**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Santhibastawad Road, Machhe**

**Belagavi - 590018, Karnataka, India**

****

**FS LAB MINI PROJECT**

**ON**

*“***CANTEEN MANAGEMENT SYSTEM***”*

Submitted in the partial fulfilment of the requirements for the **FILE STRUCTURES LABORATORY WITH MINI PROJECT(17ISL68)** course of the 6th-semester

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION SCIENCE AND ENGINEERING**

**For the Academic Year 2018-2019**

**Submitted by**

**Sushma Patil 1JS17IS079**

**Soundarya V Yadav 1JS17IS072**

**Under the Guidance of**

**Mrs.Sahana V,Asst.Prof**

**and Mrs.Sowmya K.N,Asst.Prof**

**Dept. of ISE, JSSATE**

****

**2019-2020**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**JSS ACADEMY OF TECHNICAL EDUCATION**

**JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060**

**JSS MAHAVIDYAPEETHA, MYSURU**

**JSS ACADEMY OF TECHNICAL EDUCATION**

**JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

****

**CERTIFICATE**

This is to certify that the FILE STRUCTURES mini project entitled **“CANTEEN MANAGEMENT SYSTEM”** is a work carried out by **Sushma Patil [1JS17IS079], Soundarya V Yadav [1JS17IS072]** inpartial fulfilment for the award of degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University Belagavi during the year 2019-2020.

**Signature of the Guide Signature of the Guide Signature of the Guide**

|  |
| --- |
| **Mrs.Sahana V Mrs.Sowmya K.N Dr.Dayananda P** |
| **Assistant Professor Assistant Professor Assoc.Prof.& Head** |
| Dept. of ISE Dept. of ISE Dept. of ISE |
| JSSATE, Bengaluru JSSATE,Bengaluru JSSATE,Bengaluru |

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So with gratitude, we acknowledge all those whose guidance and encouragement crowned our effort with success.

First and foremost we would like to thank his **Holiness Jagadguru Sri Shivarathri Deshikendra Mahaswamiji** and **Dr. Mrityunjaya V Latte**, Principal, JSSATE, Bangalore for providing an opportunity to carry out the Project Work as a part of our curriculum in the partial fulfilment of the degree course.

We express our sincere gratitude for our beloved Head of the department, **Dr. Dayananda P**, for his co-operation and encouragement at all the moments of our approach.

It is our pleasant duty to place on record our deepest sense of gratitude to our respected guide **Mrs.Sahana V,Assistant Professor,** for the constant encouragement, valuable help and assistance in every possible way.

We are thankful to the Project Coordinators **Mrs. Sowmya K N,** Asst. Professor and **Mrs. Sudha P R** Asst. Professor, for their continuous co-operation and support.

We would like to thank all ISE Department Teachers, non teaching staff and Library staff for providing us with their valuable guidance and for being there at all stages of our work.

**Sushma Patil Soundarya V Yadav**

**[1JS17IS079] [1JS17IS072]**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **TABLE OF CONTENTS** | **PAGE**  **No.** |
| 1. | ABSTRACT | 1 |
| 2. | INTRODUCTION | 2 |
| 3. | OVERVIEW OF FILE STRUCTURE   * TYPES OF FILES * COMPARISION BETWEEN FILE STRUCTURES, DATA STRUCTURE AND DBMS | 3 |
| 4. | INTRODUCTION TO INDEXING   * INDEXING METHODS * TYPES OF INDEXES * ADVANTAGES AND DISADVANTAGES OF INDEXING | 7 |
| 5. | PROBLEM STATEMENT | 11 |
| 6. | OBJECTIVES | 11 |
| 7. | FEATURES | 12 |
| 8. | SYSTEM REQUIREMENTS   * SOFTWARE REQUIREMENTS * HARDWARE SPECIFICATION | 13 |
| 9. | IMPLEMENTATION   * SOURCE CODE | 14 |
| 10. | RESULTS AND SCREENSHOTS | 31 |
| 11. | APPLICATIONS | 38 |
| 12. | CONCLUSION | 38 |
| 13. | REFERENCES   * BOOK REFERENCES * WEB REFERENCES | 39 |

**ABSTRACT**

This project is aimed at developing a file management system that depicts the canteen management system. Using this code, canteen can improve the efficiency of their services. Canteen management system is one of

the applications to improve the efficiency of the canteen to maintain the records of employee and customers order.

The main aim of the creating the Canteen Management was to Automate the existing traditional outdated manual management process. In this application named Canteen management system project in C++, owner can add employee details, delete employee details, modify food items, delete food items, also allows the customers to place the order, generates the bill. File handling has been used to perform all these operations.

In this project, the customer can place the order. The order will contain details such as the name of the food, food id and the quantity of the food, rating. This order is then stored in the file along with the previous orders. The user can view the bill generated. This code allows the Employee to modify the cost of the food, quantity of the food. We can also view the employee details that includes employee name, employee age, employee salary.

**INTRODUCTION**

The “Canteen Management System” has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by the existing system. Moreover, this system is designed for the particular need of the canteen to carry out operation in smooth and effective manner.

Every canteen, whether it’s for student or employee has challenge to overcome and managing the information of employees, company canteen, payment bills, canteen records. Every Canteen Management System has different needs. Therefore, we design exclusive employee/student management systems that are adapted to your managerial requirement. This is designed to assist in strategic planning, and will help you ensure that your organization/college is equipped with right level of information and details for your future goal. Also, for those busy executives who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all time. These systems will ultimately allow you to better manage resource.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error massage while entering invalid data, no formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Canteen management system, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping, thus it will help canteen in better utilization of resource.

C.M.S comes with feature of User Account to control the access and maintain security. Hence, only authorized user can use the software which reduces the risk of data stealing and data manipulation by unauthorized user. The user authentication is of two types (owner login & employee login), the owner login has supreme control over employee login. Owner can add/remove/ employee as well as manipulate the details of the employee. Another main feature of C.M.S. is Inventory. This feature

allow user to see the status of the products present in the warehouse of the canteen. It also allows user to manipulate the quantity as well as cost of the products in the inventory. The order feature of C.M.S. allow user to select the products that customer orders and display error if the quantity of product order is not available in Inventory. The Statistic feature of C.M.S. allow user to keep track of record of sales done in canteen. The user gets digital Bill of the products bought by the customer with the help of the Billing feature of the C.M.S.

C.M.S is based on concept of file handling in C++. So, our data is stored in traditional files. The data can be recovered if incase it gets lost. Concept of object- oriented programming provides easiness while programming software. The C++ concepts provides security. Our project runs on C++ compiler which can be easily installed on any system, hence provides portability.

The C.M.S has a well design interface which make a non-technical user friendly with the software. This also make decrease the load of the person involve in existing mana system. Due to file handing concept of C++ no online services are needed by user, this reduces the cost of the software services, hence making our software economical. The software is error free and doesn’t require any special/expensive technical maintenance, this increase the durability of the software. Due to requirement of vey less execution time, the software is very quick and which increases the efficiency of the software and time management. It’s very easy to update information in C.M.S.

The main goal of creating the Canteen Management was to Automate the existing traditional outdated manual management process. All efforts are made to keep the software simple to use and easy to maintain.

# OVERVIEW OF FILE STRUCTURES

1. **TYPES OF FILES**

As we know that Computers are used for storing the information for a Permanent Time or the Files are used for storing the Data of the users for a Long time Period. And the files can contain any type of information which means that they can store text, any images or pictures or data in any Format. So that there must be Some Mechanism which are used for storing the information, Accessing the information and also Performing Some Operations on the files.

When we Store a File in the System, then we have to specify the name and the type of file. The name of file will be any valid Name and Type means the application with which the file is linked.

When we say that Every File has some type, it means that every file belongs to a special type of Application software. When we provide a name to a file, we also specify the extension of the File because the system will retrieve the Contents of that file into that Application Software. For example, if there is a file which contains some paintings then this will open in software.

1. **Ordinary Files or Simple File**: Ordinary Files may belong to any type of Application. For example, notepad, paint, C program, music[mpeg], etc. All the Files that are created by a user are ordinary files. Ordinary files are used for storing information about the user programs. With the help of ordinary files, we can store the information which contain text, database, images or any other type of information.
2. **Directory files:** The Files those are Stored into a particular directory or folder. These are known as Directory Files because they belong to a particular director. For example, a folder named images contains numerous files, and the files contained within the directory images are known as directory files.
3. **Special Files:** The Special Files are those which are not created by the user or the files which are necessary to run a system. These are the files that are created by the system. All the Files of an Operating System are referred to as special files. There

are many types of special files and they are, system files or windows Files, input/output files. System files are stored using the .sys extension.

1. **FIFO Files:** The First in First Out Files are used by the System for Executing the Processes in a particular order. Which means that the Files which come first, will be executed first and the System Maintains a Order known as Sequence Order. When a user requests for a service from the system, then the requests of the users are Arranged into some files and all the Requests will be performed by the system, using some sequential order known as the FIFO order.

## Types of File Operations

Files are not created only for reading, we can also Perform some other operations on files, they are:

1. Read Operation: This is meant to read the content which is stored in the files.
2. Write Operation: This operation is used to write some data into the files.
3. Rename or Change the Name of File.
4. Copy the File from one Location to another.
5. Sorting or Arranging the Contents of File.
6. Move or Copy the File from One Place to Another.
7. Delete a File
8. Execute the file (this displays the output of the file).

# 2.COMPARISION BETWEEN FILE STRUCTURES, DATA STRUCTURES AND DATABASE MANAGEMENT SYSTEMS

File Structures is the Organization of Data in Secondary Storage Device in such a way that minimize the access time and the storage space. A File Structure is a combination of representations for data in files and of operations for accessing the data. A File Structure allows applications to read, write and modify data.

In computer science, a data structure is a data organization, management, and storage format that enables efficient access and modification. More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

Data structures serve as the basis for abstract data types (ADT). The ADT defines the logical form of the data type. The data structure implements the physical form of the data type.

Different types of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. For example, relational databases commonly use B-tree indexes for data retrieval, while compiler implementations usually use hash tables to look up identifiers.

Data structures provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Usually, efficient data structures are key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. Data structures can be used to organize the storage and retrieval of information stored in both main memory and secondary memory.

A database is an organized collection of data, generally stored and accessed

electronically from a computer system. Where databases are more complex they are often developed using formal design and modelling techniques.

The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages

# INTRODUCTION TO INDEXING

**1. METHODS OF INDEXING**

An index is a collection of data entries which is used to locate a record in a file. Index table record in a file consist of two parts, the first part consists of value of prime or non-prime attributes of file record known as indexing field and, the second part consists of a pointer to the location where the record is physically stored in memory. In general, index table is like the index of a book, that consists of the name of topic and the page number. During searching of a file record, index is searched to locate the record memory address instead of searching a record in secondary memory. On the basis of properties that affect the efficiency of searching, the indexes can be classified into two categories.

1. Ordered indexing
2. Hashed indexing

Ordered Indexing: In ordered indexing, records of file are stored in some sorted order in physical memory. The values in the index are ordered (sorted) so that binary search can be performed on the index. Ordered index can be divided into two categories.

1. Dense indexing
2. Sparse indexing

## Dense and Sparse Indexing

**Dense index** : In dense indexing there is a record in index table for each unique value of the search-key attribute of file and a pointer to the first data record with that value. The other records with same value of search-key attribute are stored sequentially.

**Advantages of Dense index**

* + It is efficient technique for small and medium sized data file.
  + Searching is comparatively fast and efficient. **Disadvantages of Dense index**
  + Index table is large and require more memory space.
  + Insertion and deletion is comparatively complex.
* In-efficient for large data files

**Sparse index** : On contrary, in sparse indexing there are only some records in index table for unique values of the search-key attribute of file and a pointer to the first data record with that value. To search a record in sparse index we search for a value that is less than or equal to value in index for which we are looking. After getting the first record, linear search is performed to retrieve the desired record. There is at most one sparse index since it is not possible to build sparse index that is not clustered

**Advantages of sparse index**

* + Index table is small and hence save memory space (especially in large files)
  + Insertion and deletion is comparatively easy. **Disadvantages of sparse index**
  + Searching is comparatively slower, since index table is searched and then linear search is performed inside secondary memory.

## Clustered and Non-Clustered Indexes

**Clustered index:** In clustered, index file records are stored physically in order of non- prime key attribute that does not have a unique value for each record. The non-prime key field is known as clustering field and index is known as clustering index. It is same as dense index. A file can have at most one clustered index as it can be clustered on at most one search key attribute. It may be sparse.

**Non-Clustered index**: An index that is not clustered is known as non-clustered index. A data file can have more than one non-clustered index.

## Primary and Secondary Index

**Primary index**: A primary index consists of all prime-key attributes of a table and a pointer to physical memory address of the record of data file. To retrieve a record on the basis of all primary key attributes, primary index is used for fast searching. Binary search is done on index table and then directly retrieve that record from physical memory. It may be sparse.

**Advantages of Primary index**

* + Search operation is very fast.
  + Index table record is usually smaller.
  + A Primary index is guaranteed not to duplicate. **Disadvantages of Primary index**
  + There is only one primary index of a table. To search a record on less than all prime-key attributes, linear search is performed on index table.
  + To create a primary index of an existing table, record should be in some sequential order otherwise database is required to be adjusted.

**Secondary index:** A secondary index provides a secondary means of accessing a data file. A secondary index may be on a candidate key field or on non-prime key attributes of a table. To retrieve a record on the basis of non-prime key attributes, secondary index can be used for fast searching. Secondary index must be dense with a index entry for every search key value and a pointer to every record in a file.

**Advantages of secondary index**

* + Improve search time if search on non-prime key attributes.
  + A data file can have more than one secondary index. **Disadvantages of Secondary index**
  + A secondary index usually needs more storage space.
  + Search time is more than primary index.
* They impose a significant overhead on the modification of database.

## Single and Multilevel Indexes

**Single level index**: A single stage index for a data file is known as single level index. A single level index cannot be divided. It is useful in small and medium size data files. If the file size is bigger, then single level, indexing is not an efficient method. Searching is faster than other indexes for small size data files

**Multilevel index** : A single index for a large size data file increases the size of index table and increases the search time that results in slower searches. The idea behind multilevel indexes is that, a single level index is divided into multiple levels, which reduces search time. In multilevel indexes, the first level index consists of two fields, the first field consists of a value of search key attributes and a second field consists of a pointer to the block (or second level index) which consists that value and so on. To search a record in multilevel index, binary search is used to find the largest of all the small value or equal to the one that needs to be searched. The pointer points to a block of the inner index. After reaching to the desired block, the desired record is

searched (in case of two-level indexing) otherwise again the largest of the small values or equal to the one that needs to be searched and so no.

Hashed Indexing To overcome the disadvantage of ordered indexing,a hash index can be created for a data file. Hashing allow us to avoid accessing an index structure. A hashed index consists of two fields, the first field consists of search key attribute value and second field consists of pointer to the hash file structure. Hashed indexing is based on values of records being uniformly distributed using a hashed function.

# 2. ADVANTAGES AND DISADVANTAGES OF INDEXING

## ADVANTAGES

* Quite easy to process,
* With proper selection of a key field, records in a large file can be searched and accessed in very quickly.
* Any field of the records can be used as the key. The key field can be numerical or alphanumerical.

## DISADVANTAGES

* Extra data structures have to be maintained (the COBOL run-time modules take care of these and it is not the programmers' concern). These extra data structures maintained on the disk can use up much disk space, especially for long key values.
* The indexed files have to be reorganized from time time to get rid of deleted records and improve performance that gets gradually decreased with addition of new records

# PROBLEM STATEMENT

The Challenges encountered by the manual system in canteens is efficiency and customer satisfaction. The experience of ordering in most canteens is not pleasant for customers. Customers have to make long queues before placing the order and when the order is placed, they have to wait near the counter until the order is prepared. Another problem is efficiency that food canteen should maintain in their standard operations and keep with quality of their product and services no matter how much crowd is present in canteen but they have to maintain efficiency as well as quality of food.

The verbal communication between the customer and the cashier for placing an order and the information about bill may also result in errors usually when the place is crowded and noisy, miscommunications are common. This code helps in solving all these faults.

# OBJECTIVES

The objectives of the Canteen Management System are:

1. It provides “better and efficient” services to Customers.
2. Reduce the workload of employee.
3. Faster retrieval of information about the Employee.
4. It helps the Customers to place the order easily.

# FEATURES

Listed below are the main features of this C++ project: **OWNER LOGIN**

* + Add Employee details: This feature allows users to add a new employee information to the canteen. Information such as name of the employee, employee age and employee salary should be provided, and the data is stored in file.
  + Display Employee details: This feature will list the details of all the employees working in the canteen.
  + Delete Employee details: This feature allows users to delete the existing employee details from the file.

## EMPLOYEE LOGIN

* + Order Food: Order food feature will allow the employees to take the order from the customers efficiently.
  + Search Item: Search item will help the customers to search for the particular food item they are looking for.
* Delete Item: Employees can delete the food item which is not available in the canteen.

# SYSTEM REQUIREMENTS

**SOFTWARE REQUIREMENTS:**

* GNU C and C++ compiler
* Microsoft Operating System (WINDOWS 10)

**HARDWARE SPECIFICATIONS**

* Processor: 1 gigahertz (GHz) or faster processor or SoC.
* RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit.
* Hard disk space: 16 GB for 32-bit OS 20 GB for 64-bit OS.
* Graphics card: DirectX 9 or later with WDDM 1.0 driver.

**IMPLEMENTAION**

**SOURCE CODE:**

#include<iostream> #include<iostream> #include<fstream> #include<windows.h> #include<dos.h> #include<stdio.h> #include<cstdlib> #include<string> #include<conio.h>

using namespace std;

COORD coord= {0,0};

void gotoxy (int x, int y)

{

coord.X=x; coord.Y=y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

class store{

public:

void storepage();

{

storepageswitch(); int can\_flag=0,no=0;

char can\_ind[5];

struct canteen{

char ind[5], item\_id[5],

item\_name[20], cost[20], quantity[20],

rating[5];

}can[20];

struct secind{

char item\_name[20], item\_id[5],

ind[5];

}sec[20],found[20];

struct index{

char item\_id[20],ind[20];

}in[20],temp; void sort\_index()

{

int i,j;

for(i=0;i<no-1;i++) for(j=0;j<no-i-1;j++)

if(strcmp(in[j].item\_id,in[j+1].item\_id)>0)

{

temp =in[j]; in[j]=in[j+1]; in[j+1]=temp;

}

}

void sort\_secind()

{

int i,j;

secind temp; for(i=0;i<no-1;i++) for(j=0;j<no-i-1;j++)

if(strcmp(sec[j].item\_name,sec[j+1].item\_name)>0)

{

temp =sec[j]; sec[j]=sec[j+1]; sec[j+1]=temp;

}

else if(strcmp(sec[j].item\_name,sec[j+1].item\_name)==0)

{

if(strcmp(sec[j].item\_id,sec[j+1].item\_id)>0)

{

temp =sec[j]; sec[j]=sec[j+1]; sec[j+1]=temp;

}

}

}

void ret\_rec(char \*ind)

{

int flag=0;

for(int i=0;i<no;i++)

{

if(strcmp(can[i].ind,ind)==0)

{

strcpy(can\_ind,ind); can\_flag=1; cout<<"Item found\n";

cout<<can[i].item\_id<<"|"<<can[i].item\_name<<"|"<<can[i].cost<<"|"<<can[i].quantity<<"

|"<<can[i].rating<<"\n";

flag=1;

}

}

if(!flag)

{

cout<<"Item search failed\n";

}

}

void search\_index(char \*item\_id)

{

int flag=0; for(int i=0;i<no;i++)

{

}

}

if(!flag)

{

}

if(strcmp(in[i].item\_id,item\_id)==0)

{

ret\_rec(in[i].ind); flag=1;

cout<<"Index search failed\n";

}

void search(char \*it\_name)

{

char it\_id[20]; int flag1=0; int k=0,i;

for(i=0;i<no;i++)

{

if(strcmp(sec[i].item\_name,it\_name)==0)

{

strcpy(found[k].item\_id,sec[i].item\_id); strcpy(found[k].item\_name,sec[i].item\_name); strcpy(found[k].ind,sec[i].ind);

flag1=1;

k++;

}

}

if(!flag1)

{

cout<<"sec ind search failed\n"; return;

}

cout<<"\n Records matching are:"; for(i=0;i<k;i++)

cout<<"\n"<<found[i].item\_name<<" "<<found[i].item\_id; cout<<"\n Choose an Item id";

cin>>it\_id; flag1=0; for(i=0;i<k;i++)

if(strcmp(found[i].item\_id,it\_id)==0)

{

search\_index(it\_id); flag1=1;

}

if(!flag1)

{

cout<<"wrong item id search failed\n";

}

}

void orderfood()

{

system("CLS");

gotoxy(40,1);cout<<"\nJSSATE CANTEEN"; gotoxy(40,2);

int choice,i,flag1,flag; fstream file1,file2,file3; char ind[5],

it\_id[20], it\_name[20], item\_id[5],

item\_name[20], cost[20], quantity[20],

rating[5];

file3.open("index.txt",ios::out); file1.open("canteen1.txt",ios::app|ios::out); file2.open("secind.txt",ios::out);

if(!file1)

{

}

int n;

cout<<"file couldnt be added"; exit(0);

cout<<"\nEnter no of food items\n"; cin>>n;

for(int i=no;i<no+n;i++)

{

label: cout<<"\nEnter "<<i+1<<" item name:\n"; cout<<"Enter Item id:"; cin>>can[i].item\_id; cout<<"Enter Item Name:"; cin>>can[i].item\_name; cout<<"Enter Cost:"; cin>>can[i].cost; cout<<"Enter Quantity:"; cin>>can[i].quantity; cout<<"Enter rating:"; cin>>can[i].rating;

int q=search\_item(can[i].item\_id,i); if(q==1)

{

cout<<"Duplicate entry, Enter again\n"; goto label;

}

file1<<i<<"|"<<can[i].item\_id<<"|"<<can[i].item\_name<<"|"<<can[i].cost<<"|"<<can[i].qua ntity<<"|"<<can[i].rating<<"\n";

}

file1.close(); no=no+n;

file1.open("canteen1.txt",ios::in); for(int i=0;i<no;i++)

{

file1.getline(ind,5,'|'); file1.getline(item\_id,5,'|'); file1.getline(item\_name,20,'|'); file1.getline(cost,20,'|'); file1.getline(quantity,20,'|'); file1.getline(rating,5,'\n'); strcpy(can[i].ind,ind);

strcpy(sec[i].ind,ind);

strcpy(in[i].ind,ind); strcpy(sec[i].item\_name,item\_name); strcpy(sec[i].item\_id,item\_id); strcpy(in[i].item\_id,item\_id);

}

"<<sec[i].ind<<endl;

sort\_index(); sort\_secind();

cout<<"\nThe sorted secondary index contents are:\n"; for(i=0;i<no;i++)

cout<<sec[i].item\_name<<" "<<sec[i].item\_id<<"

cout<<"\nThe sorted index contents are:\n"; for(i=0;i<no;i++)

cout<<in[i].item\_id<<" "<<in[i].ind<<endl; for(int i=0;i<no;i++)

file2<<sec[i].item\_name<<"|"<<sec[i].item\_id<<"|"<<sec[i].ind<<"\n";

for(int i=0;i<no;i++) file3<<in[i].item\_id<<"|"<<in[i].ind<<"\n";

cout<<" ";getch(); storepage();

file1.close(); file2.close(); file3.close();

cout<<endl<<" PRESS ANY KEY TO CONTINUE";

}

void bill()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);

int choice,i,flag1,flag; fstream file1,file2,file3;

char ind[5], it\_id[20], it\_name[20], item\_id[5],

item\_name[20], cost[20], quantity[20],

rating[5];

fstream f1; f1.open("canteen1.txt",ios::in); if(!f1)

{

cout<<"File not found"; exit(0);

}

cout<<"\nID NAME COST QUANTITY RATINGS\t\n\n"; i=0;

while(i!=no)

{

f1.getline(ind,5,'|');

f1.getline(item\_id,5,'|'); f1.getline(item\_name,20,'|'); f1.getline(cost,20,'|'); f1.getline(quantity,20,'|'); f1.getline(rating,5,'\n');

cout<<"\n"<<item\_id<<"\t"<<item\_name<<"\t\t"<<cost<<"\t"<<quantity<<"\t"<<rating<< "\t\n";

i++;

}

int totalcost=0;

cout<<"\nTotal number of food items ordered :"<<i<<"\n"; for(int f=i;f<0;f--)

{

totalcost = totalcost+(cost[f]\*quantity[f]); cout<<totalcost;

}

cout<<"\nYou will have to pay Rs.70"<<totalcost<<endl; cout<<"Thank you, JSSATE\n";

f1.close();

cout<<endl<<" PRESS ANY KEY TO CONTINUE";

cout<<" ";getch(); storepage();

}

void deletefood()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2); char it\_name[20];

cout<<" ";getch(); storepage();

}

cout<<"\nEnter Item name to be deleted\n"; cin>>it\_name;

del(it\_name);

cout<<endl<<" PRESS ANY KEY TO CONTINUE";

storepageswitch()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);cout<<" "<<endl<<endl;

int choice;

cout<<"\n\nJSSATE CANTEEN MANAGEMENT\n";

cout<<"\n1.Order Food\n2.Search Item\n3.Delete Item\n4.Generate Bill\n5.Exit\n";

cout<<"Enter your choice\n"; cin>>choice;

switch(choice)

{

case 1:

orderfood(); break;

storepageswitch(); case 2:{

foodsearch(); break;

}

case 3:{

}

case 4:{

deletefood();

break;

}

case 5:{

bill(); break;

cout<<"JSSATE CANTEEN \n";

cout<<"Thank You\n"; break;

}

default:

cout<<"Invalid Choice\n";

cout<<" ENTER CHOICE"<<endl; cout<<" ";cin>>choice;

break;

}

}

};

void employee::addemp()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);cout<<" "<<endl<<endl;

ofstream newemployee("EMPLOYEE.txt",ios::app); cout<<" ENTER NAME OF EMPLOYEE"<<endl;

cout<<" ";cin>>name; cin.sync();

cout<<" ENTER AGE OF EMPLOYEE"<<endl;

cout<<" ";cin>>age;

cout<<" ENTER SALARY OF EMPLOYEE"<<endl;

cout<<" ";cin>>sal;

newemployee<<name<<' '<<age<<' '<<sal<<endl; newemployee.close();

cout<<endl<<" EMPLOYEE ADDED"<<endl; cout<<endl<<" PRESS ANY KEY TO CONTINUE";

cout<<" ";getch(); emppage();

}

void employee::displayemp()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);cout<<" "<<endl<<endl;

ifstream employee("EMPLOYEE.txt");

cout<<" EMPLOYEE - AGE - SALARY"<<endl; cout<<" "<<endl;

while (employee>>name>>age>>sal)

{

cout<<" "<<name<<" - "<<age<<" - "<<sal<<endl ;

}

employee.close();

cout<<endl<<" PRESS ANY KEY TO CONTINUE"<<endl;

cout<<" ";getch(); emppage();

}

void employee::removeemp()

{

system("CLS"); gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);cout<<" "<<endl<<endl;

char tname[50];

ifstream emp1("EMPLOYEE.txt"); ofstream emp2("temp.txt");

cout<<" ENTER THE NAME OF EMPLOYEE WISH TO REMOVE"<<endl;

cout<<" ";cin>>tname; while(emp1>>name>>age>>sal)

{

if(strcmp(name,tname)!=0)

{

emp2<<name<<' '<<age<<' '<<sal<<endl;

}

}

emp1.close();

emp2.close(); remove("EMPLOYEE.txt"); rename("temp.txt","EMPLOYEE.txt");

cout<<endl<<" EMPLOYEE REMOVED"<<endl; cout<<endl<<" PRESS ANY KEY TO CONTINUE"<<endl;

cout<<" ";getch(); emppage();

}

void loginpage()

{

loginpage:; system("CLS");

gotoxy(40,1);cout<<"JSSATE CANTEEN";

gotoxy(40,2);cout<<" "<<endl<<endl;

cout<<" 1. OWNER LOGIN"<<endl; cout<<" 2. EMPLOYEE LOGIN"<<endl;

cout<<" 3. EXIT"<<endl<<endl; cout<<" ENTER CHOICE"<<endl; cout<<" ";cin>>ch; loginpageswitch();

}

};

void login::ownerlogin()

{

while(pass!="OWN")

{

pass="";

cout<<endl<<" ENTER OWNER PASSWORD"<<endl; cout<<" ";c=\_getch();

while(c!=13)

{

pass.push\_back(c); cout<<"\*"; c=getch();

}

if(pass=="OWN")

{

cout<<endl<<" OWNER ACCESS GRANTED"<<endl; cout<<endl<<" PRESS ANY KEY TO CONTINUE"<<endl;

cout<<" ";getch(); own();

}

else

{

cout<<endl<<" INVALID PASSWORD"<<endl;

}

}

}

void login::employeelogin()

{

while(pass!="EMP")

{

pass="";

cout<<" ENTER EMPLOYEE PASSWORD"<<endl;

cout<<" ";c=\_getch(); while(c!=13)

{

pass.push\_back(c); cout<<"\*"; c=getch();

}

if(pass=="EMP")

{

cout<<endl<<" EMPLOYEE ACCESS GRANTED"<<endl;

cout<<endl

<<" PRESS ANY KEY TO CONTINUE"<<endl;

cout<<" ";getch(); emp();

}

else

{

cout<<endl<<" INVALID PASSSWORD"<<endl;

}

}

}

# OUTPUT AND SCREENSHOTS

**MAIN MENU:**





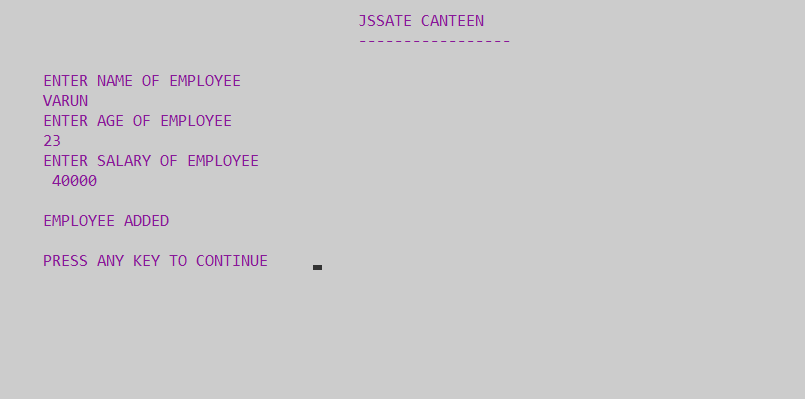
**OWNER LOGIN:**



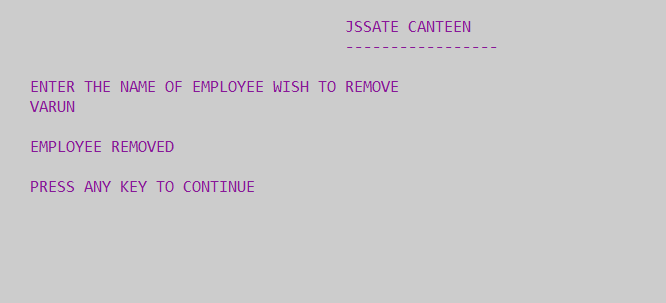
**OWNER OPERATIONS:**



**ADDING NEW EMPLOYEE:**



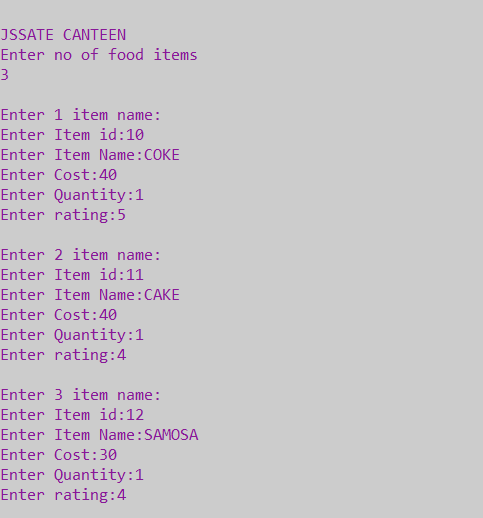
**REMOVING EMPLOYEE:**



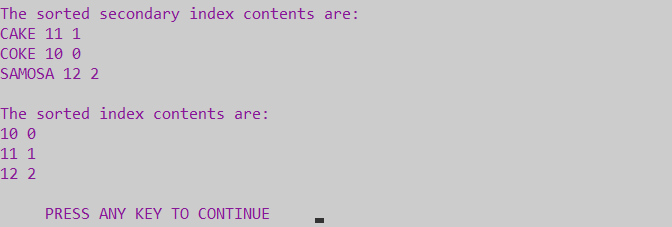
**EMPLOYEE LOGIN:**



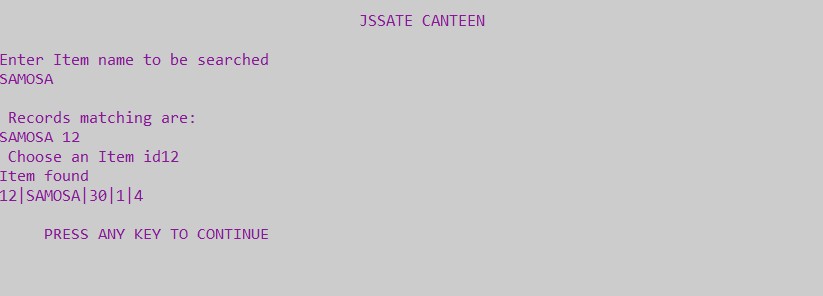
**ORDER DETAILS:**



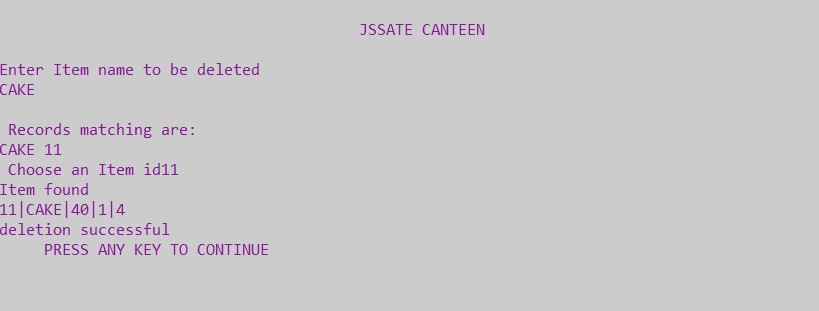
**SORTED INDEX FILE:**

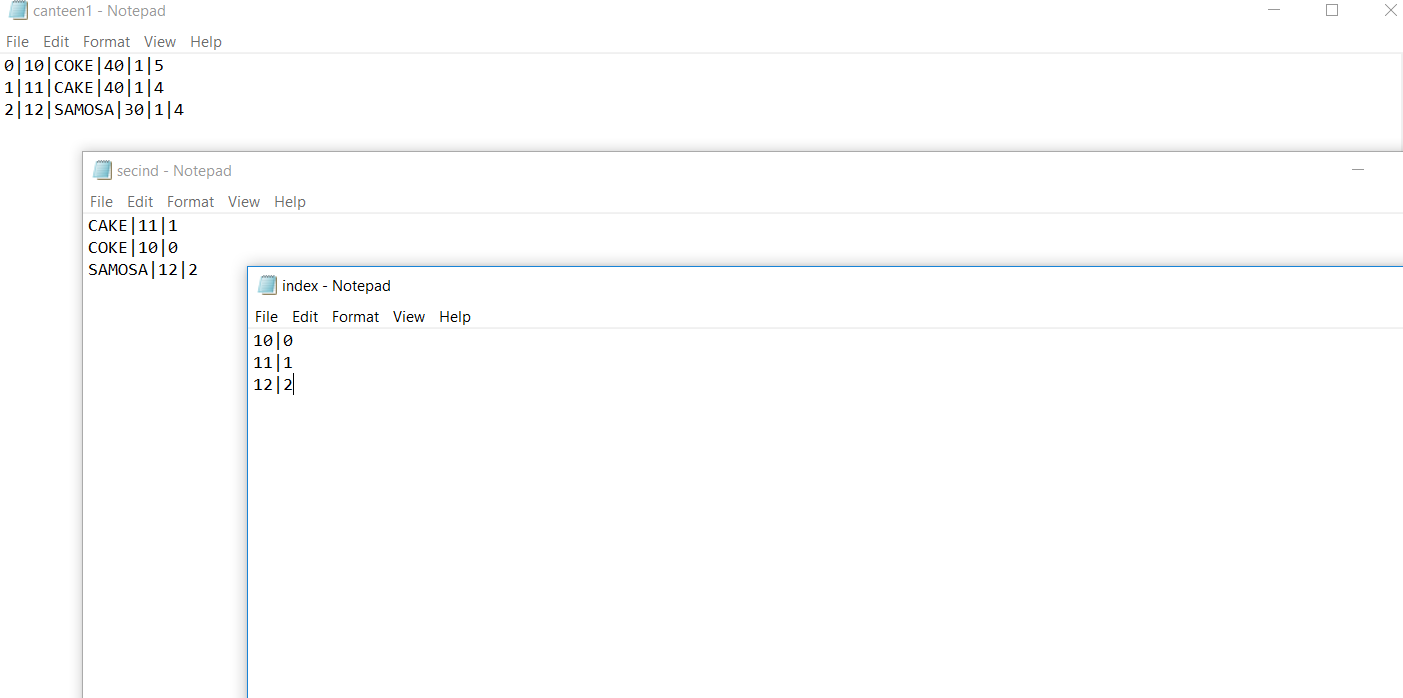


**SEARCHING:**



**DELETING:**





**APPLICATIONS**

The main application of this project is to manage the canteen. It automates the entire process which was done manually till now.

In automation of any operation we make a system which do work automatically as the respective events occurs, for which it is meant the system facilitates the customers to order the food from the canteen without any complexity.

# CONCLUSION

This Canteen Management System is used to overcome the entire problem which they are facing currently, and making the manual system to a computerized system. The customer does not have to wait in long queues. This system reduces the amount of paper work. It saves time. The details of the food items is stored in the form of a record, this record is in turn stored in a file. These records can be accessed easily and it easier to handle the records.

# BOOK REFERENCES

* File structures-an Object-Oriented Approach with C++ Michael J Folk, Bill Zoellick, Greg Riccradi, 3rdEdition, PearsonEducation, 1998.
* File structures using C++ by K.R.Venugopal, K.G.Srivinas, P.M. Krishnaraj, Tata McGraw-Hill, 2008.

**WEB REFERENCES**

* [www.wikipedia.org](http://www.wikipedia.org/)
* [http://www.w3schools.com](http://www.w3schools.com/)
* https://[www.udemy.com](http://www.udemy.com/)
* https://[www.tutorialspoint.com](http://www.tutorialspoint.com/)