DATABASE MANAGEMENT SOFTWARE  
(Store)



University Name Logo Above

Group Member Enrollment no

* Your Name Here

Unique Id Here



Submitted To:

**Teacher Name**

Contents

[DATABASE MANAGEMENT SOFTWARE (Store) 1](#_Toc502227871)

[STORE MANGEMENT SYSTEM 2](#_Toc502227877)

[Structures: 2](#_Toc502227878)

[Methods: 3](#_Toc502227879)

[Filling: 3](#_Toc502227880)

[Array: 3](#_Toc502227881)

[Do/While Loop: 4](#_Toc502227882)

[Switch Case: 4](#_Toc502227883)

[Functions: 5](#_Toc502227884)

[Code: 5](#_Toc502227885)

[Work Flow: 9](#_Toc502227886)

# STORE MANGEMENT SYSTEM

Our project (software) name is Store management system. In this software you have to login first then you can manage your store system, you can add employees and stock. After login it will ask you to choose the action you want to perform, like profile, add employees, add stock etc. This software is the way to control your store. Now we can tell you about its development and coding.

We have used functions, structures, methods, filings, arrays, do/while loops, and switch case for making our project.

## Structures:

When you create a struct object using the new operator, it gets created and the appropriate constructor is called according to the constructor's signature. Unlike classes, structs can be instantiated without using the”new” operator. In such a case, there is no constructor call, which makes the allocation more efficient. However, the fields will remain unassigned and the object cannot be used until all of the fields are initialized. This includes the inability to get or set values through auto-implemented properties.

We have used structures to add stock in our software. If you are successfully logged in, you are able to add stock in the application. You can easily add any product by entering its name, id, price etc.

## Methods:

A method is a code block that contains a series of statements. A program causes the statements to be executed by calling the method and specifying any required method arguments. In C#, every executed instruction is performed in the context of a method. The Main method is the entry point for every C# application and it is called by the common language runtime (CLR) when the program is started.

We have taken the support of methods for writing our information on the console box. All displaying material is a result of method we have used in our software.

## Filling:

To save the information permanently on the disk or reading information from the saved file through C# is known as **File Handling**. C# File Handling uses stream to save or retrieve information. A **file**is a collection of data stored on a disk with specific name, extension and directory path. When you open File using C# for reading and writing purpose it becomes stream. A stream is a sequence of bytes travelling from a source to a destination over a communication path.

We have used filling in our software for storing the information of employees and stock. If you add employee or any product in application it will be remained save until you delete this data by your own.

## Array:

Arrays works as collections of items, for instance strings. You can use them to gather items in a single group, and perform various operations on them, e.g. sorting. Besides that, several methods within the framework work on arrays, to make it possible to accept a range of items instead of just one. This fact alone makes it important to know a bit about arrays.

We have used array in our software for saving the employee name, or other information we are giving to our software.

## Do/While Loop:

In programming, it is often desired to execute certain block of statements for a specified number of times. A possible solution will be to type those statements for the required number of times. However, the number of repetition may not be known in advance (during compile time) or maybe large enough (say 10000). The best solution to such problem is loop. Loops are used in programming to repeatedly execute a certain block of statements until some condition is met.

We have used do/while loop in our whole coding for its smoothly run. Loop is the main part of our code. We use loop in each segment of the coding.

## Switch Case:

A switch case is used test variable equality for a list of values, where each value is a case. When the variable is equal to one of the cases, the statements following the case are executed. A break statement ends the switch case. The optional default case is for when the variable does not equal any of the cases. Variable used in a switch statement can only be a short, byte, int or char. The values for each case must be the same data type as the variable type.

We have used switch case in our software. When you logged into the software it will ask you for choosing the purpose for work. So switch case will very helpful in this part.

## Functions:

A function allows you to encapsulate a piece of code and call it from other parts of your code. You may very soon run into a situation where you need to repeat a piece of code, from multiple places, and this is where functions come in. The first part, public, is the visibility, and is optional. If you don't define any, then the function will be private. More about that later on. Next is the type to return. It could be any valid type in C#, or as we have done it here, void. A void means that this function returns absolutely nothing. Also, this function takes no parameters, as you can see from the empty set of parentheses, so it's actually just a tad bit boring.

We have used function for our software. It can return us the value of employees and stock. So our code will easily continue to the next step.

## Code:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace ConsoleApplication22

{

public struct AddStock

{

public int product\_id;

public string product\_name;

public string product\_companyname;

public float product\_price;

public void GetValues(int ID, string n, string n1, float p)

{

product\_id = ID;

product\_name = n;

product\_companyname = n1;

product\_price = p;

}

public void displaydetail()

{

Console.WriteLine("Product ID : {0}\nProduct Name : {1}\nCompany Name : {2} \nProduct Price : {3}", product\_id, product\_name, product\_companyname, product\_price);

}

}

class Project

{

private void Profile()

{

Console.Clear();

Console.WriteLine("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\* My Profile \*\*\*\*\*\*\*\*\*\*\n\n\n\n\t\tName : AAHM\t\t\t\t\t Contact : 03xx-xxxxxxx");

Console.WriteLine("\t\tEmail : xxxxxxxxx@email.com \t\t\t Department : Administration");

Console.WriteLine("\t\tCity : Karachi \t\t\t\t\t Country : Pakistan");

}

public void Stock()

{

Console.Clear();

Console.WriteLine("How many items you want to enter");

int it = int.Parse(Console.ReadLine());

Console.Clear();

for (int i = 0; i < it; i++)

{

AddStock st1 = new AddStock();

Console.WriteLine("Enter Product name ");

st1.product\_name = Convert.ToString(Console.ReadLine());

Console.WriteLine("Enter Product ID");

st1.product\_id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Company Name");

st1.product\_companyname = Convert.ToString(Console.ReadLine());

Console.WriteLine("Enter Product's Price");

st1.product\_price = float.Parse(Console.ReadLine());

Console.Clear();

Console.WriteLine("\t\t\t\*\*\*\*\*\*\*\*\*\* Stock Sucessfully Added \*\*\*\*\*\*\*\*\*\*");

st1.displaydetail();

Console.ReadKey();

Console.Clear();

}

}

public static void ReadFromFile(String Filename)

{

Console.Clear();

StreamReader srname = new StreamReader(Filename);

while (!srname.EndOfStream)

{

Console.WriteLine(srname.ReadLine());

}

srname.Close();

}

public static void WriteToFile(String Filename)

{

Console.Clear();

Console.WriteLine("How Many New Employees Name You Want to Add");

int e = int.Parse(Console.ReadLine());

Console.Clear();

string[] enames = new string[e];

for (int i = 0; i < enames.Length; i++)

{

Console.WriteLine("Enter Employee Name");

enames[i] = Convert.ToString(Console.ReadLine());

}

StreamWriter swname = File.AppendText(Filename);

for (int i = 0; i < enames.Length; i++)

{

swname.WriteLine(enames[i]);

}

swname.Close();

}

static void Main(string[] args)

{

int loginAttempts = 0;

for (int i = 0; i < 3; i++)

{

Console.WriteLine("Enter username");

string username = Console.ReadLine();

Console.WriteLine("Enter password");

string password = Console.ReadLine();

if (username != "valid" || password != "valid")

loginAttempts++;

else

break;

}

Console.Clear();

if (loginAttempts > 2)

{

Console.WriteLine("Login failure");

}

else

{

Console.WriteLine("Login successful");

Console.Clear();

char ans;

do

{

Console.WriteLine("\t\t\t\t\*\*\*\*\*\*\*\*\*\*You're Logged in as Admin\*\*\*\*\*\*\*\*\*\*\n\n\n\n (1) My Profile\n (2) Add Stock\n (3) Employees List \n (4) Add Employees");

Console.WriteLine("\n Enter The Respective Number To Perform Action");

int n = int.Parse(Console.ReadLine());

switch (n)

{

case 1:

{

Project f1 = new Project();

f1.Profile();

break;

}

case 2:

{

Project f2 = new Project();

f2.Stock();

break;

}

case 3:

{

String Filename = @"D:\names.txt";

ReadFromFile(Filename);

break;

}

case 4:

{

String Filename = @"D:\names.txt";

WriteToFile(Filename);

break;

}

}

Console.ReadKey();

Console.Clear();

Console.WriteLine("Do you want to Continue [Y/N]");

ans = Convert.ToChar(Console.ReadLine());

Console.Clear();

}

while (ans == 'y');

}

}

}

}

## Work Flow:

