**Basic Food Ordering System**

**Designed as a part of the Udemy Course:**

**The Complete Introduction to C++ Programming**

**By Yassin Marco**

**About the project:** Basic food ordering system is designed based on the fundamentals of C++. The name given to this mini-application is All-Space. The project proceeds with creating an account for the user as the first module.

For simplicity, only the login id and password are needed to complete account creation. Furthermore, constraints are set on the lengths of the username and password respectively. After meeting the constraints, the account is created and stored successfully.

Next, the user is expected to login using the same credentials. Once again the entered credentials undergo verification with previously entered details. After that the user may or may not add balance to his /her wallet depending on choice.

In the next module, the user can now choose to or to not order food. If interested to order he/she may proceed with adding items to a cart. If the cost of all items (including a 10% tax) exceeds the balance in the wallet then the cart is reset and extra money needs to be added to the wallet to continue adding from the beginning. If not interested to order then the user can type the appropriate option and exit.

If the user still chooses to order then he/she can do so after adding sufficient balance. After adding all required food, the checkout is next.

In the checkout module, the summary of user preferences and costs charged for the items is displayed. The tax is also additionally added to the grand total. Finally, the “Thank You” prompt, wraps up the code.

**Note:** To avoid any data type compatibility issues, most of the numerical variables are declared as double in this project. The options declared as strings were also taken as Yes/No to make it easier to understand.

**Code Explanation**

* This is the basic initialization module which prompts user to create an account. It also initialises user\_food\_units array which will be used later.

#include<iostream>

using namespace std;

int main()

{

cout<<"Welcome to the All-Space app"<<endl;

cout<<"Create your account to continue"<<endl;

cout<<"Let's start with your user name: "<<endl;

string user,pass;

int user\_food\_units[5],j; //Number of units of each food ordered

for(j=0;j<5;j++)

{

user\_food\_units[j]=0;

}

* Next we have one of the verification modules. This module verifies whether the length of chosen username is >=6 and password is >=8.

while(1){ //Check length of user name

getline(cin,user);

if(user.length()>=6)

{

break;

}

else{

cout<<"Username should be at least 6 characters in length"<<endl;

}

}

cout<<"Enter a password: "<<endl; //user registration

while(1){ //Check length of password

getline(cin,pass);

if(pass.length()>=8)

{

cout<<"Thank You for registering!"<<endl;break;

}

else{

cout<<"Password should be at least 8 characters in length"<<endl;

}

}

* The next module expects the user to login again using the credentials specified above.

cout<<"\nLogin again to order"<<endl;

while(1){ //Check valid user name

cout<<"Username: ";

string username;

getline(cin,username);

if(username!=user) //(username!=user)||password!=pass) use this statement to combine the user name and password modules

{

cout<<"Username is invalid. Please re-enter"<<endl; //If stored username doesn't match then re-enter prompt

}

else

{

break;

}

}

while(1) //Check correct password

{

cout<<"Password: ";

string password;

getline(cin,password);

if(password!=pass)

{

cout<<"Password is invalid. Please re-enter"<<endl; //If stored password doesn't match then re-enter prompt

}

else{

break;

}

} //You can also customize the module by giving a limit on the number of attempts

* In the next module, the user gets a basic welcome prompt followed by a choice to or to not add money to the wallet. The initial balance in the wallet is assumed to be $100.

cout<<"\nWelcome to the All-Space app!"<<endl;

double balance=100.00;

cout<<"Your wallet balance is $"<<balance<<endl;

cout<<"Would you like to add money to your wallet?"<<endl;

string option;

getline(cin,option);

double addmoney;

if((option=="yes") || (option=="Yes") || (option=="YES"))

{

cout<<"Enter how much money you want to add: $";

cin>>addmoney;

balance+=addmoney;

cout<<"Your balance has been updated\nBalance = $"<<balance<<endl;//Add balance

}

else

{

cout<<"Your balance remains $"<<balance<<endl;

}

* In the next module the user can choose whether he/she is interested in ordering.

string order\_interest;

cout<<"Would you like to order?(Yes/No)"<<endl;

cin>>order\_interest;

int i;

* If the user decides to order then a list of 5 items is displayed from which one can choose from. The user can choose by typing the serial number of the item and pressing enter.

if((order\_interest=="yes") || (order\_interest=="Yes") || (order\_interest=="YES"))

{

cout<<"Select what you like to order"<<endl;

string items[]={"Burger-Combo","Pizza-Lunch-Combo","Burrito-pack","Spring-Roll","choco-lava cake"};

double items\_prices[]={55.00,68.75,30.50,45.75,50.00};

cout<<" Item Price\n";

for(i=0;i<5;i++)

{

cout<<i+1<<". "<<items[i]<<"---------"<<"$"<<items\_prices[i]<<endl;

}

cout<<"Which one do you choose?\n";

int choices;

string exit\_choice;

double total\_bill=0;

* Based on the user choices, the total bill is updated and a prompt appears every time a certain item is added to cart. All of this is done easily using the cases in switch statement. The user may also choose multiple units of the same item and that is kept track of by using the user\_food\_units array and thus a prompt appears in the end of every addition whether the user is interested in proceeding to checkout or not.

while(1){

cin>>choices;

switch(choices)

{

case 1:

cout<<items[choices-1]<<" has been added to cart\n"; //[choices-1] since array indexing is form 0

total\_bill+=items\_prices[choices-1]; //update total\_bill

user\_food\_units[choices]++;

break;

case 2:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices]++;

break;

case 3:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices]++;

break;

case 4:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices]++;

break;

case 5:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices]++;

break;

default: cout<<"Invalid choice! Please try again\n";break;

}

* At each step of adding an item, the wallet is checked whether sufficient balance is available. If balance is not enough at any stage, the user is immediately prompted to add additional balance to cover for the remaining expenses. If the user is not interested to order now, then the exit prompt appears and the program stops.

if(total\_bill\*1.1>balance) //Balance is not sufficient so reset the total\_bill to 0

{

total\_bill=0;

cout<<"The total amount is subject to tax. You may have in-sufficient balance."<<endl;

cout<<"Would you like to add balance and continue?(Yes/No)"<<endl;

cin>>option;

if((option=="yes") || (option=="Yes") || (option=="YES"))

{

cout<<"How much would you like to add?"<<endl;

cin>>addmoney;

balance+=addmoney;

cout<<"Added successfully! Updated balance = $"<<balance<<"\nFill your cart again to continue"<<endl;

}

else

{

cout<<"Looking forward to see you again! Have a good day."<<endl;

break;

}

}

* If the user ends up adding sufficient balance then the user can once again begin adding all the food items (from the beginning) required and once satisfied may proceed to checkout.

else if(total\_bill!=0)

{

int checkout\_choice;

cout<<"Proceed to checkout? \n(Yes then enter 1/No then enter 0)\n";

cin>>checkout\_choice;

if(checkout\_choice)

{

break;

}

else

{

cout<<"Very well! Continue adding....."<<endl;

}

}

}

* After proceeding to check-out the whole summary of the payment is displayed which includes: food items selected, number of food items selected, overall cost of each item, tax on overall bill, the grand total and finally updated wallet balance.

cout<<"\n\nYour bill summary (inclusive of all taxes) is as follows:"<<endl;

for(i=0;i<5;i++)

{

if(user\_food\_units[i])

{

cout<<items[i]<<"---"<<"units: "<<user\_food\_units[i]<<"---price: "<<user\_food\_units[i]\*items\_prices[i]<<endl;

}

}

cout<<"Total: $"<<total\_bill<<endl;

cout<<"Taxes: $"<<0.1\*total\_bill<<endl;

cout<<"Grand Total: $"<<1.1\*total\_bill<<endl; //Bill inclusive of taxes

balance-=1.1\*total\_bill;

cout<<"Updated balance: $"<<balance<<endl;

cout<<"\nThank You for considering our app! Have a good day.\n";

}

* If the user was not interested to order in the first place, then the exit prompt as follows:

else

{

cout<<"Looking forward to see you again! Have a good day."<<endl;

}

return 0;

}

/\*…………………………………..The End………………………………………………..\*/

**Overall Code**

#include<iostream>

using namespace std;

int main()

{

cout<<"Welcome to the All-Space app"<<endl;

cout<<"Create your account to continue"<<endl;

cout<<"Let's start with your user name: "<<endl;

string user,pass;

int user\_food\_units[5],j; //Number of units of each food ordered

for(j=0;j<5;j++)

{

user\_food\_units[j]=0;

}

while(1){ //Check length of user name

getline(cin,user);

if(user.length()>=6)

{

break;

}

else{

cout<<"Username should be at least 6 characters in length"<<endl;

}

}

cout<<"Enter a password: "<<endl; //user registration

while(1){ //Check length of password

getline(cin,pass);

if(pass.length()>=8)

{

cout<<"Thank You for registering!"<<endl;break;

}

else{

cout<<"Password should be at least 8 characters in length"<<endl;

}

}

cout<<"\nLogin again to order"<<endl;

while(1){ //Check valid user name

cout<<"Username: ";

string username;

getline(cin,username);

if(username!=user) //(username!=user)||password!=pass) use this statement to combine the user name and password modules

{

cout<<"Username is invalid. Please re-enter"<<endl; //If stored username doesn't match then re-enter prompt

}

else

{

break;

}

}

while(1) //Check correct password

{

cout<<"Password: ";

string password;

getline(cin,password);

if(password!=pass)

{

cout<<"Password is invalid. Please re-enter"<<endl; //If stored password doesn't match then re-enter prompt

}

else{

break;

}

} //You can also customize the module by giving a limit on the number of attempts

cout<<"\nWelcome to the All-Space app!"<<endl;

double balance=100.00;

cout<<"Your wallet balance is $"<<balance<<endl;

cout<<"Would you like to add money to your wallet?"<<endl;

string option;

getline(cin,option);

double addmoney;

if((option=="yes") || (option=="Yes") || (option=="YES"))

{

cout<<"Enter how much money you want to add: $";

cin>>addmoney;

balance+=addmoney;

cout<<"Your balance has been updated\nBalance = $"<<balance<<endl;//Add balance

}

else

{

cout<<"Your balance remains $"<<balance<<endl;

}

string order\_interest;

cout<<"Would you like to order?(Yes/No)"<<endl;

cin>>order\_interest;

int i;

if((order\_interest=="yes") || (order\_interest=="Yes") || (order\_interest=="YES"))

{

cout<<"Select what you like to order"<<endl;

string items[]={"Burger-Combo","Pizza-Lunch-Combo","Burrito-pack","Spring-Roll","choco-lava cake"};

double items\_prices[]={55.00,68.75,30.50,45.75,50.00};

cout<<" Item Price\n";

for(i=0;i<5;i++)

{

cout<<i+1<<". "<<items[i]<<"---------"<<"$"<<items\_prices[i]<<endl;

}

cout<<"Which one do you choose?\n";

int choices;

string exit\_choice;

double total\_bill=0;

while(1){

cin>>choices;

switch(choices)

{

case 1:

cout<<items[choices-1]<<" has been added to cart\n"; /\*[choices-1] since array indexing is form 0\*/

total\_bill+=items\_prices[choices-1]; //update total\_bill

user\_food\_units[choices-1]++;

break;

case 2:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices-1]++;

break;

case 3:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices-1]++;

break;

case 4:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices-1]++;

break;

case 5:

cout<<items[choices-1]<<" has been added to cart\n";

total\_bill+=items\_prices[choices-1];//update total\_bill

user\_food\_units[choices-1]++;

break;

default: cout<<"Invalid choice! Please try again\n";break;

}

if(total\_bill\*1.1>balance) //Balance is not sufficient so reset the total\_bill to 0

{

total\_bill=0;

cout<<"The total amount is subject to tax. You may have in-sufficient balance."<<endl;

cout<<"Would you like to add balance and continue?(Yes/No)"<<endl;

cin>>option;

if((option=="yes") || (option=="Yes") || (option=="YES"))

{

cout<<"How much would you like to add?"<<endl;

cin>>addmoney;

balance+=addmoney;

cout<<"Added successfully! Updated balance = $"<<balance<<"\nFill your cart again to continue"<<endl;

}

else

{

cout<<"Looking forward to see you again! Have a good day."<<endl;

break;

}

}

else if(total\_bill!=0)

{

int checkout\_choice;

cout<<"Proceed to checkout? \n(Yes then enter 1/No then enter 0)\n";

cin>>checkout\_choice;

if(checkout\_choice)

{

break;

}

else

{

cout<<"Very well! Continue adding....."<<endl;

}

}

}

cout<<"\n\nYour bill summary (inclusive of all taxes) is as follows:"<<endl;

for(i=0;i<5;i++)

{

if(user\_food\_units[i])

{

cout<<items[i]<<"---"<<"units: "<<user\_food\_units[i]<<"---price: "<<user\_food\_units[i]\*items\_prices[i]<<endl;

}

}

cout<<"Total: $"<<total\_bill<<endl;

cout<<"Taxes: $"<<0.1\*total\_bill<<endl;

cout<<"Grand Total: $"<<1.1\*total\_bill<<endl; //Bill inclusive of taxes

balance-=1.1\*total\_bill;

cout<<"Updated balance: $"<<balance<<endl;

cout<<"\nThank You for considering our app! Have a good day.\n";

}

else

{

cout<<"Looking forward to see you again! Have a good day."<<endl;

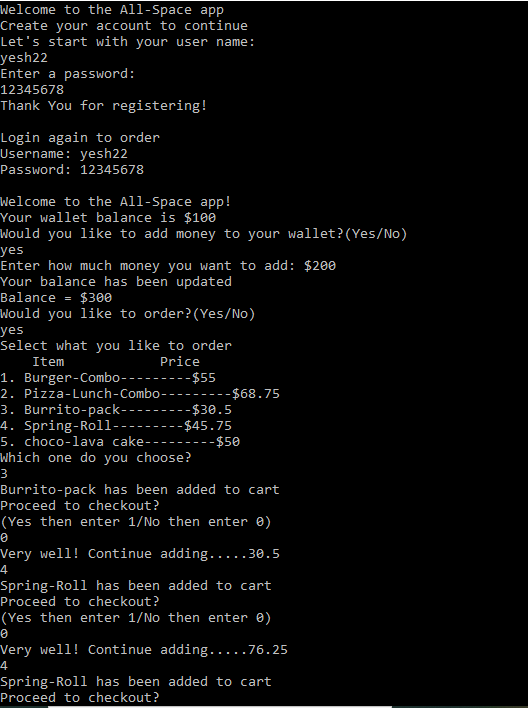
}

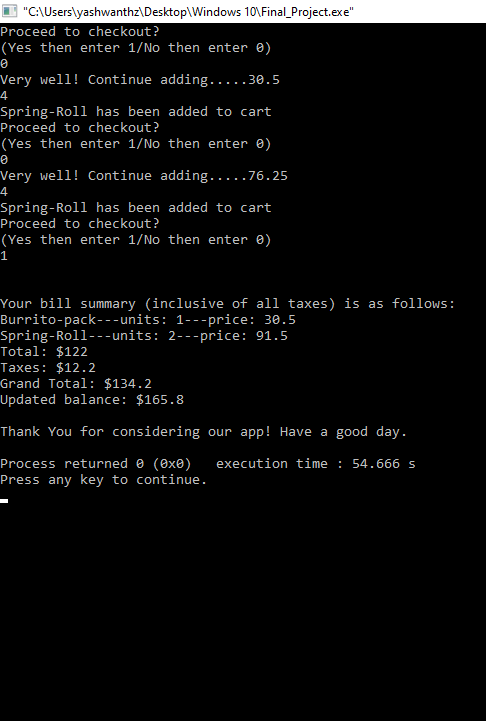
return 0;

}

/\*…………………………………..The End………………………………………………..\*/

**Sample Output**

****

****

**My suggestions to improve the code:**

* You can encrypt the password in \*s. Furthermore, you may also set constraints on how the password should be such as having at least one uppercase letter, special character or lowercase letter using various regular expressions.
* You may change the initial balance in the wallet based on your comfort. You may also add more food items to the array-list and thus more switch-cases.
* You may also customise the taxes and also make sure that user does not need to fill up the cart from scratch again and again by retaining all the cart additions and excluding the ones which created need for adding additional balance to the wallet. Overall you can maintain another set of switch cases for the user to choose after login: to add money to the wallet, to order, check previous order summaries, updating profile and so on.

**Thank You!**