**ABSTRACT**

This project develops a Departmental Store Management System (DSMS) is commonly found today at most retail store registers. Store merchandise, identified by a price code is checked out by a cashier who then accepts payment for the item(s). A DSMS is either read by a bar code scanner or manually entered by the cashier. At the completion of a sale, a receipt is created for the customer and sales information is collected for the generation of reports at a later time. A DSM system should help you alleviate the daily chores of your business. A DSM system should not interfere or make it harder for you to run your business. It should run parallel to your business operation.

A perfect DSM system should run your business for you, but not in a literal sense where it takes total control. You want one comprehensive package for your entire business, either small or large. And it should scale as much as your business grows. The system also provides for processing the return of purchased items and reimbursement to the customer. While many DSM Systems support multiple terminals that are networked together and interface with external systems (such as inventory control) the primary goal of this system is to develop a self-contained sales terminal application that supports the purchase and return of store merchandise.

**CHAPTER 1**

**INTRODUCTION**

DSMS is an abbreviation for Departmental Store Management System. The term is applicable to a retail shop or store, the checkout/cashier counter in the store, or a location where such transactions can occur in this type of environment. It can also apply to the actual Departmental Store Management (DSM) System Hardware & Software including but not limited to: electronic cash register systems, touch-screen display, barcode scanners, receipt printers, scales and pole displays. Departmental store management system are utilized in many different industries, ranging from restaurants, hotels & hospitality businesses, nail/beauty salons, casinos, stadiums, and let's not forget - the retail environments. In the most basic sense, if something can be exchanged for monetary value - a Departmental Store Management System can be used

This is a rather broad definition that can include merchandising aids, displays and the methods used to enable transactions. In our case, we're talking about the hardware and software that runs both the front counter and back-office operations of a business. A Departmental Store Management System (DSMS) is an application that executes a collection of protocols to co-ordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks. This system is good when one is has an established Shop and wants his/her stores to blinked.

One of its advantages is the ability to connect remote use with remote resources in an open (where each component is continually open to interaction with other components) and scalable (the system can easily be altered to accommodate changes in the number of users, resources and computing) way, and can also be larger and more powerful given the combined capabilities of the Departmental Store Management System. Components compared to that of Departmental Store Management System.

**CHAPTER 2**

**AIM, OBJECTIVE AND SCOPE OF THE PROJECT**

**2.1 Aim of the project**

Inventory software programs now on the market let you track usage, monitor changes in unit Costs, calculate when you need to reorder, and analyze inventory levels on an item-by-item basis. You can even control inventory right at the cash register with Departmental Store Management System (DSM) software systems. DSM software records each sale when it happens, so your inventory records are always up-to-date. Better still, you get much more information about the sale than you could gather with a manual system. By running reports based on this information, you can make better decisions about ordering and merchandising. I can analyze sales data, figure out how well all the items on our shelves sell, and adjust purchasing levels accordingly. I can maintain a sales history to help adjust our buying decisions for seasonal purchasing trends.

**2.2 Objective of the project**

Departmental Store Management (DSM) system provides businesses with the ability to computerize, systematize and correlate retail information. Where cash registers, including complex register systems, have limited information collection capacity, DSM systems can gather, store and return detailed reports on inventory trends and customer information. Additionally, DSM systems more easily integrate with numerous sales and ordering systems, including mail or online ordering systems used in conjunction with in-person sales.

**2.3 Scope of this project**

The proposed system is ‘Stores Management System’. This system is GUI based system and is user friendly. Stores Management System is accessible through the internet. Stores are required for the following purposes.

1. Capital works

2. Operation and Maintenances Works

3. Other Commercial activities like hiring equipment etc..,

The ‘Stores Management System package’ is targeted to automate the almost all of the processes mentioned above to reduce the clerical labor of the staff working in Stores both technical and as well as Accounts departments using the software industry’s latest technologies and cost effective tools there by providing the better control to the management by avoiding manual errors etc..,

**CHAPTER 3**

**REQIREMENT SPECIFICATION**

**3.1 System Requirements**

The basic requirements for the development of this mini project are as follows

**3.1.1 Hardware Configuration**

1. Processor: Intel core i3 or above
2. Ram:512 MB
3. Hard disk:20 GB

**3.1.2 Software Configuration**

1. Language: C & C++
2. IDE: DEV C++
3. Documentation tool: Microsoft office 2003 or above

**3.2 Development Environment**

**3.2.1 C++**

**C++** is a general-purpose programming language that was developed as an enhancement of the C language to include object-oriented paradigm. It is an imperative and a **compiled** language[7].

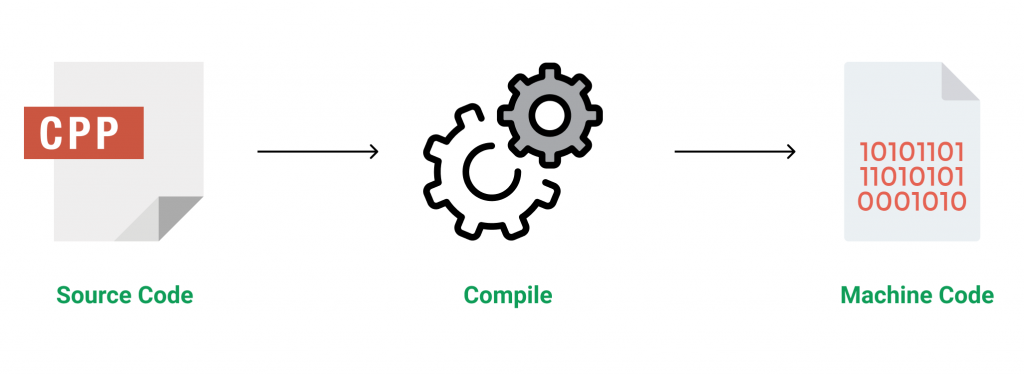


Figure 3.1 Source code compilation

C++ is a middle-level language rendering it the advantage of programming low-level (drivers, kernels) and even higher-level applications (games, GUI, desktop apps etc.). The basic syntax and code structure of both C and C++ are the same.

Some of the features & key-points to note about the programming language are as follows:

* Simple: It is a simple language in the sense that programs can be broken down into logical units and parts, has a rich library support and a variety of data-types.
* Machine Independent but Platform Dependent: A C++ executable is not platform-independent (compiled programs on Linux won’t run on Windows), however they are machine independent.
* Mid-level language: It is a mid-level language as we can do both systems-programming (drivers, kernels, networking etc.) and build large-scale user applications (Media Players, Photoshop, Game Engines etc.)
* Rich library support: Has a rich library support (Both standard ~ built-in data structures, algorithms etc.) as well 3rd party libraries (e.g. Boost libraries) for fast and rapid development.
* Speed of execution: C++ programs excel in execution speed. Since, it is a compiled language, and also hugely procedural. Newer languages have extra in-built default features such as garbage-collection, dynamic typing etc. which slow the execution of the program overall. Since there is no additional processing overhead like this in C++, it is blazing fast.
* Pointer and direct Memory-Access: C++ provides pointer support which aids users to directly manipulate storage address. This helps in doing low-level programming (where one might need to have explicit control on the storage of variables).
* Object-Oriented: One of the strongest points of the language which sets it apart from C. Object-Oriented support helps C++ to make maintainable and extensible programs. i.e. Large-scale applications can be built. Procedural code becomes difficult to maintain as code-size grows.
* Compiled Language: C++ is a compiled language, contributing to its speed.

**3.2.2 File Structure**

In computing, file system or filesystem (often abbreviated to fs) is a method and data structure that the operating system uses to control how data is stored and retrieved.[1] Without a file system, data placed in a storage medium would be one large body of data with no way to tell where one piece of data stopped and the next began, or where any piece of data was located when it was time to retrieve it [5]. By separating the data into pieces and giving each piece a name, the data is easily isolated and identified. Taking its name from the way a paper-based data management system is named, each group of data is called a "file”. The structure and logic rules used to manage the groups of data and their names is called a "file system”.

A file system consists of two or three layers. Sometimes the layers are explicitly separated, and sometimes the functions are combined. The logical file system is responsible for interaction with the user application. It provides the application program interface (API) for file operations — OPEN, CLOSE, READ, etc., and passes the requested operation to the layer below it for processing. The logical file system "manage[s] open file table entries and per-process file descriptors".[5] This layer provides "file access, directory operations, [and] security and protection".

The second optional layer is the virtual file system. "This interface allows support for multiple concurrent instances of physical file systems, each of which is called a file system implementation". The third layer is the physical file system. This layer is concerned with the physical operation of the storage device (e.g. disk). It processes physical blocks being read or written. It handles buffering and memory management and is responsible for the physical placement of blocks in specific locations on the storage medium. The physical file system interacts with the device drivers or with the channel to drive the storage device.

* + 1. **Files Operation**

A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. Six basic file operations. The OS can provide system calls to create, write, read, reposition, delete, and truncate files.

* Creating a file: Two steps are necessary to create a file.
  + Space in the file system must be found for the file.
  + An entry for the new file must be made in the directory
* Writing a file: To write a file, we make a system call specifying both the name of the file and the information to be written to the file. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
* Reading a file: To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. The system needs to keep a read pointer to the location in the file where the next read is to take place.
* Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current-file-position pointer.
* Both the read and write operations use this same pointer, saving space and reducing system complexity.
* Repositioning within a file: The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.
* Deleting a file: To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
* Truncating a file: The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged (except for file length) but lets the file be reset to length zero and its file space released.

These six basic operations comprise the minimal set of required file operations. These primitive operations can then be combined to perform other file operations (i.e., copying). The OS keeps a small table, called the open-file table, containing information about all open files. When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being actively used, it is closed by the process, and the OS removes its entry from the open-file table. Most systems require that the programmer open a file explicitly with the $open () system call before that file can be used. The $open () operation takes a file name and searches the directory, copying the directory entry into the open-file table.

This call can also accept access-mode information (create, read-only, read-write, append-only, and so on). This mode is checked against the file's permissions. If the request mode is allowed, the file is opened for the process. The $open () system call typically returns a pointer to the entry in the open-file table. This pointer, not the actual file name, is used in all I/O operations. The implementation of the $open () and $close () operations is more complicated in an environment where several processes may open the file at the same time. This may occur in a system where several different applications open the same file at the same time.

Typically, the OS uses two levels of internal tables:

* A per-process table. The per-process table tracks all files that a process has open. For instance, the current file pointer for each file is found here. Access rights to the file and accounting information can also be included.
* A system-wide table. Each entry in the per-process table in turn points to a system-wide open-file table. The system-wide table contains process-independent information, such as the location of the file on disk, access dates, and file size. Once a file has been opened by one process, the system-wide table includes an entry for the file.

Typically, the open-file table also has an open count associated with each file to indicate how many processes have e the file open. Each $close () decreases this open count, and when the open count reaches zero, the file is no longer in use, and the file's entry is removed from the open-file table. In summary, several pieces of information are associated with an open file.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

This mini project in C Department Store Management System is a simple console-built application without graphic. In this project, you can manage a typical ‘fashion wear’ department store. You can add goods, edit goods, search, delete and display the goods. File handling has been used to record the information (rate, quantity, name and code) of the added goods. You can search the goods by rate, code or quantity. And, similar goes for display; you can display the items by quantity, rate or code.

The source code for this mini project is complete and totally error-free. It is compiled in Code::Blocks with gcc compiler. Unlike other C program source codes, I haven’t displayed the source code for this mini project on Department Store Management System in C here because it’s too long – over 800 lines. You can directly download the source code plus application file from the link below.

Functions used in Department Store Management System:

I have basically divided the functions used in this project into four parent functions. And, those individual functions have been described in short comments within the source code.

* Display functions:

void curser(int);

void dbill();

void d\_mainmenu();

void display(rec \*,int,int);

void window(int,int,int,int);

void dis\_con();

void d\_search();

void highlight(int,int);

* Main Menu functions:

void bill() ;

void edit();

void add();

void del();

void exit();

* Display sub-menu functions:

void d\_code();

void d\_rate();

void d\_quan();

void d\_all();

* Checking functions:

void c\_code(char[]);

int check(char[]);

void gotoxy (int x, int y) – I have been describing this function in every C mini project published on this site. You need to understand this function as it is one of the most important one used in this Department Store Management mini project.

This function allows you to print text in any place of screen. Using this function in Code::Blocks requires coding, but it can be directly used in Turbo C. Here is a code for this function in Code::Blocks.

COORD coord = {0, 0}; // sets coordinates to (0,0) as global variables

void gotoxy (int x, int y)

{

coord.X = x; coord.Y = y; // X and Y are the coordinates

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

}

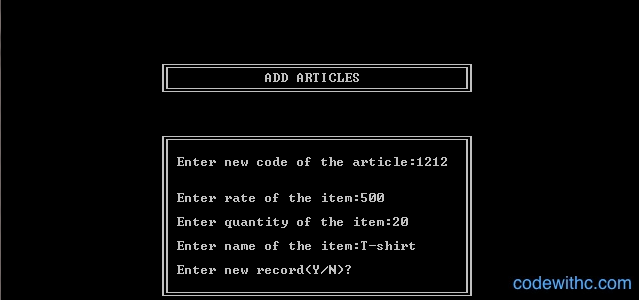
**CHAPTER 5**

**SAMPLE OUTPUT**

**5.1 Screenshots**



**Figure 5.1 Main Menu**



**Figure 5.2 Add Items**

**CHAPTER 6**

**CONCLUSION**

We have to work it to have online based DMS software for a Mini Shop. So that, their transaction process and payment report should be clear. In our country we find many shopping malls, they are used desktop base DSM software and fact many problems. I have done proper in this project are compliantly logical business model and its really sequential effective project. This project functionality, usability each and every point structural.

It is vital to narrow down the numerous selections available in order to find which DSM system will best suit a particular user. The system also provides for processing the return of purchased items and reimbursement to the customer. While many DSM Systems support multiple terminals that are networked together and

interface with external systems (such as inventory control) the primary goal of this system is to develop a self-contained sales terminal application that supports the purchase and return of store items.

**BIBLIOGRAPHY**

[1] https://www.codewithc.com/mini-project-in-c-medical-store-management-system/

[2] https://www.codewithc.com

[3] [www.youtube.com](http://www.youtube.com)

[4] [www.quora.com](http://www.quora.com)

[5] [www.wikipedia.org](http://www.wikipedia.org)

[6] <https://www.w3schools.com/cpp/cpp_files.asp>

[7] https://www.geeksforgeeks.org/introduction-to-c-programming-language/