Grocery Store Management System

Project submitted to the

SRM-University – AP, Andhra Pradesh

For the partial fulfilment of the requirements to award the degree of

Bachelor of technology

In

Computer science engineering

School of engineering and sciences

Submitted by

K.Venkatesh-AP21110010933

Under the guidance of

ManojKumar Vivekanand

SRM University – AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[December, 2022]

Abstract

The goal of the grocery shop management system is to automate the current manual system with the aid of computerised hardware and comprehensive computer software, meeting their requirements, in order to store their valuable data and information for a longer period of time with simple access and manipulation. The necessary hardware and software are readily available and simple to use.

The management system for grocery stores mentioned above can result in an error-free, secure, dependable, and quick management system. Instead of focusing on record keeping, it can help the user focus on their other activities. As a result, it will aid organisations in making better use of their resources. The company can keep computerised records updated without making duplicate inputs. In order to access the information, one does not need to be side-tracked by irrelevant information.

The goal is to automate its current manual system with the aid of computerised tools and comprehensive computer software, meeting their needs, so that their important data and information can be stored for a longer period of time with simple access and manipulation.

Introduction:

In order to overcome the issues that existed with the currently in use manual system, the "Grocery Store Management System" was created. The difficulties our current system has are supported by this programme in an effort to eliminate and, in some circumstances, decrease them. Additionally, this system is created to meet the specific requirements of the business to conduct operations efficiently and effectively.

The program is kept as simple as possible to reduce data entry errors. Additionally, it displays an error notice when you enter invalid data. The user doesn't require any formal training to use this system. This alone demonstrates how user-friendly it is. As previously mentioned, the grocery shop management system can result in an error-free, secure, dependable, and quick management system. Instead of focusing on record keeping, it can help the user focus on their other activities. As a result, it will aid organisations in making better use of their resources.

No matter how big or small a firm is, managing employee, product, and stock information is a struggle. We provide unique personnel management systems that are tailored to your managerial demands because every Grocery Shop Management System has different Customer wants. This is intended to aid with strategic planning and will help you make sure that your company has the appropriate amount of knowledge and information for your future objectives. In the end, these solutions will enable you to manage resources more effectively.

Functionalities of this project:

We have designed the project in a way that it can manage a Grocery Store's Revenue, and calculate their costs, profits, salaries.

This application allows the user to:

Add, Store and Update Product Details

Add, Store and Update Employee Details

Calculate the daily profit by finding the profit from sales, and subtracting the employee's daily wage

Use file handling to store the profits and products for next days

This software is platform independent and can be used in any system

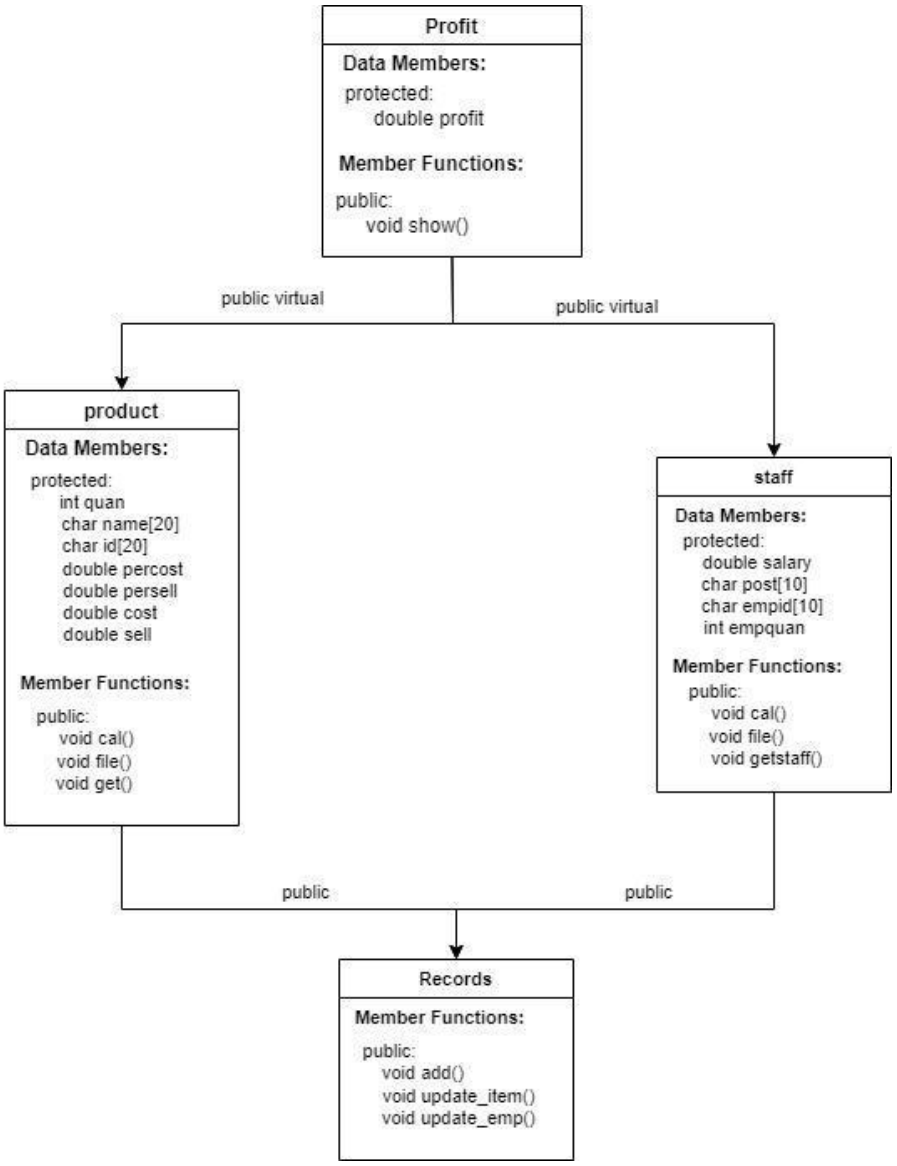
Methodology

2.1 Design:

The UML Class diagram is a graphical notation used to construct and visualize object-oriented systems. A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system:

- Classes
- their data members
- member functions (or methods)
- and the relationships among objects.

Following is the class diagram of our Project:



We used the fundamental ideas of object-oriented programming to create this project (OOP). The Object-Oriented concepts we used to create this project were classes and objects, data encapsulation and abstraction, array data members, static data members, and inheritance. To implement the OOP concepts, we employed the C++ programming language. Data members were kept as secure as possible. This program's design allows it to save employee and product information in text (.txt) files. The figure above illustrates how the ideas of public inheritance or virtual public inheritance were applied.

- protected inheritance makes the public and protected members of the base class protected in the derived class.

- Virtual public inheritance is a C++ technique that ensures that only one copy of a class's member variables is inherited by the second-level derivates

Implementation:

```
#include<fstream>
#include<iostream>
#include<string>
#include<cstdlib>
#include<cstring>
using namespace std;
```

These are the header files used in this program.

The fstream header file represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout, cerr, etc. iomanip: iomanip stands for input-output manipulators.

The string and cstring header files are used to perform operations on strings using various built-in functions.
The cstdlib header file offers functions for dynamic memory allocation, conversion between datatypes, process control, searching and sorting, math, etc.

Profit Class:

```
class Profit{
        protected:
            double profit;
        public:
            void show()
            {
                ifstream x("Profits.txt");
                if(!x)
                    cout<<"\n\t\t\tPROFIT = 0 \n";
                else
                {
                    x>>profit;
                    cout<<"\n\t\t\tTOTAL STARTING PROFIT :  "<<profit<<endl;
                    x.close();
                }
            }
};
```

This is the Profit class in the program. This class provides the profit data member, and the show() member function, which is used to display the profit from the previous executions of the program.

Product Class:

```
class product:public virtual Profit{
    protected:
        int quan;
        char name[20];
        char id[20];
        double percost;
        double persell;
        double cost;
        double sell;
    public:
        void cal();
        void file();
        void get();
};
```

This is the profit class in our program. This provides the quan, name, id, percost, presell, cost, sell data members to store various details of the product, like the quantity sold, name of the product, product ID, cost price per unit of product, selling price per unit, total cost and total selling price for that product respectively.

This also provides the cal(), file() and get() member functions to calculate the total cost and selling prices and profits from that product, write the details to a file with the title as product ID, and get the product details from the user respectively.

void cal()

```
void product::cal()
        {
            cost=percost*quan;
            sell=persell*quan;
            profit=profit+(sell-cost);
        }
```

void file()

```
void product::file()
        {
            char file[20];
            strcpy(file,id);
            strcat(file,".txt");
            ofstream f(file);
            f<< "\n\t\t\tProduct Name : "<<name<<
            "\n\t\t\tProduct ID : "<<id<<
            "\n\t\t\tCost price of product : "<<percost<<
            "\n\t\t\tSelling price of product : "<<persell<<
            "\n\t\t\tQuantity : "<<quan<<
            "\n\t\t\tTotal cost: "<<cost<<
            "\n\t\t\tSell : "<<sell<<endl;
            f.close();
            ofstream p("Profits.txt");
            p<<profit;
            p.close();
        }
```

void get()

```
void product::get()
        {
            int s;
            cout<<"\n\t\t\tNumber of products to be added : ";
            cin>>s;
            for(int i=0;i<s;i++)
            {
                cout<<endl;
                cout<<"\t\t\tEnter product name :";
                cin>>name;
                cout<<"\t\t\tEnter product id : ";
                cin>>id;
                cout<<"\t\t\tEnter cost price of product : ";
                cin>>percost;
                cout<<"\t\t\tEnter selling price of product : ";
                cin>>persell;
                cout<<"\t\t\tTotal products sold quantity : ";
                cin>>quan;
                cal();
                file();
            }
        }
```

Staff Class:

```
class staff:public virtual Profit{
    protected:
        double salary;
        char post[10];
        char empid[10];
        int empquan;
    public:
        void cal();
        void file();
        void getstaff();
};
```

This is the staff class in our program. This provides the salary, post, empid, empquan data members to store the salaries, employee names, employee IDs and the number of employees respectively

We also have the cal(), file() and getstaff() member functions to calculate the total salaries and remove it from profit, write the details to a file with the title as employee ID, and get the employee details from the user respectively

void cal()

```
void staff::cal()
        {
            profit=(profit-(salary*empquan)/30);
        }
```

void file()

```
void staff::file()
        {
            char file[20];
            strcpy(file,empid);
            strcat(file,".txt");
            ofstream f(file);
            f<< "\n\t\t\tNumber of working employees : "<<empquan<<
            "\n\t\t\tEmployee Salary : "<<salary<<
            "\n\t\t\tEmployee Name : "<<post<<
            "\n\t\t\tEmployee ID : "<<empid<<endl;
            f.close();
            ofstream p("Profits.txt");
            p<<profit;
            p.close();
        }
```

void getstaff()

```
void staff::getstaff()
        {
            cout<<"\t\t\tEnter Salary : ";
            cin>>salary;
            cout<<"\t\t\tEnter Number of Employees : ";
            cin>>empquan;
            for(int i=0;i<empquan;i++)
            {
                cout<<endl;
                cout<<"\t\t\tEnter Employee name : ";
                cin>>post;
                cout<<"\t\t\tEnter Employee ID : ";
                cin>>empid;
                cal();
                file();
            }
        }
```

Records Class:

```
class Records:public staff, public product
{
    public:
        void add();
        void update_item();
        void update_emp();
};
```

The Records class is used to add and update product or employee entries into the system. This class has three member functions add(), update_item() and update_emp() to perform these operations.

void add()

```
void Records::add()
        {
            int ch;
            char name[20];
            while(1)
            {
                cout<<endl<<"\t\t\t1. Add Products\n\t\t\t2. Add employees\n\t\t\t3. Exit\n";
                cout<<"\t\t\tEnter choice : ";
                cin>>ch;
                if(ch==1)
                {
                    get();
                }
                else if(ch==2)
                {
                    getstaff();
                }
                else if(ch==3)
                    break;
                else
                    cout<<"\t\t\tInvaid Option. Try again\n";
            }
        }
```

void update_item()

```
void Records::update_item()
        {
            char id[20];
            char pid[20];
            char c;
            cout<<"\n\t\t\tProduct ID to modify : ";
            cin>>id;
            char file[20];
            char file2[20];
            strcpy(file,id);
            strcat(file,".txt");
            fstream fout(file, ios::in|ios::out);
            if(!fout)
            {
                cout<<"\t\t\tProduct ID not found\n";
            }
            else
            {
                cout<<"\t\t\tProduct found! \n";
                cout<<"\n\t\t\tUpdate product name : ";
                cin>>name;
                cout<<"\t\t\tUpdate Cost Price per product : ";
                cin>>percost;
                cout<<"\t\t\tUpdate Selling Price per product : ";
                cin>>persell;
                cout<<"\t\t\tUpdate total products sold quantity : ";
                cin>>quan;
                cost=percost*quan;
                sell=persell*quan;
                profit=profit +(sell-cost)*365;
                fout<<"\n\t\t\tProduct name : "<<name<<
                "\n\t\t\tProduct id: "<<id<<
                "\n\t\t\tCost per product: " <<percost<<
                "\n\t\t\tSell per product: "<<persell<<
                "\n\t\t\tQuantity: "<<quan<<
                "\n\t\t\tTotal cost: "<<cost<<
                "\n\t\t\tSell: "<<sell;
                fout.close();
            }
        }
```

void update_emp()

```
void Records::update_emp()
        {
            char id[20];
            char c;
            cout<<"\n\t\t\tEnter employee ID to modify : ";
            cin>>id;
            char file[20];
            strcpy(file,id);
            strcat(file,".txt");
            fstream fout(file, ios::in|ios::out);
            if(!fout)
            {
                cout<<"\t\t\tEmployee ID not found\n";
            }
            cout<<"Update Employee Name : ";
            cin>>post;
            fout<< "\n\t\t\tNumber of working employees: "<<empquan<<
            "\n\t\t\tEmployee Salary: "<<salary<<
            "\n\t\t\tEmployee Name :"<<post<<
            "\n\t\t\tEmployee ID: "<<empid;
            fout.close();
        }
```

We also have an independent function void start() which displays the menu and checks the user input choice with the available cases, and performs the respective operation. This function creates an object of the class Records named “a” or “d”. It then uses these objects to do many tasks using the functions directly in Records class, or inherited from the staff or product class to the Records class.

void start()

```
void start()
{
    while(1)
    {
        cout<<"\n\t\t\t==============================\n";
        cout<<"\n\t\t\t    WELCOME TO GROCERY MART   \n";
        cout<<"\n\t\t\t==============================\n\n";
        cout<<"\n\t\t\t1. Add Entries\n\n\t\t\t2. Show profit"<<
        "\n\n\t\t\t3. Search Product Details\n\n\t\t\t4. Search Employee Details"<<
        "\n\n\t\t\t5. Modify Product Details\n\n\t\t\t6. Modify Employee Details\n\n\t\t\t7. Exit\n";
        cout<<"\n\n\t\t\tSelect your choice : ";
        int u;
        cin>>u;
        if(u==1)
        {
            Records a;
            a.show();
            a.add();
        }
        else if(u==2)
        {
            Records d;
            d.show();
        }
        else if(u==3)
        {
            char id[20];
            char c;
            cout<<"\n\t\t\tEnter ID to search : ";
            cin>>id;
            char file[20];
            strcpy(file,id);
            strcat(file,".txt");
            ifstream x(file);
            if(!x)
            {
                cout<<"\n\t\t\tProduct ID not found\n";
            }
            while(x)
            {
                x.get(c);
                cout<<c;
            }
            x.close();
        }
```

```
else if(u==4)
        {
            char eid[20];
            char d;
            cout<<"\n\t\t\tEnter employee ID to search : ";
            cin>>eid;
            char file[20];
            strcpy(file,eid);
            strcat(file,".txt");
            ifstream y(file);
            if(!y)
            {
                cout<<"\n\t\t\tEmployee ID not found. \n";
            }
            while(y)
            {
                y.get(d);
                cout<<d;
            }
            y.close();
        }
        else if(u==5)
        {
            Records u;
            u.update_item();
        }
        else if(u==6)
        {
            Records v;
            v.update_emp();
        }
        else if(u==7)
        {
            break;
        }
        else cout<<"\t\t\tInvalid option. Please select one of the available options\n";
    }
}
```

We also have the main() function, where the execution of the program starts. The main functions first display the options to Open or Close the store. If the user selects to open the store, then the start() function is called and the menu is displayed, and various operations are performed. If the user selects to close the store, then the program is terminated.

int main()

```
int main(){
    int ch;
    while(1)
    {
        system("cls");
        cout<<"\n \t\t\t=========================\n";
        cout<<"\n\t\t\t  WELCOME TO GROCERY MART\n";
        cout<<"\n \t\t\t=========================\n\n";
        cout<<"\n\t\t\t  1. OPEN STORE";
        cout<<"\n\t\t\t  2. CLOSE STORE";
        cout<<"\n\n\t\t\t Enter Your choice: ";
        cin>>ch;
        switch(ch){
            case 1:
            system("cls");
                start();
                break;
            case 2:
                exit(0);
                break;
            default:
                cout<<"Choice Unvalid, try again\n";
                break;
        }
    }
    return 0;
}
```

Results

Opening Menu:

```
=========================

  WELCOME TO GROCERY MART

=========================


  1. OPEN STORE
  2. CLOSE STORE

 Enter Your choice: 1
```

Main Menu:

```
=============================

    WELCOME TO GROCERY MART

=============================


1. Add Entries

2. Show profit

3. Search Product Details

4. Search Employee Details

5. Modify Product Details

6. Modify Employee Details

7. Exit


Select your choice : 1

PROFIT = 0

1. Add Products
2. Add employees
3. Exit
Enter choice : 1
```

Adding Products:

```
Number of products to be added : 1

Enter product name :Ketchup
Enter product id : P101
Enter cost price of product : 25
Enter selling price of product : 30
Total products sold quantity : 100

1. Add Products
2. Add employees
3. Exit
Enter choice : 2
```

Adding Employee:

```
Enter Salary : 3000
Enter Number of Employees : 1

Enter Employee name : John
Enter Employee ID : EMP01

1. Add Products
2. Add employees
3. Exit
Enter choice : 3
```

Searching Product:

```
=============================

    WELCOME TO GROCERY MART

=============================


1. Add Entries

2. Show profit

3. Search Product Details

4. Search Employee Details

5. Modify Product Details

6. Modify Employee Details

7. Exit


Select your choice : 3

Enter ID to search : P101

Product Name : Ketchup
Product ID : P101
Cost price of product : 25
Selling price of product : 30
Quantity : 100
Total cost: 2500
Sell : 3000
```

Searching for Employee:

```
==============================

    WELCOME TO GROCERY MART

==============================


1. Add Entries

2. Show profit

3. Search Product Details

4. Search Employee Details

5. Modify Product Details

6. Modify Employee Details

7. Exit


Select your choice : 4

Enter employee ID to search : EMP01

Number of working employees : 1
Employee Salary : 3000
Employee Name : John
Employee ID : EMP01
```

Modify Product Details:

```
==============================

    WELCOME TO GROCERY MART

==============================


1. Add Entries

2. Show profit

3. Search Product Details

4. Search Employee Details

5. Modify Product Details

6. Modify Employee Details

7. Exit


Select your choice : 5

Product ID to modify : P101
Product found!

Update product name : Mayo
Update Cost Price per product : 25
Update Selling Price per product : 30
Update total products sold quantity : 150
```

Modify Employee Details:

```
==============================

    WELCOME TO GROCERY MART

==============================


1. Add Entries

2. Show profit

3. Search Product Details

4. Search Employee Details

5. Modify Product Details

6. Modify Employee Details

7. Exit


Select your choice : 6

Enter employee ID to modify : EMP01
Update Employee Name : Jake
```

Exiting the Program.

```
                        ==========================

                          WELCOME TO GROCERY MART

                        ==========================


                          1. OPEN STORE
                          2. CLOSE STORE

                         Enter Your choice: 2
PS C:\Users\adivi\Documents\Programs\CPP\Project>
```

Concluding Remarks

In the process of making this project, we have understood the object-oriented concepts of C++ and how to implement them. This program can be used for maintaining a record of sales and profits and the employee records in a file system in a PC. This program follows the object-oriented approach using classes, objects, and inheritance to model a real-world problem.