|  |
| --- |
| ID: 04200101466 |
| NAME: Md. Mehedi Hasan |
| COURSE CODE: CSE 1290 |
| COURSE TITLE: Software Development I (C Programming) |

Submitted to

Sanjida Akter  
Senior lecturer, CSE, NUB

**SOURCE CODE AND OUTPUT**

(POS System)

**File Name:** main.cpp

**Code:**

**#include<iostream>**

**#include <string>**

**#include "tabulate/table.hpp"**

**#include "header/menu/mainMenu.h"**

**using namespace tabulate;**

**using namespace std;**

**void successMessage();**

**void errorMessage();**

**int main() {**

**int value;**

**while(1){**

**value = mainMenu();**

**if(value == 10000000){**

**break;**

**}**

**}**

**return 0;**

**}**

**File Name:** mainMenu.h

**Code:-**

#include<iostream>

#include<stdlib.h>

#include "../../tabulate/table.hpp"

#include "productMenu.h"

#include "customerMenu.h"

#include "supplierMenu.h"

#include "../modules/product/viewSale.h"

using namespace tabulate;

using namespace std;

int mainMenu(){

int option;

Table universal\_constants;

universal\_constants.add\_row({"","Please select an option:"});

universal\_constants.add\_row({"","Product 1"});

universal\_constants.add\_row({"","Customer 2"});

universal\_constants.add\_row({"","Supplier 3"});

universal\_constants.add\_row({"","Stock 4"});

universal\_constants.add\_row({"","Invoices 5"});

universal\_constants.add\_row({"","Profit 6"});

universal\_constants.add\_row({"","EXIT 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left(" ")

.border\_right(" ")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::magenta);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green);

universal\_constants[2][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[2][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::underline})

.font\_style({FontStyle::bold})

.font\_background\_color(Color::cyan);

universal\_constants[3][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[3][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::blue);

universal\_constants[4][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[4][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::yellow);

universal\_constants[5][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[5][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

universal\_constants[6][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[6][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::cyan);

universal\_constants[7][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[7][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

std::cout << universal\_constants << std::endl;

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

cin>>option;

system ("CLS");

switch(option){

int value;

case 1:

while(1){

value = porductMenu();

if(value == 10000000){

break;

}

}

break;

case 2:

while(1){

value = customerMenu();

if(value == 10000000){

break;

}

}

break;

case 3:

while(1){

value = supplierMenu();

if(value == 10000000){

break;

}

}

break;

case 4:

cout<<"4 selected";

break;

case 5:

viewSale();

break;

case 6:

cout<<"6 selected";

break;

case 0:

return 10000000;

break;

default:

errorMessage("Invalid Option!");

}

return option;

}

**OUTPUT:**



**File Name:** productMenu.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include "../../tabulate/table.hpp"

#include "../modules/product/addProduct.h"

#include "../message/errorMessage.h"

#include "../modules/product/viewProduct.h"

#include "../modules/product/buyProduct.h"

#include "../modules/product/saleProduct.h"

#include "../modules/product/editProduct.h"

#include "../modules/product/deleteProduct.h"

using namespace tabulate;

using namespace std;

int porductMenu(){

int option;

Table universal\_constants;

universal\_constants.add\_row({"","Please select an option:"});

universal\_constants.add\_row({"","Add Product 1"});

universal\_constants.add\_row({"","Edit Product 2"});

universal\_constants.add\_row({"","Delete Product 3"});

universal\_constants.add\_row({"","Buy Product 4"});

universal\_constants.add\_row({"","Sale Product 5"});

universal\_constants.add\_row({"","View Product 6"});

universal\_constants.add\_row({"","BACK 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left(" ")

.border\_right(" ")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::magenta);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green);

universal\_constants[2][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[2][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::underline})

.font\_style({FontStyle::bold})

.font\_background\_color(Color::cyan);

universal\_constants[3][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[3][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::blue);

universal\_constants[4][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[4][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::yellow);

universal\_constants[5][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[5][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

universal\_constants[6][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[6][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::cyan);

universal\_constants[7][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[7][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

std::cout << universal\_constants << std::endl;

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

cin>>option;

system ("CLS");

switch(option){

case 1:

addProduct();

break;

case 2:

editProduct();

break;

case 3:

deleteProduct();

break;

case 4:

buyProduct();

break;

case 5:

saleProduct();

break;

case 6:

viewProduct();

break;

case 0:

return 10000000;

break;

default:

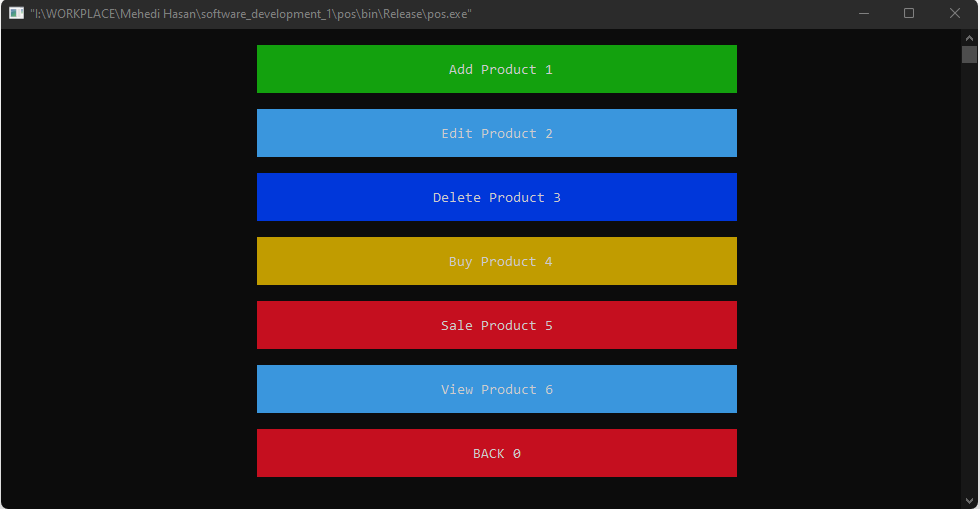
errorMessage("Invalid Option!");

}

return option;

}

**OUTPUT:**



**File Name:** productClass.h

**Code:-**

#include<iostream>

#include <string>

#include<stdlib.h>

using namespace std;

string name;

struct Product{

int id;

int purchasePrice;

int salePrice;

int inStock;

string date;

string name;

struct Product \*next;

};

typedef struct Product products;

products \*start = NULL;

**File Name:** addProduct.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include <string>

#include <ctime>

#include "../../store.h"

#include "productClass.h"

#include "../../message/successMessage.h"

#include "../../../tabulate/table.hpp"

int currentProductId = 1;

using namespace std;

using namespace tabulate;

products \*getNode(){

products\* newNode;

newNode = (products\*) malloc(sizeof(products));

string name;

cout<<"Enter product name:";

getline(cin, name);

getline(cin, name);

newNode->name = name;

system ("CLS");

cout<<"Enter product purchase price:";

int purchasePrice;

cin>>purchasePrice;

newNode->purchasePrice = purchasePrice;

system ("CLS");

cout<<"Enter product sale price:";

int salePrice;

cin>>salePrice;

newNode->salePrice = salePrice;

system ("CLS");

time\_t now = time(0);

char\* dt = ctime(&now);

newNode->date = dt;

newNode->id = currentProductId;

newNode->inStock = 0;

newNode->next = NULL;

return newNode;

}

void addProduct(){

products \*newNode;

products \*temp;

newNode = getNode();

currentProductId++;

Table movies;

movies.add\_row({"Id", "Name", "Purchase Price", "Sale Price","In Stock", "Date"});

movies.add\_row({to\_string(newNode->id), newNode->name, to\_string(newNode->purchasePrice), to\_string(newNode->salePrice),to\_string(newNode->inStock), newNode->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

if(start == NULL){

start = newNode;

std::cout << movies << std::endl;

successMessage("Product added successfully.");

}else{

temp = start;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

std::cout << movies << std::endl;

successMessage("Product added successfully.");

}

}

**File Name:** successMessage.h

**Code:-**

#include<iostream>

#include <string>

#include "../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void successMessage(string message){

int option;

Table universal\_constants;

universal\_constants.add\_row({"",message,""});

universal\_constants.add\_row({"","Back 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left("")

.border\_right("")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green)

.width(20);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::green);

universal\_constants[0][2].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green)

.width(30);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

std::cout << universal\_constants << std::endl;

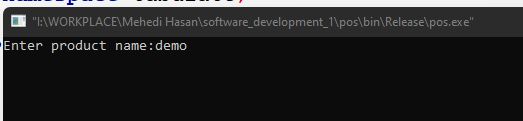
cout<<"\t\t\t\t\t\t\t";

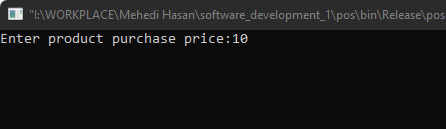
cin>>option;

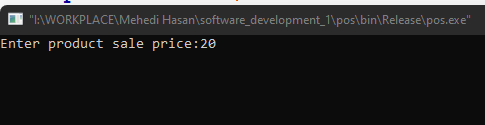
system ("CLS");

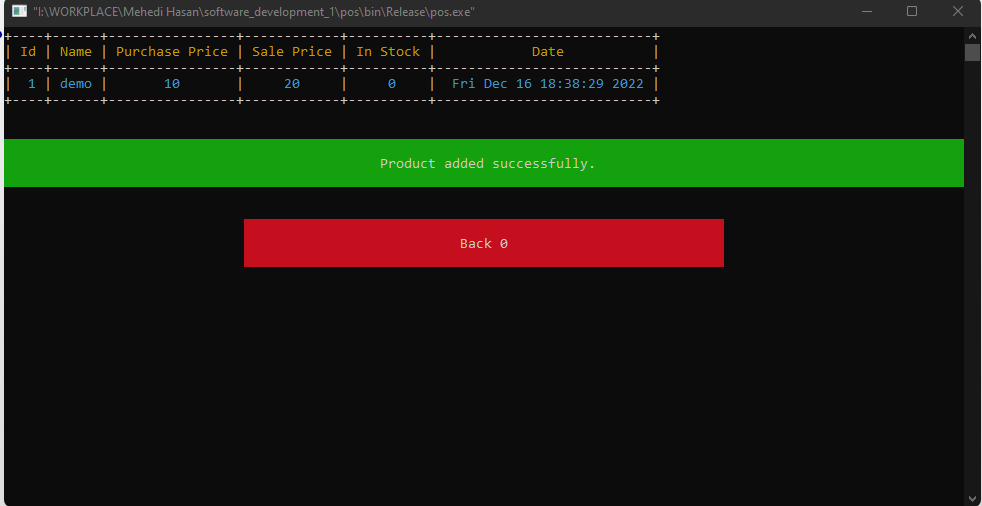
}

**OUTPUT:**









**File Name:** editProduct.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void editProduct(){

int productId;

viewProduct();

cout<<"Enter a product id : ";

cin>>productId;

system ("CLS");

products\* EP = findProduct(productId);

string name;

cout<<"Current product name : ->[ " <<EP->name <<" ]" <<endl;

cout<<"Enter product name:";

getline(cin, name);

getline(cin, name);

EP->name = name;

system ("CLS");

cout<<"Current purchase price : ->[ " <<EP->purchasePrice <<" ]" <<endl;

cout<<"Enter product purchase price:";

int purchasePrice;

cin>>purchasePrice;

EP->purchasePrice = purchasePrice;

system ("CLS");

cout<<"Current sale price : ->[ " <<EP->salePrice <<" ]" <<endl;

cout<<"Enter product sale price:";

int salePrice;

cin>>salePrice;

EP->salePrice = salePrice;

system ("CLS");

Table movies;

movies.add\_row({"Id", "Name", "Purchase Price", "Sale Price","In Stock", "Date"});

movies.add\_row({to\_string(EP->id), EP->name, to\_string(EP->purchasePrice), to\_string(EP->salePrice),to\_string(EP->inStock), EP->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

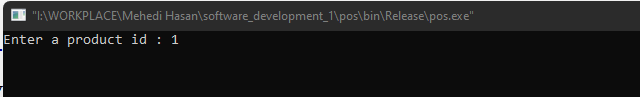
}

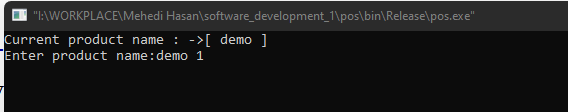
std::cout << movies << std::endl;

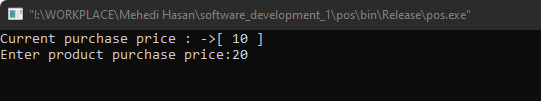
successMessage("Product update successfully.");

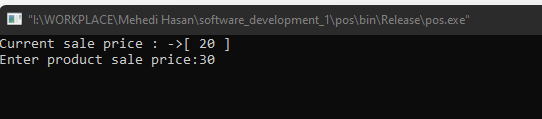
}

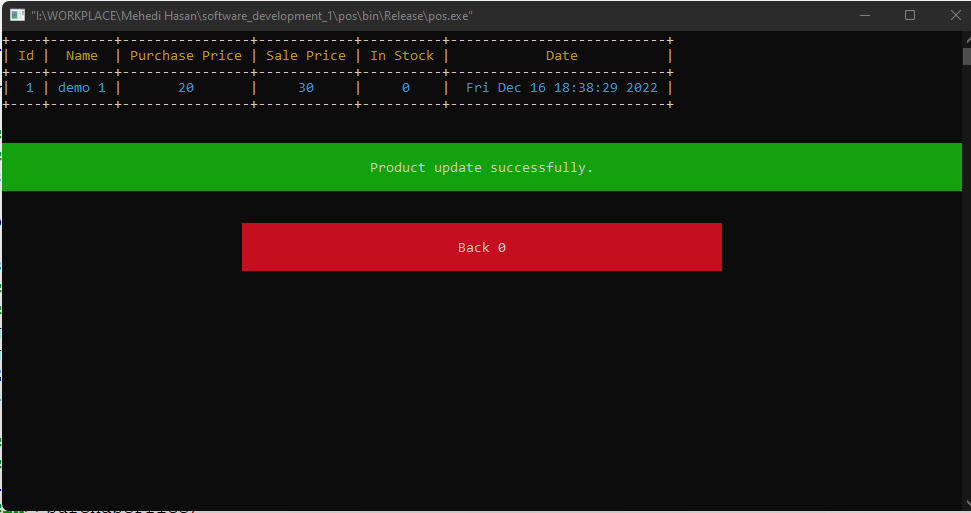
**OUTPUT:**











**File Name:** deleteProduct.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void deleteProduct(){

viewProduct();

int id;

cout<< "Enter product id:";

cin>>id;

system ("CLS");

products \*dataFPDL = start;

products \*dataFPDL2;

products \*dataFPDL3;

if(start->id == id && start->next == NULL){

free(start);

start = NULL;

successMessage("Product delete successfully.");

}else{

if(start->id == id ){

free(start);

start = dataFPDL->next;

successMessage("Product delete successfully.");

}else{

while(dataFPDL != NULL){

if(dataFPDL->next->id == id){

if(dataFPDL->next->next == NULL){

dataFPDL2 = dataFPDL->next->next;

free(dataFPDL2);

dataFPDL->next = NULL;

successMessage("Product delete successfully.");

break;

}else{

dataFPDL2 = dataFPDL->next->next;

dataFPDL3 = dataFPDL->next;

free(dataFPDL3);

dataFPDL->next = dataFPDL2;

successMessage("Product0 delete successfully.");

break;

}

}

dataFPDL = dataFPDL->next;

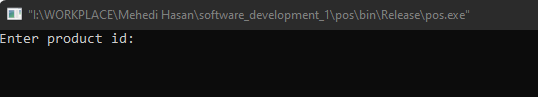
}

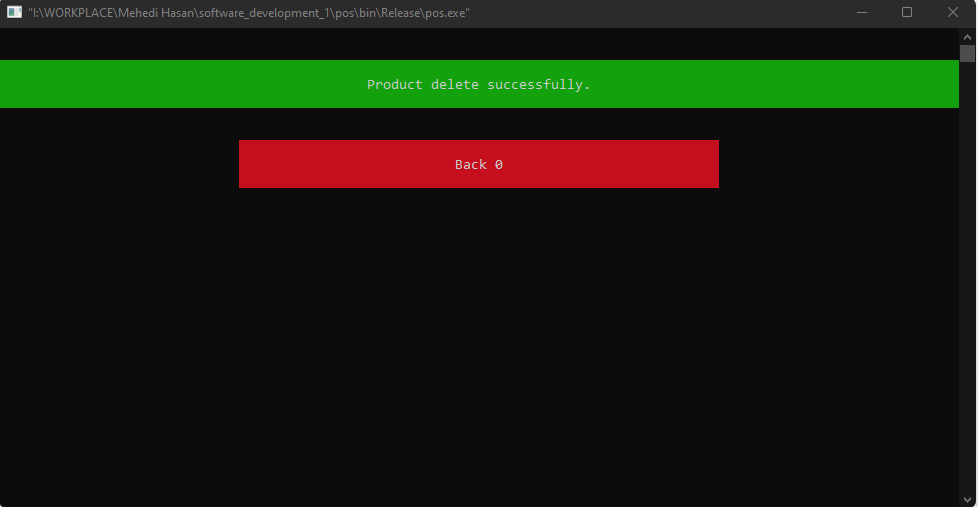
}

}

}

**OUTPUT:**





**File Name:** supplierMenu.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include "../../tabulate/table.hpp"

#include "../modules/supplier/addSupplier.h"

#include "../modules/supplier/EditSupplier.h"

#include "../modules/supplier/deleteSupplier.h"

using namespace tabulate;

using namespace std;

int supplierMenu(){

int option;

Table universal\_constants;

universal\_constants.add\_row({"","Please select an option:"});

universal\_constants.add\_row({"","Add Supplier 1"});

universal\_constants.add\_row({"","Edit Supplier 2"});

universal\_constants.add\_row({"","Delete Supplier 3"});

universal\_constants.add\_row({"","View Supplier 4"});

universal\_constants.add\_row({"","Supplier Invoices 5"});

universal\_constants.add\_row({"","BACK 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left(" ")

.border\_right(" ")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::magenta);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green);

universal\_constants[2][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[2][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::underline})

.font\_style({FontStyle::bold})

.font\_background\_color(Color::cyan);

universal\_constants[3][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[3][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::blue);

universal\_constants[4][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[4][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::yellow);

universal\_constants[5][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[5][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

universal\_constants[6][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[6][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::cyan);

std::cout << universal\_constants << std::endl;

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

cin>>option;

system ("CLS");

switch(option){

case 1:

addSupplier();

break;

case 2:

editSupplier();

break;

case 3:

deleteSupplier();

break;

case 4:

viewSupplier();

break;

case 5:

cout<<"5 selected";

break;

case 0:

return 10000000;

break;

default:

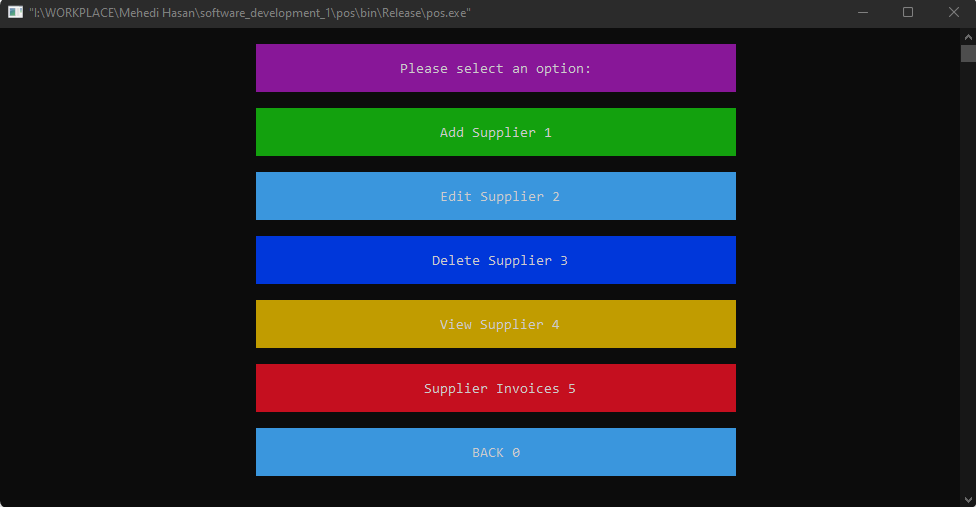
cout<<"selected";

}

return option;

}

**OUTPUT:**

****

**File Name:** supplierClass.h

**Code:-**

#include<iostream>

#include <string>

#include<stdlib.h>

using namespace std;

struct Supplier{

int id;

string name;

string email;

string phone;

string address;

string date;

struct Supplier \*next;

};

typedef struct Supplier suppliers;

suppliers \*startS = NULL;

**File Name:** AddSupplier.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include <string>

#include <ctime>

#include "../../store.h"

#include "../../../tabulate/table.hpp"

int currentSupplierId = 1;

using namespace std;

using namespace tabulate;

suppliers \*getSNode(){

suppliers\* newNode;

newNode = (suppliers\*) malloc(sizeof(suppliers));

string nam;

cout<<"Enter supplier name:";

getline(cin, nam);

getline(cin, nam);

newNode->name = nam;

system ("CLS");

cout<<"Enter supplier email:";

string email;

getline(cin, email);

newNode->email = email;

system ("CLS");

cout<<"Enter supplier phone:";

string phone;

getline(cin, phone);

newNode->phone = phone;

system ("CLS");

cout<<"Enter supplier address:";

string address;

getline(cin, address);

newNode->address = address;

system ("CLS");

time\_t now = time(0);

char\* dt = ctime(&now);

newNode->date = dt;

newNode->id = currentSupplierId;

newNode->next = NULL;

return newNode;

}

void addSupplier(){

suppliers \*newNode;

suppliers \*temp;

newNode = getSNode();

currentSupplierId++;

Table movies;

movies.add\_row({"Id", "Name", "Email", "phone","Address", "Date"});

movies.add\_row({to\_string(newNode->id), newNode->name, newNode->email, newNode->phone, newNode->address, newNode->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

if(startS == NULL){

startS = newNode;

std::cout << movies << std::endl;

successMessage("Supplier added successfully.");

}else{

temp = startS;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

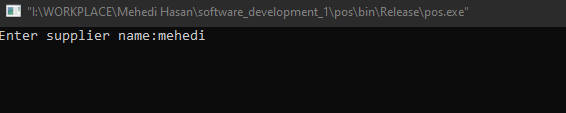
std::cout << movies << std::endl;

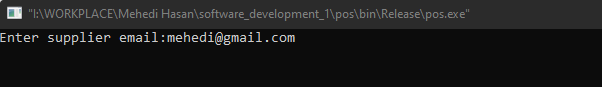
successMessage("Supplier added successfully.");

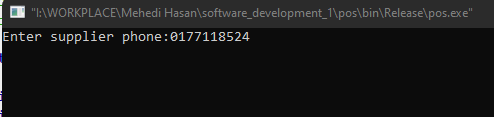
}

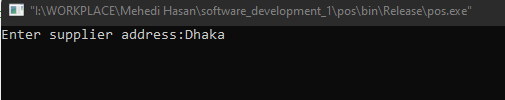
}

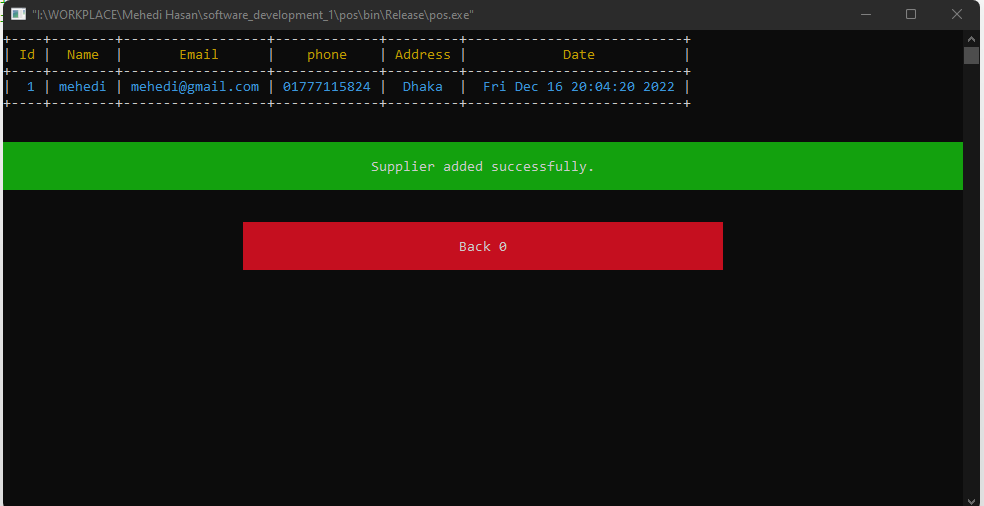
**OUTPUT:**

****









**File Name:** editSupplier.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void editSupplier(){

int supplierId;

viewSupplier();

cout<<"Enter a supplier id : ";

cin>>supplierId;

system ("CLS");

suppliers\* ES = findSupplier(supplierId);

cout<<"Current supplier name : ->[ " <<ES->name <<" ]" <<endl;

string nam;

cout<<"Enter supplier name:";

getline(cin, nam);

getline(cin, nam);

ES->name = nam;

system ("CLS");

cout<<"Current supplier email : ->[ " <<ES->email <<" ]" <<endl;

cout<<"Enter supplier email:";

string email;

getline(cin, email);

ES->email = email;

system ("CLS");

cout<<"Current supplier phone : ->[ " <<ES->phone <<" ]" <<endl;

cout<<"Enter supplier phone:";

string phone;

getline(cin, phone);

ES->phone = phone;

system ("CLS");

cout<<"Current supplier address : ->[ " <<ES->address <<" ]" <<endl;

cout<<"Enter supplier address:";

string address;

getline(cin, address);

ES->address = address;

system ("CLS");

Table movies;

movies.add\_row({"Id", "Name", "Email", "phone", "Address", "Date"});

movies.add\_row({to\_string(ES->id), ES->name, ES->email, ES->phone, ES->address, ES->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

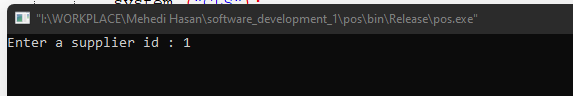
}

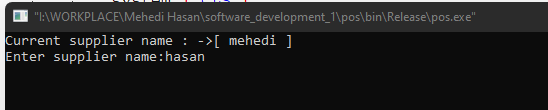
std::cout << movies << std::endl;

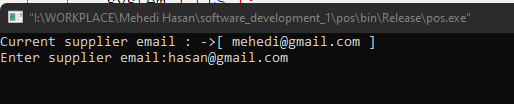
successMessage("Supplier update successfully.");

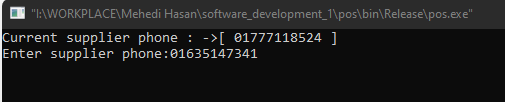
}

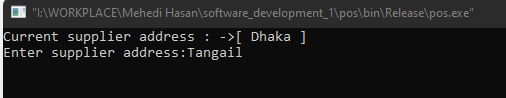
**OUTPUT:**

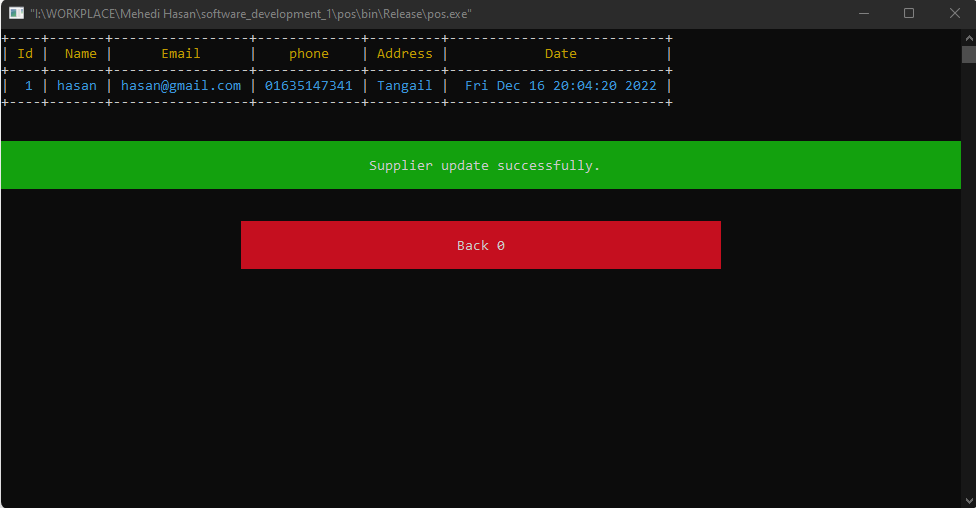












**File Name:** viewSupplier.h

**Code:-**

#include <iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void viewSupplier(){

suppliers \*dataS = startS;

Table movies;

movies.add\_row({"Id", "Name", "Email", "Phone","Address", "Date"});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

int n = 0;

string total ;

while(dataS != NULL){

n++;

movies.add\_row({to\_string(dataS->id), dataS->name, dataS->email, dataS->phone, dataS->address, dataS->date});

dataS = dataS->next;

}

std::cout << movies << std::endl;

total = "Total suppliers: " + to\_string(n);

successMessage(total);

}

**File Name:** findSupplier.h

**Code:-**

#include<iostream>

#include "supplierClass.h"

using namespace std;

suppliers \*findSupplier(int id){

suppliers \*dataFS = startS;

while(dataFS != NULL){

if(dataFS->id == id){

return(dataFS);

}

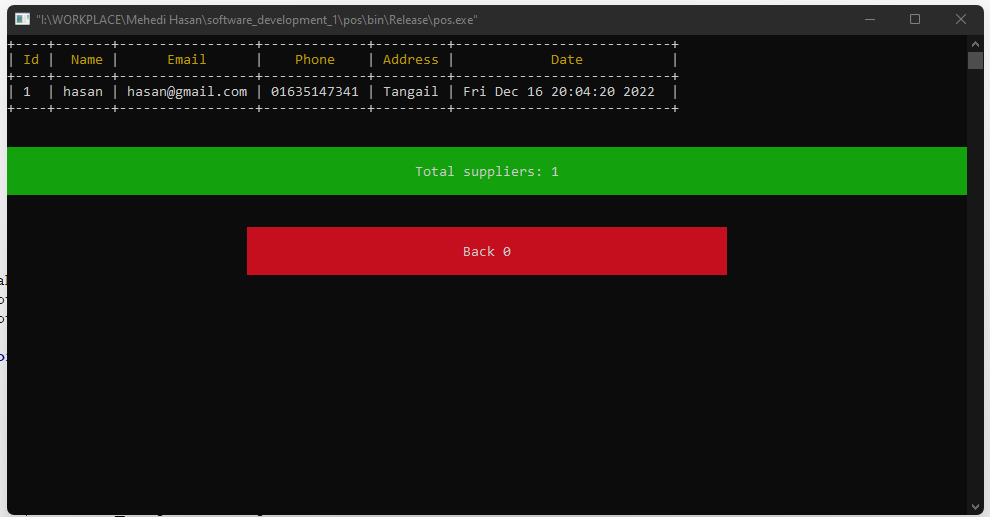
dataFS = dataFS->next;

}

return(NULL);

}

**OUTPUT:**



**File Name:** deleteSupplier.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void deleteSupplier(){

viewSupplier();

int id;

cout<< "Enter supplier id:";

cin>>id;

system ("CLS");

suppliers \*dataFSDL = startS;

suppliers \*dataFSDL2;

suppliers \*dataFSDL3;

if(startS->id == id && startS->next == NULL){

free(startS);

startS = NULL;

successMessage("Supplier delete successfully.");

}else{

if(startS->id == id ){

free(startS);

startS = dataFSDL->next;

successMessage("Supplier delete successfully.");

}else{

while(dataFSDL != NULL){

if(dataFSDL->next->id == id){

if(dataFSDL->next->next == NULL){

dataFSDL2 = dataFSDL->next->next;

free(dataFSDL2);

dataFSDL->next = NULL;

successMessage("Supplier delete successfully.");

break;

}else{

dataFSDL2 = dataFSDL->next->next;

dataFSDL3 = dataFSDL->next;

free(dataFSDL3);

dataFSDL->next = dataFSDL2;

successMessage("Supplier delete successfully.");

break;

}

}

dataFSDL = dataFSDL->next;

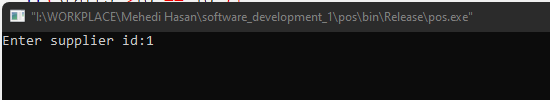
}

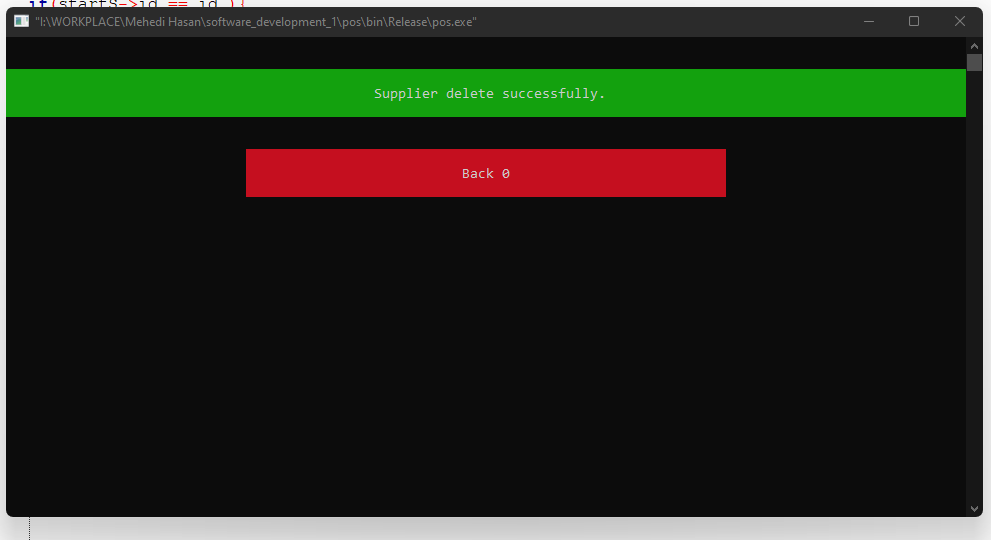
}

}

}

**OUTPUT:**





**File Name:** customerMenu.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include "../../tabulate/table.hpp"

#include "../modules/customer/addCustomer.h"

#include "../modules/customer/editCustomer.h"

#include "../modules/customer/deleteCustomer.h"

using namespace tabulate;

using namespace std;

int customerMenu(){

int option;

Table universal\_constants;

universal\_constants.add\_row({"","Please select an option:"});

universal\_constants.add\_row({"","Add Customer 1"});

universal\_constants.add\_row({"","Edit Customer 2"});

universal\_constants.add\_row({"","Delete Customer 3"});

universal\_constants.add\_row({"","View Customer 4"});

universal\_constants.add\_row({"","Customer Invoices 5"});

universal\_constants.add\_row({"","BACK 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left(" ")

.border\_right(" ")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::magenta);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::green);

universal\_constants[2][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[2][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::underline})

.font\_style({FontStyle::bold})

.font\_background\_color(Color::cyan);

universal\_constants[3][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[3][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::blue);

universal\_constants[4][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[4][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::yellow);

universal\_constants[5][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[5][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

universal\_constants[6][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[6][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::cyan);

std::cout << universal\_constants << std::endl;

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

cin>>option;

system ("CLS");

switch(option){

case 1:

addCustomer();

break;

case 2:

editCustomer();

break;

case 3:

deleteCustomer();

break;

case 4:

viewCustomer();

break;

case 5:

cout<<"5 selected";

break;

case 0:

return 10000000;

break;

default:

cout<<"selected";

}

return option;

}

**OUTPUT:**



**File Name:** customerClash.h

**Code:-**

#include<iostream>

#include <string>

#include<stdlib.h>

using namespace std;

struct Customer{

int id;

string email;

string phone;

string address;

string date;

string name;

struct Customer \*next;

};

typedef struct Customer customers;

customers \*startC = NULL;

**File Name:** addCustomer.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include <string>

#include <ctime>

#include "../../store.h"

//#include "customerClass.h"

#include "../../../tabulate/table.hpp"

int currentCustomerId = 1;

using namespace std;

using namespace tabulate;

customers \*getCNode(){

customers\* newNode;

newNode = (customers\*) malloc(sizeof(customers));

string nam;

cout<<"Enter customer name:";

getline(cin, nam);

getline(cin, nam);

newNode->name = nam;

system ("CLS");

cout<<"Enter customer email:";

string email;

getline(cin, email);

newNode->email = email;

system ("CLS");

cout<<"Enter customer phone:";

string phone;

getline(cin, phone);

newNode->phone = phone;

system ("CLS");

cout<<"Enter customer address:";

string address;

getline(cin, address);

newNode->address = address;

system ("CLS");

time\_t now = time(0);

char\* dt = ctime(&now);

newNode->date = dt;

newNode->id = currentCustomerId;

newNode->next = NULL;

return newNode;

}

void addCustomer(){

customers \*newNode;

customers \*temp;

newNode = getCNode();

currentCustomerId++;

Table movies;

movies.add\_row({"Id", "Name", "Email", "phone","Address", "Date"});

movies.add\_row({to\_string(newNode->id), newNode->name, newNode->email, newNode->phone, newNode->address, newNode->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

if(startC == NULL){

startC = newNode;

std::cout << movies << std::endl;

successMessage("Customer added successfully.");

}else{

temp = startC;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

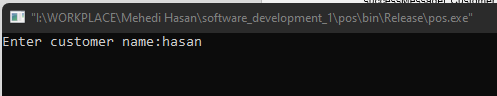
std::cout << movies << std::endl;

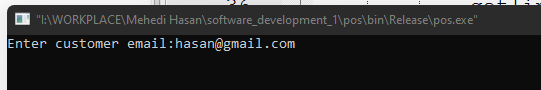
successMessage("Customer added successfully.");

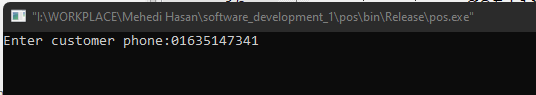
}

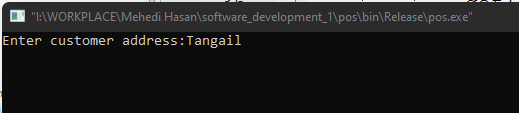
}

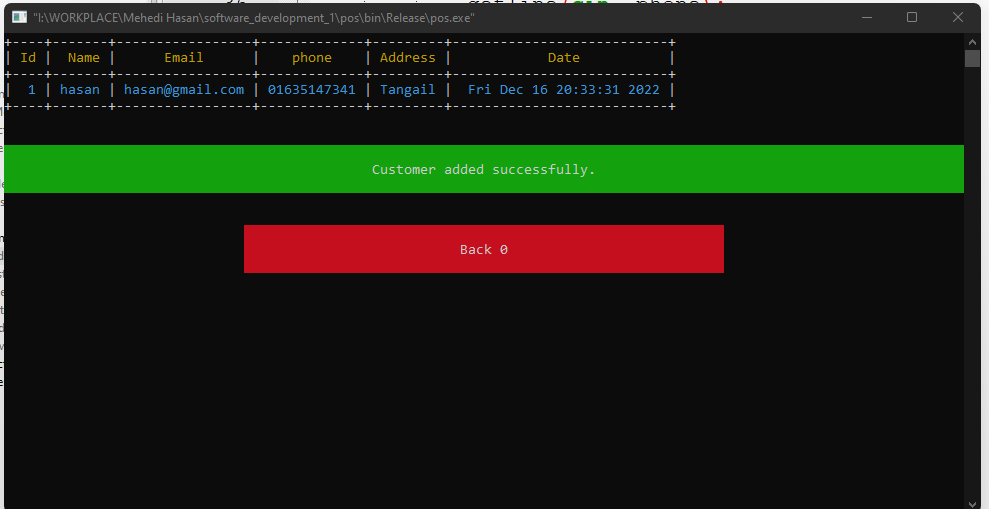
**OUTPUT:**











**File Name:** editCustomer.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void editCustomer(){

int customerId;

viewCustomer();

cout<<"Enter a supplier id : ";

cin>>customerId;

system ("CLS");

customers\* EC = findCustomer(customerId);

cout<<"Current supplier name : ->[ " <<EC->name <<" ]" <<endl;

string nam;

cout<<"Enter supplier name:";

getline(cin, nam);

getline(cin, nam);

EC->name = nam;

system ("CLS");

cout<<"Current supplier email : ->[ " <<EC->email <<" ]" <<endl;

cout<<"Enter supplier email: ";

string email;

getline(cin, email);

EC->email = email;

system ("CLS");

cout<<"Current supplier phone : ->[ " <<EC->phone <<" ]" <<endl;

cout<<"Enter supplier phone: ";

string phone;

getline(cin, phone);

EC->phone = phone;

system ("CLS");

cout<<"Current supplier address : ->[ " <<EC->address <<" ]" <<endl;

cout<<"Enter supplier address: ";

string address;

getline(cin, address);

EC->address = address;

system ("CLS");

Table movies;

movies.add\_row({"Id", "Name", "Email", "phone", "Address", "Date"});

movies.add\_row({to\_string(EC->id), EC->name, EC->email, EC->phone, EC->address, EC->date});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

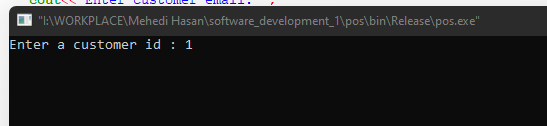
}

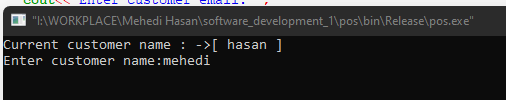
std::cout << movies << std::endl;

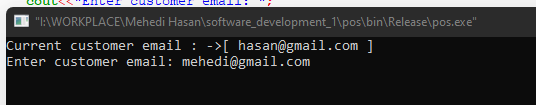
successMessage("Customer update successfully.");

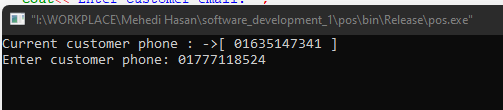
}

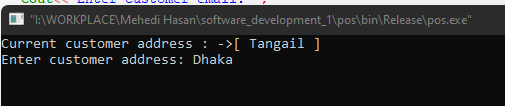
**OUTPUT:**

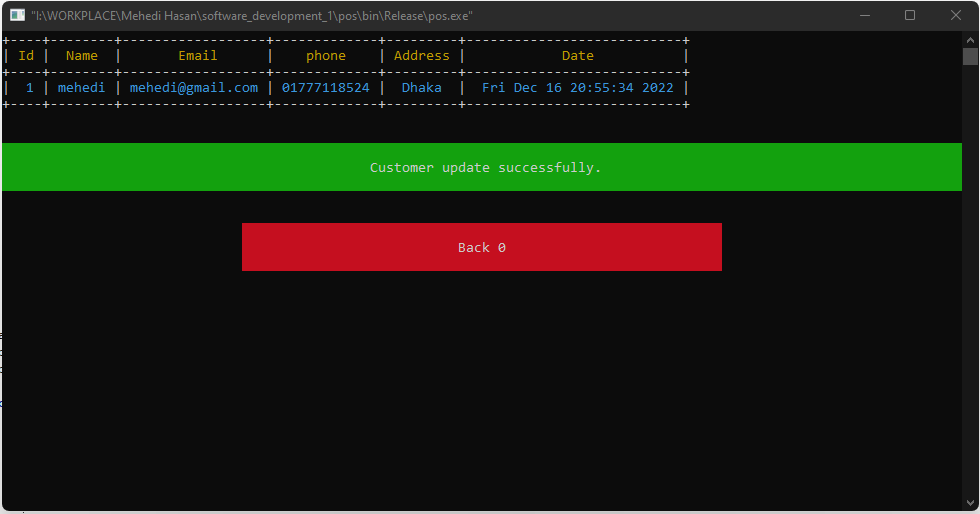












**File Name:** viewCustomer.h

**Code:-**

#include <iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void viewCustomer(){

customers \*dataC = startC;

Table movies;

movies.add\_row({"Id", "Name", "Email", "Phone","Address", "Date"});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

int n = 0;

string total ;

while(dataC != NULL){

n++;

movies.add\_row({to\_string(dataC->id), dataC->name, dataC->email, dataC->phone, dataC->address, dataC->date});

dataC = dataC->next;

}

std::cout << movies << std::endl;

total = "Total customers: " + to\_string(n);

successMessage(total);

}

**File Name:** findCustomer.h

**Code:-**

#include<iostream>

#include "customerClass.h"

using namespace std;

customers \*findCustomer(int id){

customers \*dataFC = startC;

while(dataFC != NULL){

if(dataFC->id == id){

return(dataFC);

}

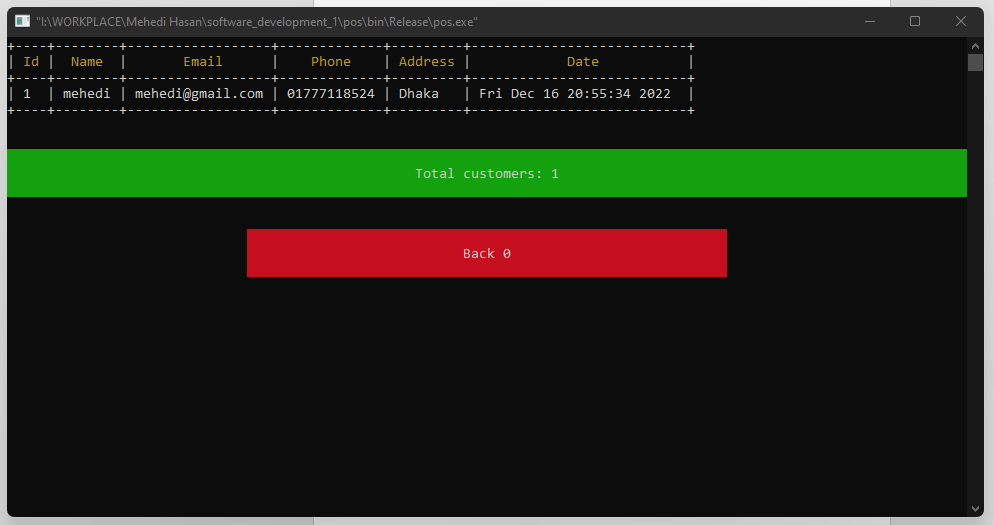
dataFC = dataFC->next;

}

return(NULL);

}

**OUTPUT:**



**File Name:** deleteCustomer.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void deleteCustomer(){

viewCustomer();

int id;

cout<< "Enter customer id:";

cin>>id;

system ("CLS");

customers \*dataFCDL = startC;

customers \*dataFCDL2;

customers \*dataFCDL3;

if(startC->id == id && startC->next == NULL){

free(startC);

startC = NULL;

successMessage("Customer delete successfully.");

}else{

if(startC->id == id ){

free(startC);

startC = dataFCDL->next;

successMessage("Customer delete successfully.");

}else{

while(dataFCDL != NULL){

if(dataFCDL->next->id == id){

if(dataFCDL->next->next == NULL){

dataFCDL2 = dataFCDL->next->next;

free(dataFCDL2);

dataFCDL->next = NULL;

successMessage("Customer delete successfully.");

break;

}else{

dataFCDL2 = dataFCDL->next->next;

dataFCDL3 = dataFCDL->next;

free(dataFCDL3);

dataFCDL->next = dataFCDL2;

successMessage("Customer delete successfully.");

break;

}

}

dataFCDL = dataFCDL->next;

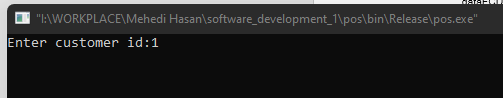
}

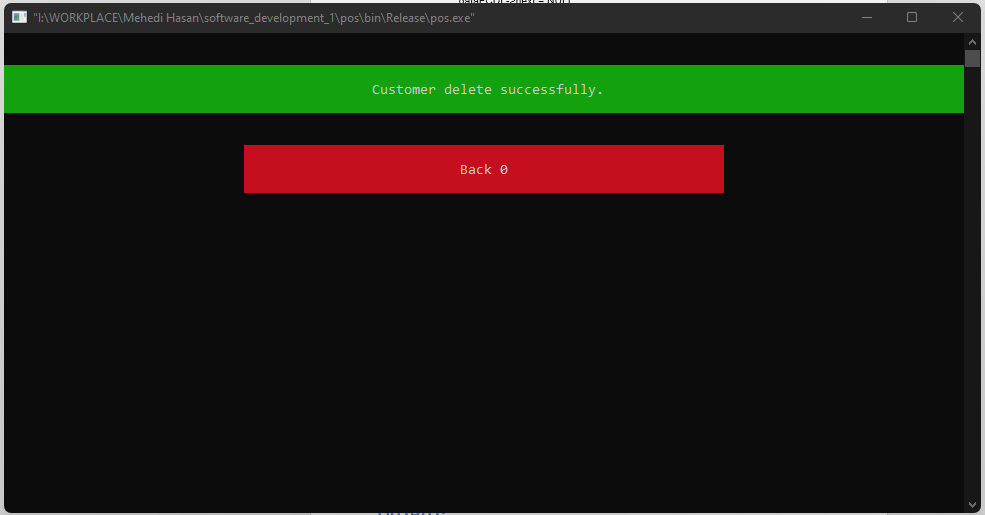
}

}

}

**OUTPUT:**





**File Name:** buyProduct.h

**Code:-**

#include<iostream>

#include <stdlib.h>

#include "findProduct.h"

#include "../supplier/findSupplier.h"

#include "../supplier/viewSupplier.h"

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

int currentBuyProductId = 1;

struct buyProductList{

int id;

int productId;

int supplierId;

int purchasePrice;

int totalPrice;

int quantity;

string date;

string productName;

string supplierName;

struct buyProductList \*next;

};

typedef struct buyProductList buyProducts;

buyProducts \*startB = NULL;

buyProducts \*getBNode(){

buyProducts\* newNode;

newNode = (buyProducts\*) malloc(sizeof(buyProducts));

int productId;

viewProduct();

cout<<"Enter product id: ";

cin>>productId;

products\* p = findProduct(productId);

newNode->productId = p->id;

newNode->productName = p->name;

newNode->purchasePrice = p->purchasePrice;

system ("CLS");

int supplierId;

viewSupplier();

cout<<"Enter supplier id: ";

cin>>supplierId;

suppliers\* s = findSupplier(supplierId);

newNode->supplierId = s->id;

newNode->supplierName = s->name;

system ("CLS");

cout<<"Enter product quantity: ";

int quantity;

cin>>quantity;

newNode->quantity = quantity;

p->inStock = p->inStock + quantity;

system ("CLS");

time\_t now = time(0);

char\* dt = ctime(&now);

newNode->date = dt;

newNode->id = currentBuyProductId;

newNode->totalPrice = quantity \* p->purchasePrice;

newNode->next = NULL;

return newNode;

}

void buyProduct(){

buyProducts \*newNode;

buyProducts \*temp;

newNode = getBNode();

currentBuyProductId++;

Table movies;

movies.add\_row({"Id", "PId", "PName", "SId", "SName", "Purchase Price", "Quantity", "Total Price", "Date"});

movies.add\_row({to\_string(newNode->id), to\_string(newNode->productId), newNode->productName, to\_string(newNode->supplierId), newNode->supplierName, to\_string(newNode->purchasePrice), to\_string(newNode->quantity), to\_string(newNode->totalPrice), newNode->date});

for (size\_t i = 0; i < 7; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

if(startB == NULL){

startB = newNode;

std::cout << movies << std::endl;

successMessage("Product buy successfully.");

}else{

temp = startB;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

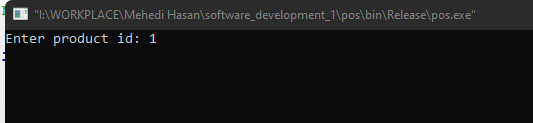
std::cout << movies << std::endl;

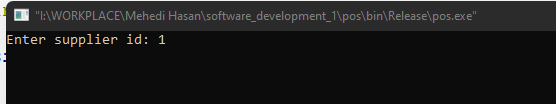
successMessage("Product buy successfully.");

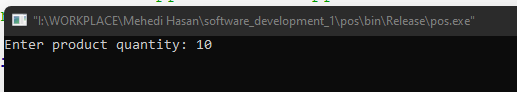
}

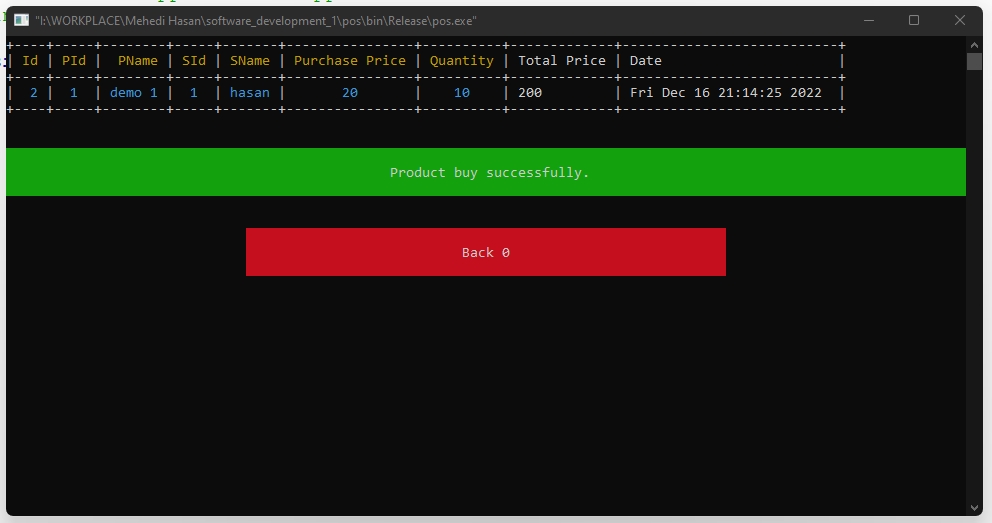
}

**OUTPUT:**









**File Name:** saleProduct.h

**Code:-**

#include<iostream>

#include <stdlib.h>

//#include "findProduct.h"

#include "../customer/findCustomer.h"

#include "../customer/viewCustomer.h"

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

int currentSaleProductId = 1;

struct saleProductList{

int id;

int productId;

int customerId;

int salePrice;

int totalPrice;

int quantity;

string date;

string customerName;

string productName;

struct saleProductList \*next;

};

typedef struct saleProductList saleProducts;

saleProducts \*startPS = NULL;

saleProducts \*getPSNode(){

saleProducts\* newNode;

newNode = (saleProducts\*) malloc(sizeof(saleProducts));

int productId;

viewProduct();

cout<<"Enter product id: ";

cin>>productId;

products\* p2 = findProduct(productId);

newNode->productId = p2->id;

newNode->productName = p2->name;

newNode->salePrice = p2->salePrice;

system ("CLS");

int customerId;

viewCustomer();

cout<<"Enter customer id: ";

cin>>customerId;

customers\* pc = findCustomer(customerId);

newNode->customerId = pc->id;

newNode->customerName = pc->name;

system ("CLS");

cout<<"Enter product quantity: ";

int quantity;

cin>>quantity;

newNode->quantity = quantity;

p2->inStock = p2->inStock - quantity;

system ("CLS");

time\_t now = time(0);

char\* dt = ctime(&now);

newNode->date = dt;

newNode->id = currentSaleProductId;

newNode->totalPrice = quantity \* p2->salePrice;

newNode->next = NULL;

return newNode;

}

void saleProduct(){

saleProducts \*newNode;

saleProducts \*temp;

newNode = getPSNode();

currentSaleProductId++;

Table movies;

movies.add\_row({"Id", "PId", "PName", "CId", "CName", "Sale Price", "Quantity", "Total Price", "Date"});

movies.add\_row({to\_string(newNode->id), to\_string(newNode->productId), newNode->productName, to\_string(newNode->customerId), newNode->customerName, to\_string(newNode->salePrice), to\_string(newNode->quantity), to\_string(newNode->totalPrice), newNode->date});

for (size\_t i = 0; i < 7; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

movies[1][i].format()

.font\_color(Color::cyan)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

if(startPS == NULL){

startPS = newNode;

std::cout << movies << std::endl;

successMessage("Product sale successfully.");

}else{

temp = startPS;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

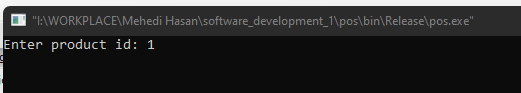
std::cout << movies << std::endl;

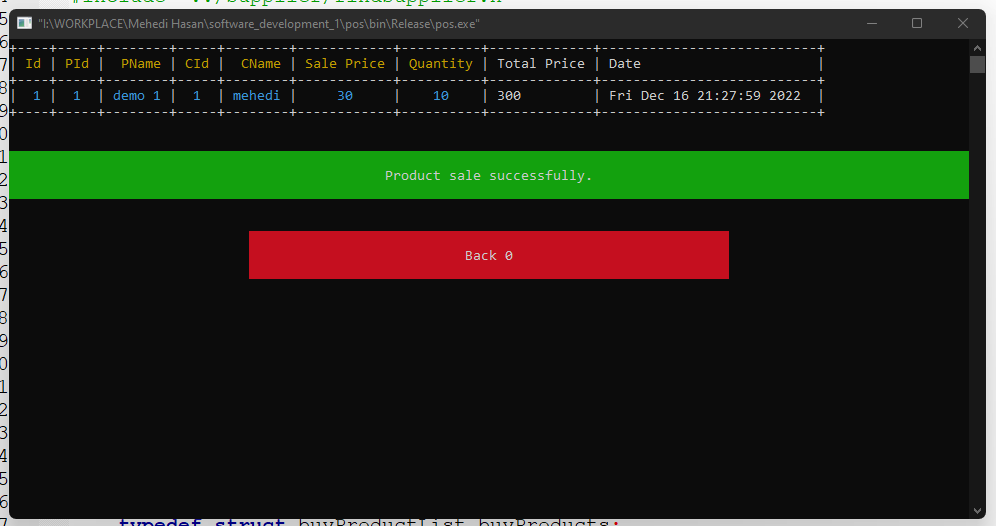
successMessage("Product sale successfully.");

}

}

**OUTPUT:**





**File Name:** viewSale.h

**Code:-**

#include <iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void viewSale(){

saleProducts \*dataPS1 = startPS;

Table movies;

movies.add\_row({"Id", "PId", "PName", "CId", "CName", "Sale Price", "Quantity", "Total Price", "Date"});

for (size\_t i = 0; i < 7; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

int nn = 0;

string total ;

while(dataPS1 != NULL){

nn++;

movies.add\_row({to\_string(dataPS1->id), to\_string(dataPS1->productId), dataPS1->productName, to\_string(dataPS1->customerId), dataPS1->customerName, to\_string(dataPS1->salePrice), to\_string(dataPS1->quantity), to\_string(dataPS1->totalPrice), dataPS1->date});

//cout<<dataPS1->customerName <<"--";

dataPS1 = dataPS1->next;

}

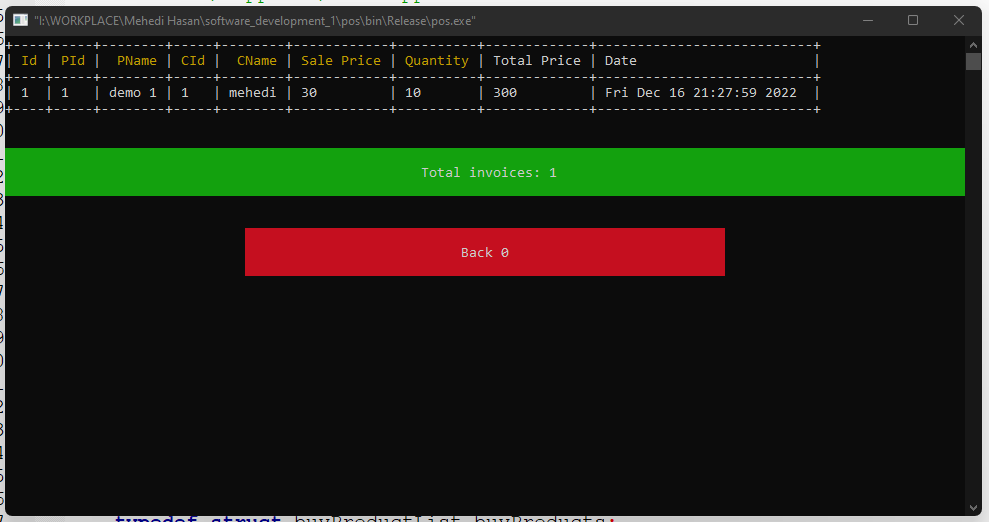
std::cout << movies << std::endl;

total = "Total invoices: " + to\_string(nn);

successMessage(total);

}

**OUTPUT:**



**File Name:** deleteProduct.h

**Code:-**

#include<iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void deleteProduct(){

viewProduct();

int id;

cout<< "Enter product id:";

cin>>id;

system ("CLS");

products \*dataFPDL = start;

products \*dataFPDL2;

products \*dataFPDL3;

if(start->id == id && start->next == NULL){

free(start);

start = NULL;

successMessage("Product delete successfully.");

}else{

if(start->id == id ){

free(start);

start = dataFPDL->next;

successMessage("Product delete successfully.");

}else{

while(dataFPDL != NULL){

if(dataFPDL->next->id == id){

if(dataFPDL->next->next == NULL){

dataFPDL2 = dataFPDL->next->next;

free(dataFPDL2);

dataFPDL->next = NULL;

successMessage("Product delete successfully.");

break;

}else{

dataFPDL2 = dataFPDL->next->next;

dataFPDL3 = dataFPDL->next;

free(dataFPDL3);

dataFPDL->next = dataFPDL2;

successMessage("Product0 delete successfully.");

break;

}

}

dataFPDL = dataFPDL->next;

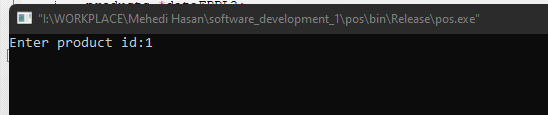
}

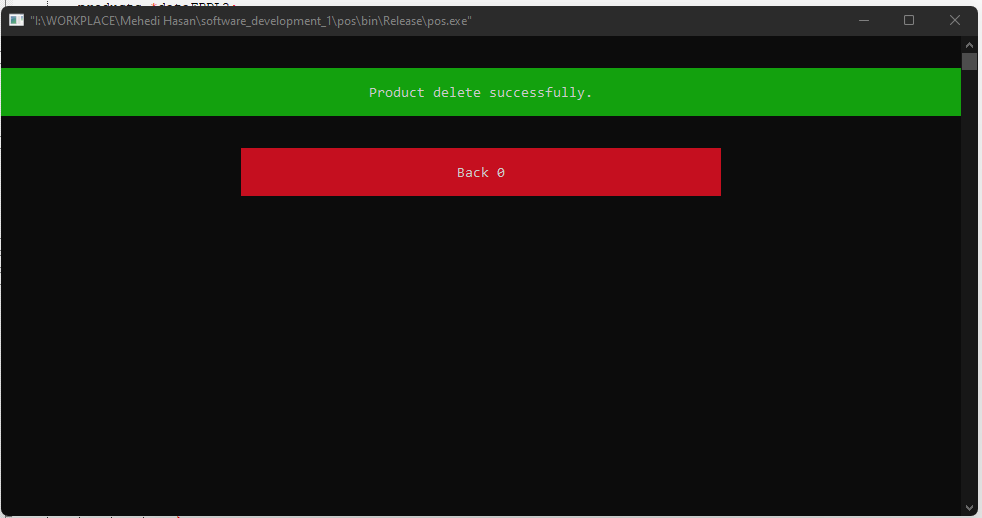
}

}

}

**OUTPUT:**





**File Name:** viewProduct.h

**Code:-**

#include <iostream>

#include "../../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void viewProduct(){

products \*dataP = start;

Table movies;

movies.add\_row({"Id", "Name", "Purchase Price", "Sale Price","In Stock", "Date"});

for (size\_t i = 0; i < 6; ++i) {

movies[0][i].format()

.font\_color(Color::yellow)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold});

}

int n = 0;

string total ;

while(dataP != NULL){

n++;

movies.add\_row({to\_string(dataP->id), dataP->name, to\_string(dataP->purchasePrice), to\_string(dataP->salePrice),to\_string(dataP->inStock), dataP->date});

dataP = dataP->next;

}

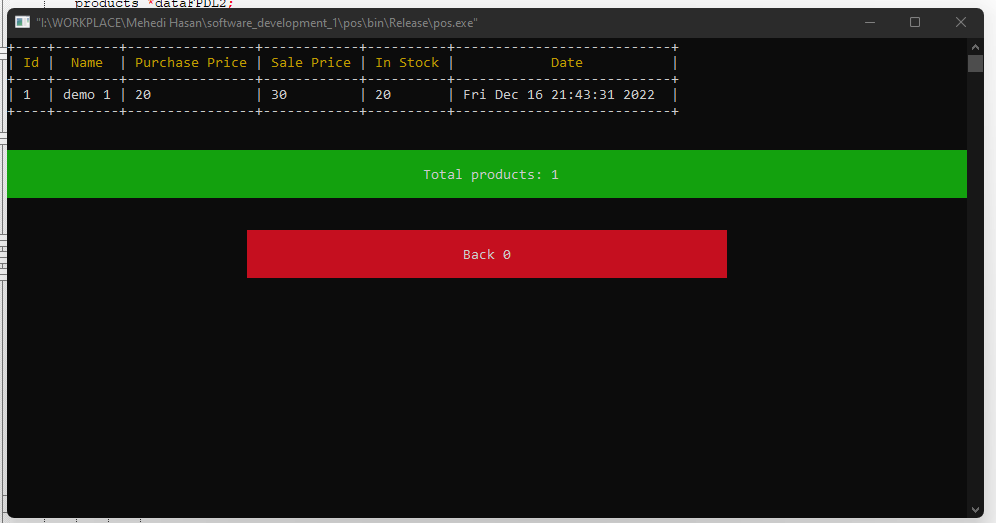
std::cout << movies << std::endl;

total = "Total products: " + to\_string(n);

successMessage(total);

}

**OUTPUT:**



**File Name:** errorMessage.h

**Code:-**

#include<iostream>

#include <string>

#include "../../tabulate/table.hpp"

using namespace std;

using namespace tabulate;

void errorMessage(string message){

int option;

Table universal\_constants;

universal\_constants.add\_row({"",message,""});

universal\_constants.add\_row({"","Back 0"});

universal\_constants.format()

.font\_style({FontStyle::bold})

.border\_top(" ")

.border\_bottom(" ")

.border\_left("")

.border\_right("")

.corner(" ");

universal\_constants[0][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::red)

.width(20);

universal\_constants[0][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::red);

universal\_constants[0][2].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.font\_background\_color(Color::red)

.width(30);

universal\_constants[1][0].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(30);

universal\_constants[1][1].format()

.padding\_top(1)

.padding\_bottom(1)

.font\_align(FontAlign::center)

.font\_style({FontStyle::bold})

.width(60)

.font\_background\_color(Color::green);

std::cout << universal\_constants << std::endl;

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

cin>>option;

system ("CLS");

}

**OUTPUT:**

