favorite meals. As a result of that we choose to develop a food ordering system as our assignment objective.

The developed food ordering system will provide the customer with the option of selecting the meal by giving him the chance to order a meal by showing him the ordered list that contains all options. It also will give the staff the ability to update the menu by adding new meals and put them in the right alphabetical order as well as the ability of searching for food items and other functionalities that will be further discussed.

1.1 Objective

- To develop a system that will surely satisfy the customer service.
- Effective modern day food ordering system.
- Businesslike and well organized in a very straightforward manner .
- Consisting basic/necessary service features.
- To design a system that is able to accommodate huge amounts of orders at a time.
- accuracy and reliability.
- To evaluate its performance and acceptability in terms of security, user friendliness.
- minimize the time of ordering.
- To improve the communication between the client and the server and
- Maintain distance.
- Promote a safe environment.
- Minimum contact.
- Manual listing of orders by the waiters/waitresses may result to slow response in customer service. Hence, if the restaurant uses the proposed system, manipulation of orders to the customers is so easy and quick.
- Ensure customer satisfaction.

1.2 Explanation of the data structure and techniques used:

Queue is essentially a list of data items, commands, etc. stored in such a way that it can be retrieved in a certain order, usually the order of insertion. Queue is open at both ends. One end is always used to insert data (enqueue) and the other end is used to delete data (dequeue). Queue follows the methodology of first-in-first-out (FIFO).

In the linked queue, two pointers are stored in the memory the front pointer and the rear pointer. The front pointer contains the address of the starting element of the queue, while the back pointer contains the address of the last element of the queue.

Insertion and deletion shall be performed at the rear and front ends respectively. If both front and back are NULL, it indicates that the queue is empty.

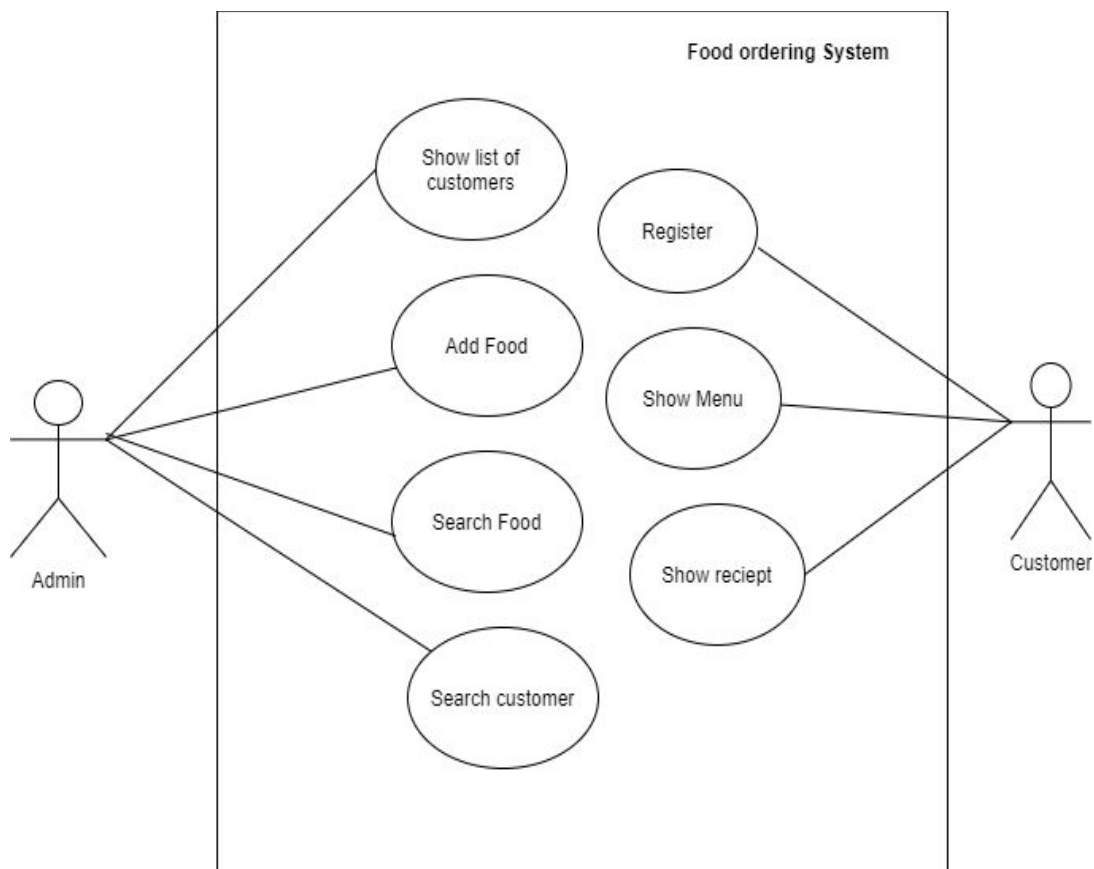
In our project we have used queue for both food list and customer list. Our system offers the customers to have their own Id by logging in and later order food from the menu. So, the customer data are stored by the help of queue.

Admin of the system can update the food items based on the availability. So, once they add a food item, it is stored in the queue by the help of enqueue() function and later the admin can delete a food item with the help of dequeue() function process

2.0 System Analysis & Design

2.1 System Requirements

2.1.1 Use case diagram



2.1.2 Use Case Description for food ordering system

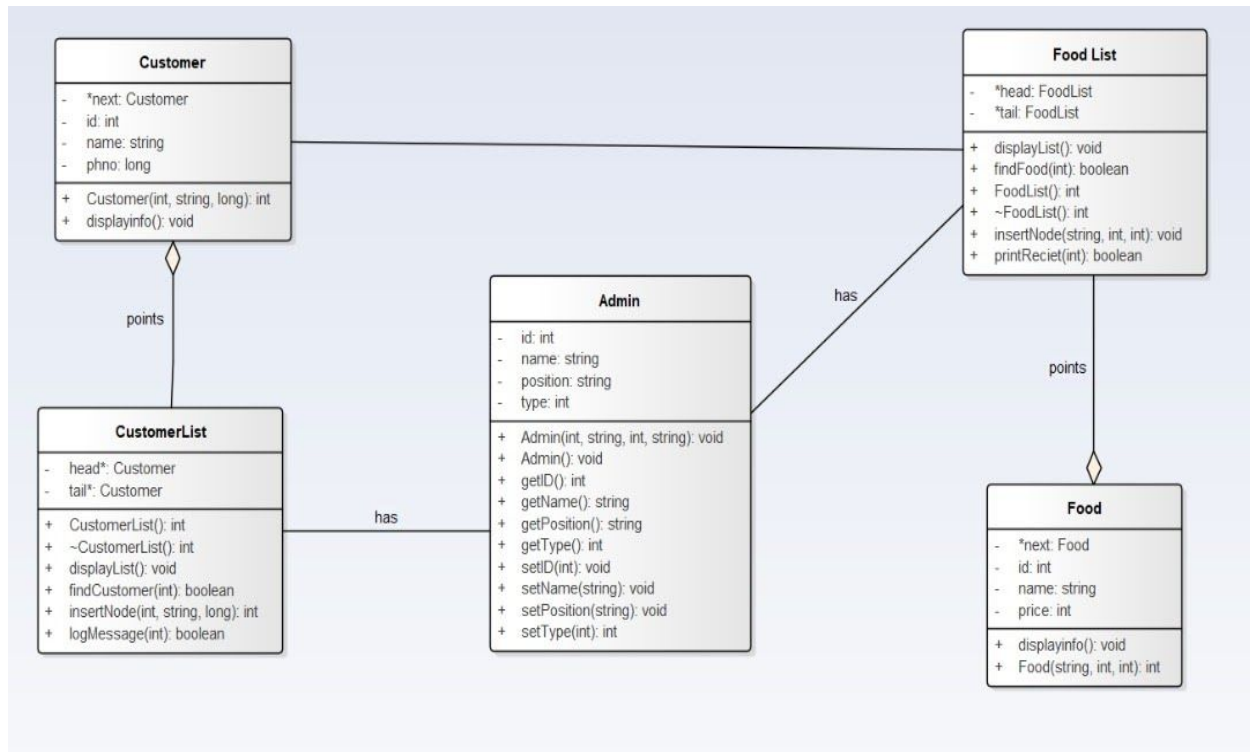
Actor	Task
Admin	<p>The admin in our food ordering system is the person who is in charge of controlling the system. He is responsible for adding food items in the menu.</p> <p>The admin is also able to search a specific customer in the database, showing the list of customers, and search for food item</p>
Customer	<p>The customer is the user who is able to register in the system, view all the food menu, as well as request to show the receipt of his order.</p>

2.1.3 Detail Description for each Use Case

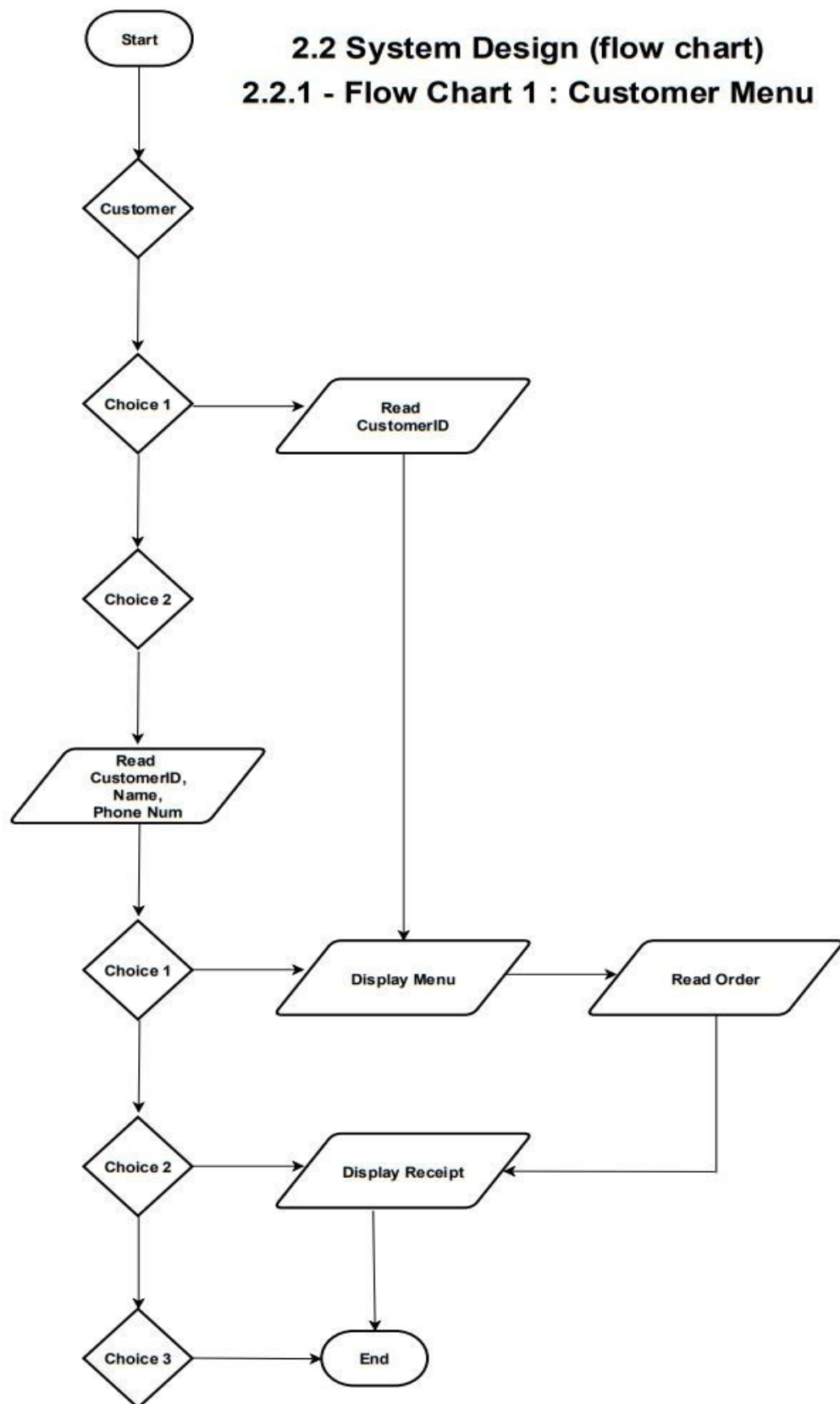
Use case	Description
Show list of customers	<p>It shows the record of all the customers that have been registered in the system</p> <p>Accessed by: admin</p>
Add Food	<p>It is the function that enables the admin to add new food items to the menu of the restaurant by adding an id, name, and price for each added item.</p> <p>Accessed by :admin</p>

Search Food	<p>This use case allows the admin to search for specific food items from the menu by keying in the id number of the item.</p> <p>Accessed by :admin</p>
Search customer	<p>The admin is allowed to access the registered customers in the system and search for a certain person using his id.</p> <p>Accessed by :admin</p>
Register	<p>The system allows new customers to register themselves in the system .</p> <p>Accessed by:New customers.</p>
Show menu	<p>Users can view all the menu items added in the system and choose their order from it.</p> <p>Accessed by:customer</p>
Show Receipt	<p>After the customer chooses items from the food menu , he is allowed to request to view the receipt.</p> <p>Accessed by:customer</p>

2.2 Class Diagram

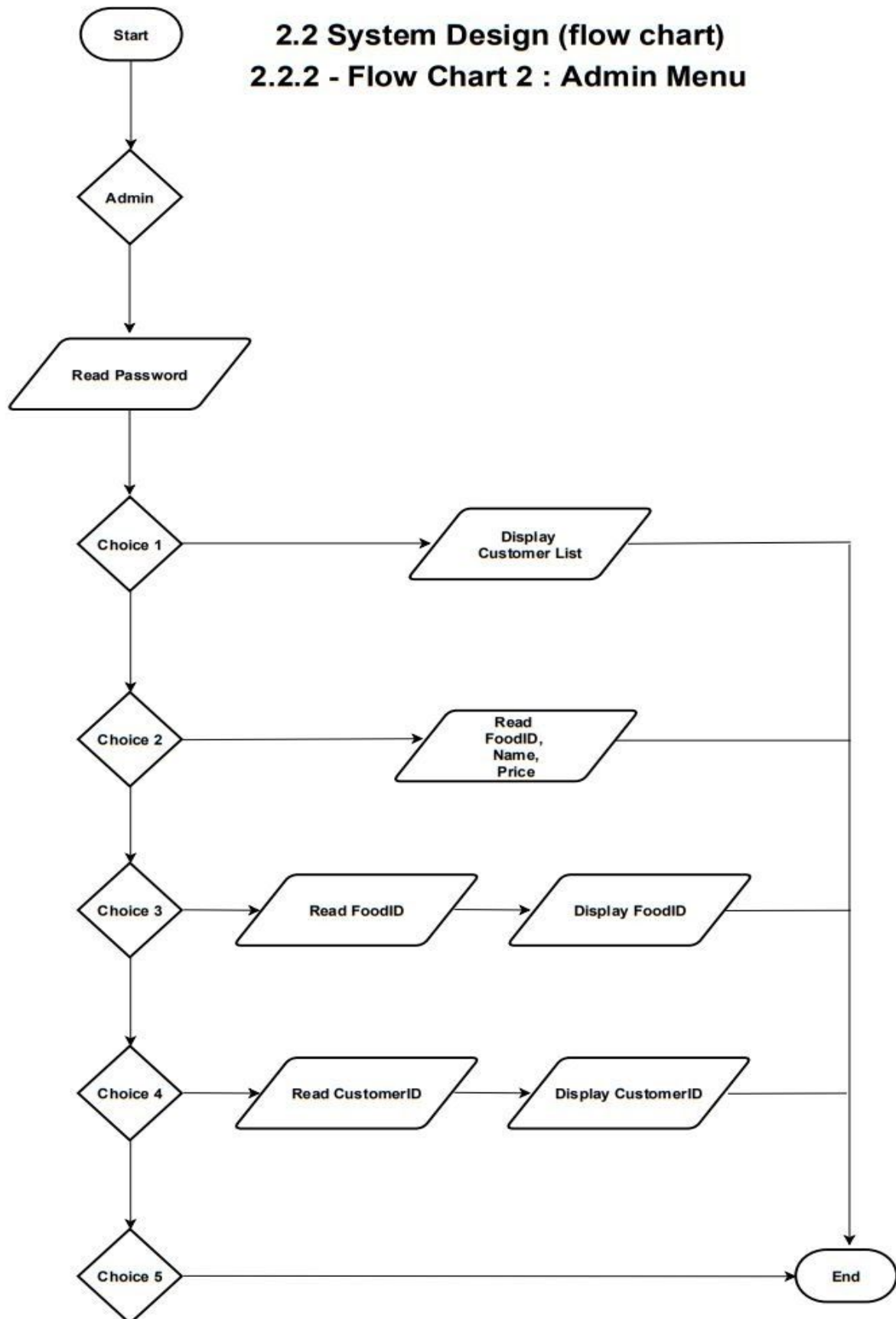


2.3 Flow chart



2.2 System Design (flow chart)

2.2.2 - Flow Chart 2 : Admin Menu



2.4 Techniques used

Explanation of the data structure and techniques used

Queue is essentially a list of data items, commands, etc. stored in such a way that it can be retrieved in a certain order, usually the order of insertion. Queue is open at both ends. One end is always used to insert data (enqueue) and the other end is used to delete data (dequeue). Queue follows the methodology of first-in-first-out (FIFO).

In the linked queue, two pointers are stored in the memory the front pointer and the rear pointer. The front pointer contains the address of the starting element of the queue, while the back pointer contains the address of the last element of the queue.

Insertion and deletion shall be performed at the rear and front ends respectively. If both front and back are NULL, it indicates that the queue is empty.

In our project we have used queue for both food list and customer list. Our system offers the customers to have their own Id by logging in and later order food from the menu. So, the customer data are stored by the help of queue.

Admin of the system can update the food items based on the availability. So, once they add a food item, it is stored in the queue by the help of enqueue() function and later the admin can delete a food item with the help of dequeue() function process

3.0 System Prototype

```
Are you a :  
1. Customer  
2. Admin  
_
```

When the user opens the system there will be two choices to choose from .

Select whether you are a customer or an admin.

```
Welcome to our food ordering system !  
1. Existing Customer  
2. New Customer  
Please make a selection to continue :  
_
```

```
Welcome to our food ordering system !  
1. Existing Customer  
2. New Customer  
Please make a selection to continue :2  
Enter id :10  
Enter name : omar  
Enter Phone Number :050505050_  
_
```

If the user was a customer he must choose whether he is a new or existing customer.

If he was a new customer then he has to input id, name and phone number so the system can take record and add him to the database. But if the user was an existing customer he may proceed with the order.

```
Welcome! Please select your choice
1. Show Menu
2. Show Receipt
3. Exit
```

After that the customer must select the action that he would like to take.

```
Welcome! Please select your choice
1. Show Menu
2. Show Receipt
3. Exit
1

*****MENU*****
ID      Name                      Price
3       Hot Dog                    8
6       Spicy Chicken Mcdeluxe    17
7       Roti chinai                20
4       McNuggets                  20
8       pasta 5003
8       pasta 5003
Choose your Orders (Press 0 once you are done)3
Choose your Orders (Press 0 once you are done)6
Choose your Orders (Press 0 once you are done)4
Choose your Orders (Press 0 once you are done)0
Thank you for shopping with us!
Press any key and then press enter to continue
```

If the customer selected 1 which is show menu, a menu will pop up containing all the food list that the restaurant offers. Then he should choose his order by keying in the id of the preferred food.

```
Welcome! Please select your choice
1. Show Menu
2. Show Receipt
3. Exit
2
```

If the customer wants to view the receipt he can choose number 2 .

```
Thank you for using us as your preferred platform!!
Your order is :

ID      Name                Price
3       Hot Dog              8
4       McNuggets             20
6       Spicy Chicken Mcdeluxe 17
Press any key to continue . . . █
```

A list of the customer's order will be shown with their prices respectively .

```
Are you a :
1. Customer
2. Admin
2
```

```
Enter Password : password_
```

If the user was an admin he must key in the password to enter the admin view.

```
Welcome! Please select your choice
1. Show list of Customers
2. Add Food
3. Search food
4. Search Customer
5. Exit
1
```

The admin is given different choices from the customer where he can choose to view the list of the registered customers , add new food to the menu, search for specific food item , search for a customer, or he may exit the system.

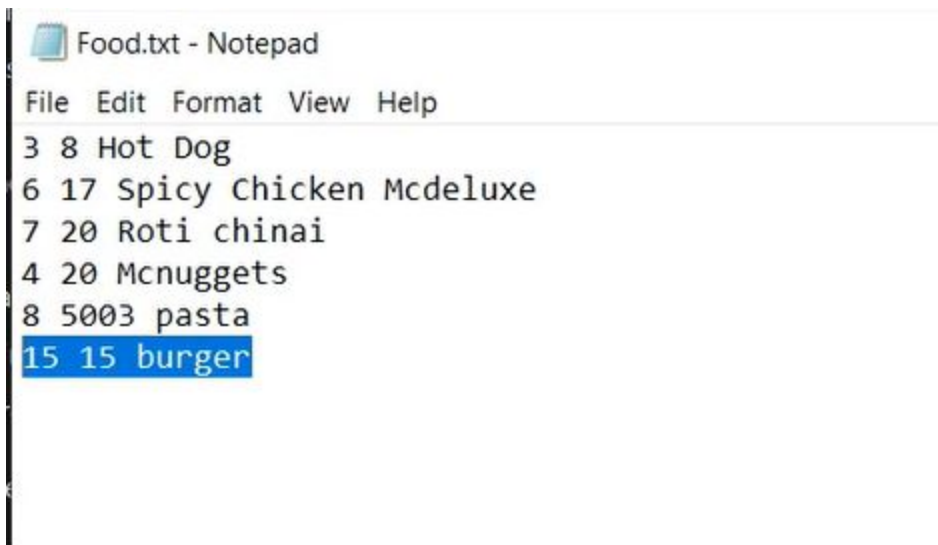
```
*****Customers*****
ID      Name      Phone
1       Omar      32323
5       Alaa      3544675
3       Ali       233432
663     angelo    1111933595
8       angelina   55505005
10      omar      50505050

Press anything and then press enter to continue
```

If the admin chose to show the list of customers , a record of the registered customers will appear.


```
Welcome! Please select your choice
1. Show list of Customers
2. Add Food
3. Search food
4. Search Customer
5. Exit
2
Enter id :15
Enter name : burger
Enter price :15
```

And if he chose to add a food item to the menu , he must input the id , name and price of the food then it will be added directly.



```
Food.txt - Notepad
File Edit Format View Help
3 8 Hot Dog
6 17 Spicy Chicken Mcdeluxe
7 20 Roti chinai
4 20 McNuggets
8 5003 pasta
15 15 burger
```

Welcome! Please select your choice

1. Show list of Customers

2. Add Food

3. Search food

4. Search Customer

5. Exit

3

Enter food id : 3

ID	Name	Price
3	Hot Dog	8

Enter anything and then press enter to continue ! _

Here ,the admin chose to search for food and the item with the keyed in id appeared.

Welcome! Please select your choice

1. Show list of Customers

2. Add Food

3. Search food

4. Search Customer

5. Exit

4

Enter customer id : 10

ID	Name	Phone
10	omar	50505050

Enter anything to continue ! _

```
Welcome! Please select your choice
1. Show list of Customers
2. Add Food
3. Search food
4. Search Customer
5. Exit
5
```

And finally the admin can search for a certain customer or he can choose to exit the system

4.0 Development Activities

Meeting Date	Members Present	Activity	Member Tasks
11/01/2021	All members present.	Discussion regarding requirements of the mini project and divided tasks for parts 1,2 and 3.	<p>OMAR HAMED & ALAA:</p> <p>Have an initial idea on implementation of the different algorithms. Coding the food ordering system.</p> <p>HABIBA: Plan System design (class diagram).</p> <p>VIMALRAJ: Introduction, objectives and report In-charge.</p>

<p>23/01/2021</p>	<p>All members present.</p>	<p>Discussion regarding report & presentation for mini project and divided tasks for parts 4 and 5.</p>	<p>OMAR HAMED & ALAA: Implementing the necessary project requirements & specifications into food ordering system code.</p> <p>VIMALRAJ & HABIBA: brainstormed necessary elements for report.</p> <p>VIMALRAJ: Making System Design Algorithm used: Flowchart</p> <p>HABIBA: Use case diagram, function explanation and prototype.</p>
--------------------------	-----------------------------	---	---

<p>27/01/2021</p>	<p>All members present.</p>	<p>Edit and Compile full Report & Record video presentation.</p>	<p>OMAR HAMED & ALAA: Review the flowchart. Testing the code, and bug fixing.</p> <p>VIMALRAJ & HABIBA: Correcting and Finalizing Complete report.</p> <p>All members participate in the video presentation.</p>
--------------------------	-----------------------------	--	--

Source Code

Admin.cpp

```
#include "Admin.hpp"
#include <iostream>

using namespace std;

Admin::Admin()
{
    id = 0;
    type = 0;
    position = " ";
    name = " ";
}

// constructor with arguments
Admin::Admin(int i, string n, int t, string e)
{
    id = i;
    type = t;
    position = e;
    name = n;
}

//Accessors and mutators
void Admin::setID(int a)
{
    id = a;
}

int Admin::getID()
{
    return id;
}
```

```

}

void Admin::SetType(int a)
{
    type = a;
}

int Admin::getType()
{
    return type;
}

void Admin::SetPosition(string a)
{
    position = a;
}

string Admin::getPosition()
{
    return position;
}

void Admin::setName(string a)
{
    name = a;
}

string Admin::getName()
{
    return name;
}

```

Admin.hpp

```

#ifndef Admin_H
#define Admin_H
#include <iostream>

using namespace std;

class Admin
{
private:
    int id, type;

```



```

        string position, name;

public:
    Admin();

    // constructor with arguments
    Admin(int i, string n, int t, string
e);

    //Accessors and mutators

    void setID(int a);

    int getID();

    void SetType(int a);

    int getType();

    void SetPosition(string a);

    string getPosition();

    void setName(string a);

    string getName();
};
#endif

```

Customer.cpp

```

#include "Customer.hpp"
#include <iostream>

using namespace std;

Customer::Customer(int _id, string _name,
long _phno)

```

```
{
    name = _name;
    id = _id;
    phno = _phno;
    next = NULL;
}

void Customer::displayinfo()
{
    cout << id << "\\t" << name << "\\t" <<
phno << endl;
}
```

Customer.hpp

```
#ifndef Customer_H
#define Customer_H
#include <iostream>

using namespace std;

class Customer
{
public:
    int id;
    string name;
    long phno;
    Customer *next = NULL;
    Customer(int, string, long);
    void displayinfo();
};
#endif
```

customerList.cpp

```
#include <iostream>
#include "Customer.hpp"
#include "CustomerList.hpp"
#include <windows.h>

using namespace std;

CustomerList::CustomerList()
{
    head = NULL;
    tail = NULL;
}

CustomerList::~~CustomerList()
{
    Customer *currNode = head;
    Customer *nextNode = NULL;

    while (currNode != NULL)
    {
        nextNode = currNode->next;
        delete currNode;
        currNode = nextNode;
    }

    head = NULL;
}

void CustomerList::insertNode(int id,
string name, long phon)
{
    Customer *temp = new Customer(id, name,
phon);
    temp->next = NULL;

    if (head == NULL)
```

```

        {
            head = temp;
            tail = temp;
        }
    else
    {
        tail->next = temp;
        tail = tail->next;
    }
}

void CustomerList::displayList()
{
    Customer *curr = head;
    while (curr != NULL)
    {
        curr->displayinfo();
        curr = curr->next;
    }
}

bool CustomerList::findCustomer(int i)
{
    Customer *temp = head;

    while (temp != NULL)
    {
        if (temp->id == i)
        {
            cout << "ID\tName \tPhone" <<
endl;

            temp->displayinfo();
            return true;
        }
        temp = temp->next;
    }

    return false;
}

```

```

bool CustomerList::logMessage(int i)
{
    Customer *temp = head;

    while (temp != NULL)
    {
        if (temp->id == i)
        {
            system("CLS");
            cout << "Welcome Back Mr/Mrs "
<< temp->name;
            Sleep(1500);

            return true;
        }
        temp = temp->next;
    }

    return false;
}

```

customerList.hpp

```

#ifndef CustomerList_H
#define CustomerList_H
#include <iostream>
#include "Customer.hpp"

using namespace std;

class CustomerList
{
private:
    Customer *head;
    Customer *tail;

public:

```

```
CustomerList();
~CustomerList();

void insertNode(int, string, long);
bool findCustomer(int);
bool logMessage(int);
void displayList();
};
#endif
```

Food.cpp

```
#include "Food.hpp"
#include <iostream>

using namespace std;

Food::Food(string _name, int _id, int
_price)
{
    name = _name;
    id = _id;
    price = _price;
    next = NULL;
}

//Display function to show the id, price
and name of the food
void Food::displayinfo()
{
    cout << id << "\t" << name << "\t" <<
price << endl;
}
```

Food.hpp

```
#ifndef Food_H
#define Food_H

#include <iostream>

using namespace std;

class Food
{
public:
    string name;
    int id;
    int price;
    Food *next;

    Food(string, int, int);

    //Display function to show the id,
    price and name of the food
    void displayinfo();
};
#endif
```

FoodList.cpp

```
#include "FoodList.hpp"
#include "Food.hpp"
#include <iostream>
using namespace std;
```

```
FoodList::FoodList()
{
    head = NULL;
    tail = NULL;
}

FoodList::~~FoodList()
{
    Food *currNode = head;
    Food *nextNode = NULL;

    while (currNode != NULL)
    {
        nextNode = currNode->next;
        delete currNode;
        currNode = nextNode;
    }

    head = NULL;
}

void FoodList::insertNode(string name, int
id, int price)
{
    Food *temp = new Food(name, id, price);
    temp->next = NULL;

    if (head == NULL)
    {
        head = temp;
        tail = temp;
    }
    else
    {
        tail->next = temp;
        tail = tail->next;
    }
}
```



```

void FoodList::displayList()
{
    Food *curr = head;
    while (curr != NULL)
    {
        curr->displayinfo();
        curr = curr->next;
    }
}

bool FoodList::printReciet(int i)
{
    Food *temp = head;

    while (temp != NULL)
    {
        if (temp->id == i)
        {
            cout << temp->id << "\t" <<
temp->name << "\t" << temp->price << endl;

            return true;
        }
        temp = temp->next;
    }

    return false;
}

bool FoodList::findFood(int i)
{
    Food *temp = head;

    while (temp != NULL)
    {
        if (temp->id == i)
        {
            cout << "\nID\t Name\t\t
Price" << endl;

```

```

        temp->displayinfo();
        return true;
    }
    temp = temp->next;
}

return false;
}

```

FoodList.hpp

```

#ifndef foodlist_H
#define foodlist_H
#include "Food.hpp"
#include <iostream>
using namespace std;

class FoodList
{
    Food *head;
    Food *tail;

public:
    FoodList();
    ~FoodList();

    void insertNode(string,int, int);
    bool findFood(int);
    bool printReciet(int);
    void displayList();
};
#endif

```

Main.cpp

```
#include <iostream>
#include <fstream>
#include <windows.h>
#include "Customer.hpp"
#include "Food.hpp"
#include "Admin.hpp"
#include "FoodList.hpp"
#include "CustomerList.hpp"
using namespace std;

//function to read the food file, it reads
id, price and name of the food and sets it.
//files operation is used

int ReadFood(FoodList &a)
{
    fstream file;
    file.open("Food.txt", ios::in);
    int size = 0;
    int id, price;
    string name;
    while (!file.eof())
    {
        file >> id;
        file >> price;
        getline(file, name);
        a.insertNode(name, id, price);

        size++;
    }

    return size;
}

//function for insertion sort to sort foods
id
void InsertionSort(int a[], int n)
```

```

{
    int i, j, min, temp;
    for (i = 0; i < n - 1; i++)
    {
        min = i;
        for (j = i + 1; j < n; j++)
            if (a[j] < a[min])
                min = j;
        temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }
}

//this function is to read customer from
the file.
//does the same thing as read food function
int ReadCustomer(CustomerList &a)
{
    fstream file;

    file.open("Customers.txt", ios::in);
    int size = 0, id;
    long phno;
    string name;
    while (!file.eof())
    {
        file >> id;
        file >> phno;
        getline(file, name);
        a.insertNode(id, name, phno);
        size++;
    }
    return size;
}

// this menu is for admin to do the
following operations

```

```

// the password for the admin is "password"
itself
int dispAdminMenu()
{
    system("CLS");
    int a;
    cout << "Welcome! Please select your
choice " << endl
        << "1. Show list of Customers" <<
endl
        << "2. Add Food" << endl
        << "3. Search food" << endl
        << "4. Search Customer" << endl
        << "5. Exit" << endl;

    cin >> a;
    return a;
}

// this menu is for customer to do the
following operations
int DispCustMenu()
{
    system("CLS");
    int a;
    cout << "Welcome! Please select your
choice " << endl
        << "1. Show Menu" << endl
        << "2. Show Receipt" << endl
        << "3. Exit" << endl;

    cin >> a;
    return a;
}

int main()
{
    //variables initialized

    FoodList fList;
    CustomerList cList;

```

```

int c, a, sizea = 0, sizeb, y;
int rec[100];
fstream file;

sizea = ReadFood(fList);
sizeb = ReadCustomer(cList);
//this is the first screen to run the
app
// we can select both admin and
customer.
//for admin, it will request for a
password, and the password is "password"
itself
cout << " Are you a : " << endl
    << "1. Customer " << endl
    << "2. Admin" << endl;
cin >> c;

if (c == 1)
{
    A:
    system("CLS");
    //variables initilized
    int p, id;
    string name;
    char any;
    long phno;
    //after pressing customer, this
menu will be shown
    cout << "Welcome to our food
ordering system !" << endl
        << "1. Existing Customer" <<
endl
        << "2. New Customer" << endl
        << "Please make a selection to
continue :";
    cin >> p;
    if (p != 1 && p != 2)
    {

```

```

        cout << "Invalid Choice please
try again!";
        goto A;
    }

    if (p == 2)
    {
        //for registering new customer
        cout << "Enter id :";
        cin >> id;
        cin.ignore(100, '\n');
        cout << "Enter name : ";
        getline(cin, name);
        cout << "Enter Phone Number :";
        cin >> phno;
        file.open("Customers.txt",
ios::out | ios::app);
        file << endl
            << id << " " << phno << "
" << name;
    }
    else if (p == 1)
    {
        //for existing customer which
only requires cusotmer id
        int ec, flag = 0;
        cout << "Enter Customer id : ";
        cin >> ec;
        for (int k = 0; k < sizeb; k++)
        {
            if (cList.logMessage(k))
            {
                goto here;
            }
        }
        if (flag == 0)
        {
            cout << "Sorry Invalid
Customer ID !!";

```

```

        Sleep(1000);
        goto A;
    }
}

here:
    a = DispCustMenu();
    switch (a)
    {
        //for fod ordering
        //fd is food class object and by
the object it gets the id, name and price
of the food
        case 1:
        {
            int s = 1;
            cout <<
"\n*****MENU*****"
                << "\nID\t Name\t\t
Price" << endl;

            fList.displayList();

            y = 0;
            while (s != 0)
            {
                //for ordering the food by
food's id number
                // if the user enters 0, it
terminates
                cout << "Choose your Orders
(Press 0 once you are done)";
                cin >> s;
                rec[y] = s;
                y++;
            }
            InsertionSort(rec, y);

            cout << "Thank you for shopping
with us!";

```



```

        cout << "\n Press any key and
then press enter to continue";
        cin >> any;
        goto here;
        break;
    }
    case 2:
    {
        // for showing the order
        system("CLS");
        cout << "\n\n Thank you for
using us as your preferred platform!!" <<
endl
        << "Your order is : " <<
endl
        << "\nID\t Name\t\t Price"
<< endl;
        for (int i = 0; i < y; i++)
        {
            fList.printReciet(rec[i]);
        }
        break;
    }
    case 3:
    {
        cout << endl
        << "Have a Nice Day!";
        break;
    }
}
else if (c == 2)
{
    //Admin
    //Admin password is "password"
itself
    system("CLS");
    char any;
    char pas[20];
    int b;

```

```

        cout << "Enter Password : ";
        cin >> pas;
        if (strcmp(pas, "password") != 0)
        {
            cout << "ACCESS DENIED!!";
        }
        else
        {
            here2:
                b = dispAdminMenu();
                switch (b)
                {
                    case 1:
                    {
                        cout <<
"\n\n*****Customers*****" <<
endl
                        << "ID\tName \tPhone"
<< endl;

                        cList.displayList();

                        cout << "\nPress anything
and then press enter to continue";
                        cin >> any;
                        goto here2;
                        break;
                    }
                    case 2:
                    {
                        // this case is for the
admin to add new food to the food list
                        int id, price;
                        string name;
                        cout << "Enter id :";
                        cin >> id;
                        cin.ignore(100, '\n');
                        cout << "Enter name : ";
                        getline(cin, name);
                        cout << "Enter price :";
                        cin >> price;

```

```

        //files operation, to add
new items to food.tx file
        file.open("Food.txt",
ios::out | ios::app);
        file << id << " " << price
<< " " << name << endl;
        goto here2;
        break;
    }
    case 3:
    {
        //searching is used here
        //to search for the food
identified by the food id
        int ec, flag = 0;
        cout << "\n\n Enter food id
: ";

        cin >> ec;

        if (fList.findFood(ec))
        {

            flag = 1;
        }

        if (flag == 0)
        {
            cout << "Invalid ID";
        }
        cout << "\n Enter anything
and then press enter to continue ! ";
        cin >> any;
        goto here2;
        break;
    }
    case 4:
    {
        //for searching the
existing customer by using customer id
        int ec, flag = 0;

```

```

        cout << "\n\n Enter
customer id : ";
        cin >> ec;

        if (cList.findCustomer(ec))
        {
            // cout << "ID\tName
\tPhone";

            cout << endl;

            flag = 1;
        }

        if (flag == 0)
        {
            cout << "Invalid ID";
        }
        cout << "\n Enter anything
to continue ! ";
        cin >> any;
        goto here2;
        break;
    }

    case 5:
    {
        cout << endl
            << "Have a Nice Day!";
        break;
    }
}

}

system("pause");
return 0;
}

```

-----END OF DOCUMENTATION-----

