ACKNOWLEDGEMENT

We would like to express the gratitude and sincere thanks to our respected and esteemed guide **Mr.Tej Sahi** Lecturer, Department of Computer Science for his valuable guidance, encouragement and help for completing this project. His useful suggestions for this whole work and co-operative behavior is sincerely acknowledged.

Also we would also like to express our sincere thanks to our college **New Summit College**, friends and others who directly or indirectly helped us to complete this project.

Ajay Jha

Purnima Karki

Srijana Raut

ABSTRACT

This project introduces a unique policy for managing the daily transactions in a canteen of a school, college or an organization. This mini project is entitled “Canteen Management System” and it aims at developing the canteen organization system using the object oriented language (C++) that enables a canteen to maintain its daily transactions. This project is designed to help the canteen department to make proper management of billing, ordering and managing its food supply. The application uses class and object concept, constructors and other features of C++ to generate menus, show messages.

This system includes reporting of all canteen expenses, food ordered, payment strategies using Data Report. This management system contains a database of each food item and updates the records in a daily basis. It can be implemented on a system which can be easily accessed by the user or customer and can order his/her desired food item.

There are separate menus for different kinds of task such as creating new file, adding new item to the file, modifying the items in file at any place, deleting particular item in file and deleting file. It is user friendly in the sense that there are several features in the program to guide the user in necessity.

TABLE OF CONTENTS

[ACKNOWLEDGEMENT iii](#_Toc396851146)

[ABSTRACT iv](#_Toc396851147)

[**INTRODUCTION TO OOP** 1](#_Toc396851148)

[PROJECT OVERVIEW 5](#_Toc396851149)

[Introduction 5](#_Toc396851150)

[Objective 5](#_Toc396851151)

[Project Features 5](#_Toc396851152)

[System Requirements 5](#_Toc396851153)

[**LITERATURE REVIEW** 6](#_Toc396851154)

[**ALGORITHM** 8](#_Toc396851155)

[**SOURCE CODE** 12](#_Toc396851156)

[**RESULTS** 22](#_Toc396851487)

[**EPILOGUE** 25](#_Toc396851489)

[Limitations 25](#_Toc396851490)

[Future Enhancements 25](#_Toc396851491)

[Conclusion 26](#_Toc396851492)

[REFERENCES 27](#_Toc396851493)

**INTRODUCTION TO OOP**

Object-oriented programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data.

The programming challenge was seen as how to write the logic, not how to define the data. Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. Examples of objects range from human beings (described by name, address, and so forth) to buildings and floors (whose properties can be described and managed) down to the little widgets on a computer desktop (such as buttons and scroll bars).

Basically, OOP consists of the following features:

* Dynamic Dispatch: When a method is invoked on an object, the object itself determines what code gets executed by looking up the method at run time in a table associated with the object. This feature distinguishes an object from an abstract data type (or module), which has a fixed (static) implementation of the operations for all instances. It is a programming methodology that gives modular component development while at the same time being very efficient.
* Encapsulation (or multi-methods, in which case the state is kept separate)
* Subtype polymorphism
* Object inheritance (or delegation)

The features of OOP are as follows:

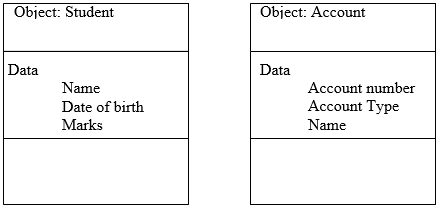
1. **Object**: Objects are the entities in an object oriented system through which we perceive the world around us. We naturally see our environment as being composed of things which have recognizable identities & behavior. The entities are then represented as objects in the program. They may represent a person, a place, a bank account, or any item that the program must handle. For example Automobiles are objects as they have size, weight, color etc. as attributes (i.e. data) and starting, pressing the brake, turning the wheel, pressing accelerator pedal etc. as operation (that is functions).

Example of objects:

* Physical Objects: Automobiles in traffic flow
* Simulation: Countries in economic model, air craft in traffic – control system
* Computer user environment objects: Windows, menus, icons etc.
* Data storage constructs: Stacks, Trees etc.
* Human entities: Employees, student, teacher etc.
* Geometric objects: Point, line, triangle etc.

Objects mainly serve the following purposes:

* Understanding the real world and a practical base for designers
* Decomposition of a problem into objects depends on the nature of problem.



1. **Classes**: A class is a collection of objects of similar type. For example manager, peon, secretary, clerk are member of the class employee and class vehicle includes objects car, bus, etc. It defines a data type, much like a struct in C programming language and built in data type(int char, float etc). It specifies what data and functions will be included in objects of that class. Defining class doesn’t create an object but class is the description of object’s attributes and behaviors. For example,

Person Class : Attributes: Name, Age, Sex etc.

Behaviors: Speak(), Listen(), Walk()

Vehicle Class: Attributes: Name, model, color, height etc

Behaviors: Start(), Stop(), Accelerate() etc.

When class is defined, objects are created as

<classname> <objectname> ;

If employee has been defined as a class, then the statement employee manager;

Will create an object manager belonging to the class employee.

Each class describes a possibly infinite set of individual objects, each object is said to be an instance of its class and each instance of the class has its own value for each attribute but shares the attribute name and operations with other instances of the class. The following points gives the idea of class:

* A class is a template that unites data and operations.
* A class is an abstraction of the real world entities with similar properties.
* Ideally, the class is an implementation of abstract data type.

1. **Encapsulation and Data Abstraction**: The wrapping up of data and function into a single unit is called encapsulation. Encapsulation is most striking feature of a class. The data is not accessible from outside of class. Only member function can access data on that class. The insulation of data from direct access by the program is called data hiding. That is data can be hidden making them private so that it is safe from accidental alteration details or explanation. It focuses the outside view of an object, separating its essential behavior from its implementation. The class is a construct in C++ for creating user-defined data types called Abstract Data Types (ADT).
2. **Inheritance**: Inheritance is the process by which objects of one class acquire the characteristics of object of another class. In OOP, the concept of inheritance provides the idea of reusability. We can use additional features to an existing class without modifying it. This is possible by deriving a new class (derived class) from the existing one (base class).This process of deriving a new class from the existing base class is called inheritance. It supports the concept of hierarchical classification. It allows the extension and reuse of existing code without having to rewrite the code.
3. **Polymorphism**: Polymorphism means “having many forms”. The polymorphism feature allows different objects to respond to the same message in different ways, the response specific to the type of object. Polymorphism is important when object oriented programs dynamically creating and destroying the objects in runtime. Example of polymorphism in OOP is operator overloading, function overloading.

For example operator symbol ‘+’ is used for arithmetic operation between two numbers, however by overloading (means given additional job) it can be used over Complex Object like currency that has Rs and Paisa as its attributes, complex number that has real part and imaginary part as attributes. By overloading same operator ‘+’ can be used for different purpose like concatenation of strings.

PROJECT OVERVIEW

Introduction

This program is based around the dynamics of a user input environment. It's a database software that manages items for small canteens and hotels. The goal of the software is to provide efficient method of managing items in such canteens/hotels by the use of various functions that allows user to create, add, modify and delete the items stored in the database.

Objective

The following are the major objectives of this project

* Use classes and objects to make the program object oriented
* Use various user defined function as well
* Combine looping statement
* Combine various file handling operations

Project Features

* Object oriented program using classes and objects
* User interactive menus

System Requirements

The general system requirements are:

* Windows XP/Vista/7/8/10 or Mac OS
* 512 MB or greater RAM
* 1 GHz or greater processor
* 128 MB or greater video graphics
* Pre-installed C++ Compilers

**LITERATURE REVIEW**

The concept of this project was to create a database management software for a canteen or a particular kind of food court. It is implemented as a program to create a file that stores the information about the food items-adding, deleting, displaying, and ordering the food items in a canteen or food court. This project named “Canteen Management System” implements the use in real world perspective to help the canteen or a food court management to ease their transactions for satisfying the customer’s order who visit their canteen or food court.

In order to start off with a programming project, we need to have sufficient knowledge about the programming language, the system used and also the similar systems that are in use. In the programming section, we need to understand the basic concepts class, objects and file handling. For this, we referred to “Object Oriented Programming in C++” by Robert Lafore and few websites to get the clear idea. For further learning about file handling we referred to “Object Oriented Programming” by E. Balagurusamy and other materials. These books helped us to know about the file handling library function and its syntax. These books were very informative and were enormously influential in this project. By using file handling, we performed the functions such as to add new information, delete, and search and view the information about the food item in a canteen. We can proceed through various tasks like to add data item, delete data item, modify, and view data item stored in the database.

Our objective to create a transaction management system for a food business organization such as a canteen. Canteen is mainly popular in educational institutions as well as other organizations in which people spend their most of the time. Such organization can definitely be the offices or schools/colleges where the employees or students want to erase their hunger and depend on the canteen for their food. The motive to create this kind of management system started when we observed the cafeteria of our own college. With more than two thousand students in total, the canteen has to serve above fifteen hundred students and staff in a day. To take a note of the expenses and to manage the proper transaction, it is definitely a difficult job. The canteen has a token system which is provided to each student after the payment is done. Since all the calculations is done by humans, it is sure that mistakes can be done and may result loss if there comes any mistake in payment and transactions. To avoid this, we developed this concept and tried to implement it by developing a program. As we know number of canteens and hotels are increasing day by day. Along with the increase of canteens and hotels number of people visiting those place will also increase and so will the complexity in handling the customers. The quantity of items in canteens/hotels will increase so will increase the complexity in managing those items. The traditional way of selling the item to the customer is very time consuming. So, there should be a method that can efficiently manage the information about these products and utilize them to help in activities such as entry, viewing and billing of products. The billing process using this program will be very fast compared to the traditional billing which used to be done manually. There are other advantages of the database management system such as some database management system are capable of creating a report on the basis of the daily sale in the department store and profit obtained. So, canteen management system is very useful and efficient in canteens/hotels.

**ALGORITHM**

**Main**

Step 1: Start

Step 2: call the function “intro”

Step 3: create a menu 1.CUSTOMER 2.ADMINISTRATOR 3.EXIT

Step 4: switch the menu list and input the user choice

Step 5: case 1: request for order if yes call the function “place\_order”

Case 2: call the function “admin\_menu”

Case 3: Exit

Step 6: stop the program.

**Intro**

Step 1: start

Step 2: display the information of group member of project and whom you going to submit

Step 3: stop.

**place\_order**

Step 1: start

Step 2: input the item number and quantity for place the order.

Step 3: open the “canteen.dat”, if not empty

Calculate the price according to the rate quantity and display the bill.

If empty, give the message.

Step 4: stop the program.

**Admin\_menu**

Step 1: start

Step 2: create a menu

1:ADMIN MENU

2:CREATE ITEM

3:QUERY

4:MODIFY ITEMS

5:DELETE ITEMS

6:VIEW ITEMS MENU

Step 3: input the choice

Step 4: switch the menu

Step 5: case 1: clear screen

Case 2: display \_ all

Case 3: display\_sp(num)

Case4: modify \_ items

Case5: delete\_items

Step 6: call the function “menu()”

Step 7: stop

**void menu**

Step 1: start

Step 2: open “canteen.dat”

Step 3: if empty give the message

Else, calculate the bill according to the order placement and display the output format

Step 4: stop

**void delete\_item**

Step 1: start

Step 2: enter the number of items to be deleted

Step 3: open “canteen.dat”

Step 4: if empty leave the message

Else, make the temporary file copy all the items except the input one

Rename the temporary as “canteen.dat”

Step 5: give the message ‘record deleted’

Step 6: stop

**void modify\_items**

Step 1: start

Step 2: input the item number to be modified

Step 3: open the “canteen.dat”

Step 4: if item not found give the message ‘record not found’

Step 5: modify the items accordingly to the user input

Step 6: leave the message ‘record updated’

Step 7: close “canteen.dat”

Step 8: close

**void display\_all**

Step 1: open the “canteen.dat”

Step 2: if record not found leave the message ‘record not exist’

Else, display all the record in the effective output format

Step 3: Stop

**void write\_items**

Step 1: start

Step 2: open the file “canteen.dat”

Fp.open(‘canteen.dat’,ios::out| ios::app);

It.create\_item();

Fp.close();

Step 3: display the message item has been created

Step 4: closep

**void create\_item**

Step 1: start

Step 2: input the item number, name of item, price of item, discount percentage

Step 3: stop **void\_item**

Step 1: start

Step 2: input the item number, name of item, price of item, discount percentage

Step 3: stop

**SOURCE CODE**

#include<conio.h>

#include<stdio.h>

#include<process.h>

#include<fstream>

#include<iostream>

#include<stdlib.h>

using namespace std;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CLASS USED IN PROJECT

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class item

{

int itemno;

char name[50];

float price,qty,tax,dis;

public:

void create\_item()

{

cout<<"\nPlease Enter The Item No. ";

cin>>itemno;

cout<<"\n\nPlease Enter The Name of The item ";

gets(name);

cout<<"\nPlease Enter The Price of The item ";

cin>>price;

cout<<"\nPlease Enter The Discount (%) ";

cin>>dis;

}

void show\_item()

{

cout<<"\nThe item No. "<<itemno;

cout<<"\nThe Name of The item : ";

puts(name);

cout<<"\nThe Price of The item : "<<price;

cout<<"\nDiscount : "<<dis;

}

int retitemno()

{

return itemno;

}

float retprice()

{

return price;

}

char\* retname()

{

return name;

}

int retdis()

{

return dis;

}

}; //class ends here

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// global declaration for stream object, object

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fstream fp;

item it;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to write in file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void write\_item()

{

char nitem;

do

{

fp.open("canteen.dat",ios::out|ios::app);

it.create\_item();

fp.write((char\*)&it,sizeof(item));

fp.close();

cout<<"\n\nThe Item Has Been Created ";

cout<<"\nDo you want to create new item (Y/N)";

cin>> nitem;

}while(nitem=='y'||nitem=='Y');

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to read records from file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void display\_all()

{

cout<<"\n\n\n\t\tDISPLAY ALL RECORD !!!\n\n";

fp.open("canteen.dat",ios::in);

while(fp.read((char\*)&it,sizeof(item)))

{

it.show\_item();

cout<<"\n\n====================================\n";

getch();

}

fp.close();

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to read specific record from file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void display\_sp(int n)

{

int flag=0;

fp.open("canteen.dat",ios::in);

while(fp.read((char\*)&it,sizeof(item)))

{

if(it.retitemno()==n)

{

it.show\_item();

flag=1;

}

}

fp.close();

if(flag==0)

cout<<"\n\nrecord not exist";

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to modify record of file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void modify\_item()

{

int no,found=0;

cout<<"\n\n\tTo Modify ";

cout<<"\n\n\tPlease Enter The Item No.";

cin>>no;

fp.open("canteen.dat",ios::in|ios::out);

while(fp.read((char\*)&it,sizeof(item)) && found==0)

{

if(it.retitemno()==no)

{

it.show\_item();

cout<<"\nPlease Enter The New Details of Item"<<endl;

it.create\_item();

int pos=-1\*sizeof(it);

fp.seekp(pos,ios::cur);

fp.write((char\*)&it,sizeof(item));

cout<<"\n\n\t Record Updated";

found=1;

}

}

fp.close();

if(found==0)

cout<<"\n\n Record Not Found ";

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to delete record of file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void delete\_item()

{

int no;

cout<<"\n\n\n\tDelete Record";

cout<<"\n\nPlease Enter The Item no. of The Item You Want To Delete";

cin>>no;

fp.open("canteen.dat",ios::in|ios::out);

fstream fp2;

fp2.open("Temp.dat",ios::out);

fp.seekg(0,ios::beg);

while(fp.read((char\*)&it,sizeof(item)))

{

if(it.retitemno()!=no)

{

fp2.write((char\*)&it,sizeof(item));

}

}

fp2.close();

fp.close();

remove("canteen.dat");

rename("Temp.dat","canteen.dat");

cout<<"\n\n\tRecord Deleted ..";

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to display all products price list

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void menu()

{

fp.open("canteen.dat",ios::in);

if(!fp)

{

cout<<"ERROR!!! FILE COULD NOT BE OPEN\n\n\n Go To Admin Menu to create File";

cout<<"\n\n\n Program is closing ....";

getch();

exit(0);

}

cout<<"\n\n\t\t MENU\n\n";

cout<<"====================================================\n";

cout<<"P.NO.\t\tNAME\t\tPRICE\n";

cout<<"====================================================\n";

while(fp.read((char\*)&it,sizeof(item)))

{

cout<<it.retitemno()<<"\t\t"<<it.retname()<<"\t\t"<<it.retprice()<<endl;

}

fp.close();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function to place order and generating bill for item

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void place\_order()

{

int order\_arr[50],quan[50],c=0;

float amt,damt,total=0;

char ch='Y';

menu();

cout<<"\n============================";

cout<<"\n PLACE YOUR ORDER";

cout<<"\n============================\n";

do

{

cout<<"\n\nEnter The Item No.: ";

cin>>order\_arr[c];

cout<<"\nQuantity in number : ";

cin>>quan[c];

c++;

cout<<"\nDo You Want To Order Another Item ? (y/n)";

cin>>ch;

}while(ch=='y' ||ch=='Y');

cout<<"\n\nThank You For Placing The Order";getch();

cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INVOICE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\nItem No.\tItem Name\tQuantity \tPrice \tAmount \tAmount after discount\n";

for(int x=0;x<=c;x++)

{

fp.open("canteen.dat",ios::in);

fp.read((char\*)&it,sizeof(item));

while(!fp.eof())

{

if(it.retitemno()==order\_arr[x])

{

amt=it.retprice()\*quan[x];

damt=amt-(amt\*it.retdis()/100);

cout<<"\n"<<order\_arr[x]<<"\t"<<it.retname()

<<"\t"<<quan[x]<<"\t"<<it.retprice()<<"\t"<<amt<<"\t"<<damt;

total+=damt;

}

fp.read((char\*)&it,sizeof(item));

}

fp.close();

}

cout<<"\n\n\t\t\t\t\tTOTAL = "<<total;

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// INTRODUCTION FUNCTION

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void intro()

{

cout<<endl<<endl;

cout<<endl<<endl<<endl;

cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

cout<<"\n\t\t\t\* CANTEEN MANAGEMENT SYSTEM \*";

cout<<"\n\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

cout<<endl<<endl<<endl;

cout<<"\n\nSUBMITTED BY";

cout<<"\t\t\tSUMITED TO"<<endl;

cout<<"\nAjay Jha";

cout<<"\t\t\t Mr.Tej Sahi";

cout<<"\n\nPurnima Karki";

cout<<"\t\t\t Department of Computer Science and Information Technology";

cout<<"\n\nSrijana Raut";

cout<<"\t\t\t New Summit College,Old-Baneshowr";

getch();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ADMINSTRATOR MENU FUNCTION

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void admin\_menu()

{

char ch2;

cout<<"\n\n\n\tADMIN MENU";

cout<<"\n\n\t1.CREATE ITEM";

cout<<"\n\n\t2.DISPLAY ALL ITEM";

cout<<"\n\n\t3.QUERY ";

cout<<"\n\n\t4.MODIFY ITEM";

cout<<"\n\n\t5.DELETE ITEM";

cout<<"\n\n\t6.VIEW ITEM MENU";

cout<<"\n\n\t7.BACK TO MAIN MENU";

cout<<"\n\n\tPlease Enter Your Choice (1-7) ";

ch2=getche();

switch(ch2)

{

case '1':

write\_item();

break;

case '2': display\_all();

break;

case '3':

int num;

cout<<"\n\n\tPlease Enter The Item No. ";

cin>>num;

display\_sp(num);

break;

case '4': modify\_item();break;

case '5': delete\_item();break;

case '6': menu();

getch();

case '7':

break;

default:

cout<<"\a";admin\_menu();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// THE MAIN FUNCTION OF PROGRAM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

char ch;

char yes;

intro();

top:

do

{

cout<<"\n\n\n\tMAIN MENU";

cout<<"\n\n\t01. CUSTOMER";

cout<<"\n\n\t02. ADMINISTRATOR";

cout<<"\n\n\t03. EXIT";

cout<<"\n\n\tPlease Select Your Option (1-3) ";

ch=getche();

switch(ch)

{

case '1':

menu();

cout<<"Do you like to place an order(Y/N)"<<endl;

cin>>yes;

if(yes=='y'||yes=='Y')

{

place\_order();

}

else

goto top;

getch();

break;

case '2': admin\_menu();

break;

case '3':exit(0);

default :cout<<"\a";

}

}while(ch!='3');}

**RESULTS**

Fig: Home Screen

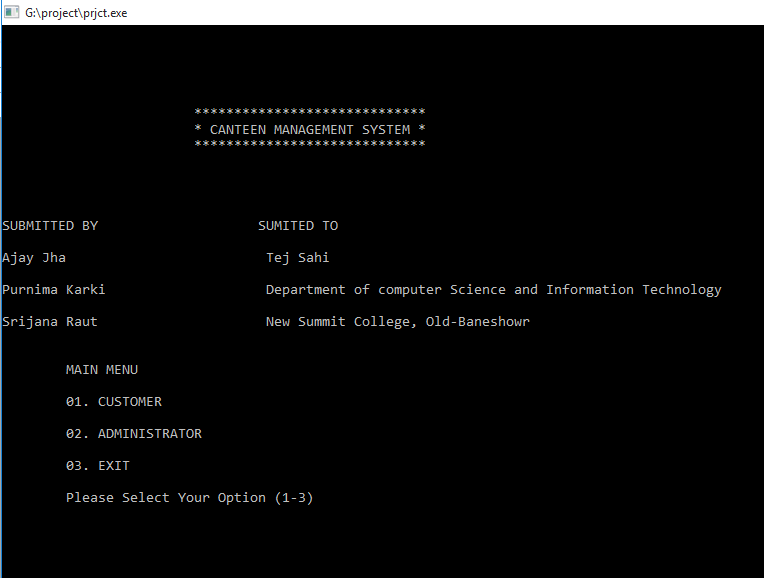


Fig: User Select Menu

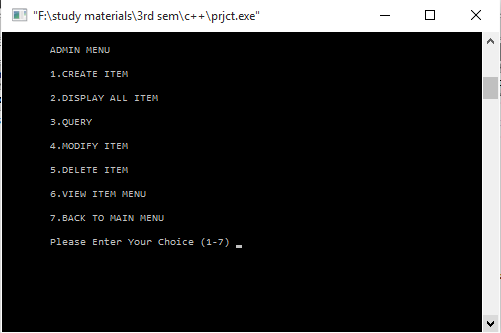


Fig: Administrator Menu

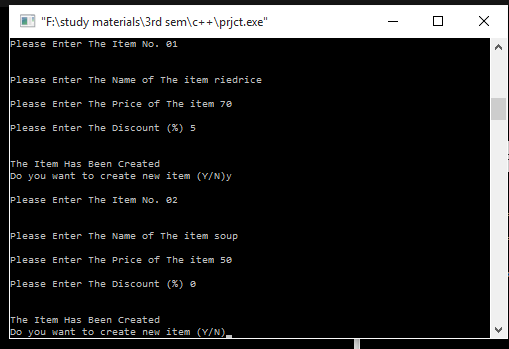


Fig: Updating Item in Database

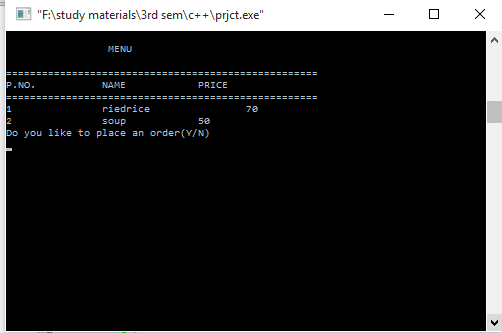


Fig: Item Menu

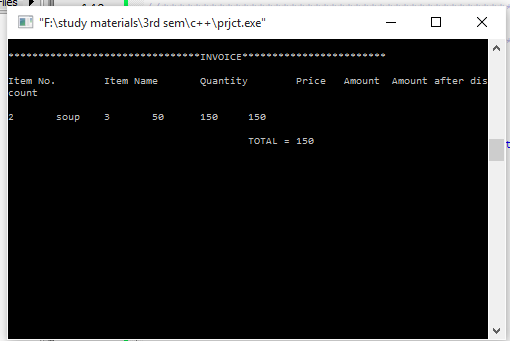


Fig: Invoice

**EPILOGUE**

## Limitations

Despite of our sincere efforts for making software as practicable as possible, this project still has some limitations. Firstly, we were unable to use graphical functions. The program is less user interactive since graphics is not present. Secondly there is no pointing system such as mouse pointer to select different functions. The feature of mouse pointer would have made this software more user-friendly. Since the program does not isolate user from administrator anybody can change or update the information about the items present in the database. Any user can access and modify the data in database which is its major limitation. Bill can only be shown while the program is running and cannot be printed which another drawback of this project. Data entry and billing is done in this software manually entering the information of data items.

Future Enhancements

Despite of our hard efforts this project could not reach the goal that we had expected. Due to lack of complete knowledge of functions and compilers, we could not use all of the features in an Object Oriented Programming language like C++. We have discussed the limitations above, but it is not that it cannot be upgraded. Much change can be introduced in order to take such implementations to higher level. Firstly, involvement of interactive graphics can be done so that any user can use this software. Introduction of mouse pointer will do great help for the program and make it interactive. We will also introduce forms for data entry. The administrator will only be allowed to add, modify and delete the data items in database while other user can perform billing, search and view data items in database. A provision of bill invoice printing support system can also be introduced, which will take this project to a whole new level.

Conclusion

After months of hard work and devotion, we finally implemented such a system, which was able to handle the transactions and management of a canteen. The limitations have described our failure to achieve what we expected while starting this project and what we have ended up with. Still, improvements is always possible and proper upgrading will definitely make this project come to use in any of the canteen or other kind of food courts where it is difficult to handle hundreds of customers at a time.

If we ignore the limitations for a while, we gained a lot from this project. This project is the outcome of a month long effort and in this time that was spent in developing, we learnt a lot about programming and especially we sharpened our skills in using file handling in C++. Also this project work helped us to make feel of working in real time environment. We have tried to complete the project without making any mistakes. But there will be always some minor errors and a little space for improvement. Suggestions and advice are welcome for correction and improvement.

REFERENCES

[1] E. Balagurusamy. *Object Oriented Programming with C++.* India: Tata McGraw-Hill, 2008.

[2] Adobe, “Features of OOP”, Internet: <http://www.adobe.com/devnet/actionscript/learning/oop-concepts/objects-and-classes.html>. [March 28 2017]