**MOBILE SHOP MANAGEMENT SYSTEM**



Session: 2022 – 2026

**Submitted by:**

Afeera Fatima 2022-CS-151

**Supervised by:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[1. DESCRIPTION 3](#_Toc128721583)

[2. USERS OF APPLICATION: 3](#_Toc128721584)

[2.1 Owner of shop 3](#_Toc128721585)

[2.2 Employee 3](#_Toc128721586)

[2.3 Customer 3](#_Toc128721587)

[3. FUNCTIONAL REQUIREMENTS: 4](#_Toc128721588)

[3.1 Admin 4](#_Toc128721589)

[3.2 Employee 5](#_Toc128721590)

[3.3 Customer 6](#_Toc128721591)

[4. WireFrames 7](#_Toc128721592)

[4.1 Owner’s Wireframes 7](#_Toc128721593)

[4.3 Employee’s Wireframes 11](#_Toc128721594)

[5. DATA STRUCTURE 13](#_Toc128721595)

[5.1 Parallel Arrays 13](#_Toc128721596)

[5.2 Counter Variables 15](#_Toc128721597)

[6. FUNCTION PROTOTYPES 15](#_Toc128721598)

[6.1 File Handling 16](#_Toc128721599)

[6.2 All Functions 16](#_Toc128721600)

[7. FUNCTIONS WORKING FLOW 18](#_Toc128721601)

[8. Complete Code of Business Application 20](#_Toc128721602)

[9. Weakness in Business Application 103](#_Toc128721603)

[10. Future Directions 103](#_Toc128721604)

# DESCRIPTION

This is a basic mobile shop management system developed in C++ programming language. The system helps manage the operations of a mobile phone shop, including employee management, sales tracking, stock details, and customer reviews. With this system, the user can add, update, and delete products, view customer information, process sales .The system aims to simplify the operations of a mobile shop, making it more efficient and effective in managing its business.

# USERS OF APPLICATION:

Mainly, these are the users of this application:

# Owner of shop

He will keep an eye on employees of his shop and their attendance. He can view customer’s feedback so that he can make this better.

## Employee

He is the worker for mobile shop who update the stock details and store the daily record.

### **Customer**

Who purchases the product.

# FUNCTIONAL REQUIREMENTS:

## Admin

|  |  |  |
| --- | --- | --- |
| **User Id** | **As a** | **I want to perform** |
|  |  | Add and remove employees. |
| View employee’s list |
| Daily sale of the shop |
| Know what needs to be restock |
| Change Password |
| Attendance and Pay of Employees |
| View Categories and subcategories |
| View Feedback’s of Customer |

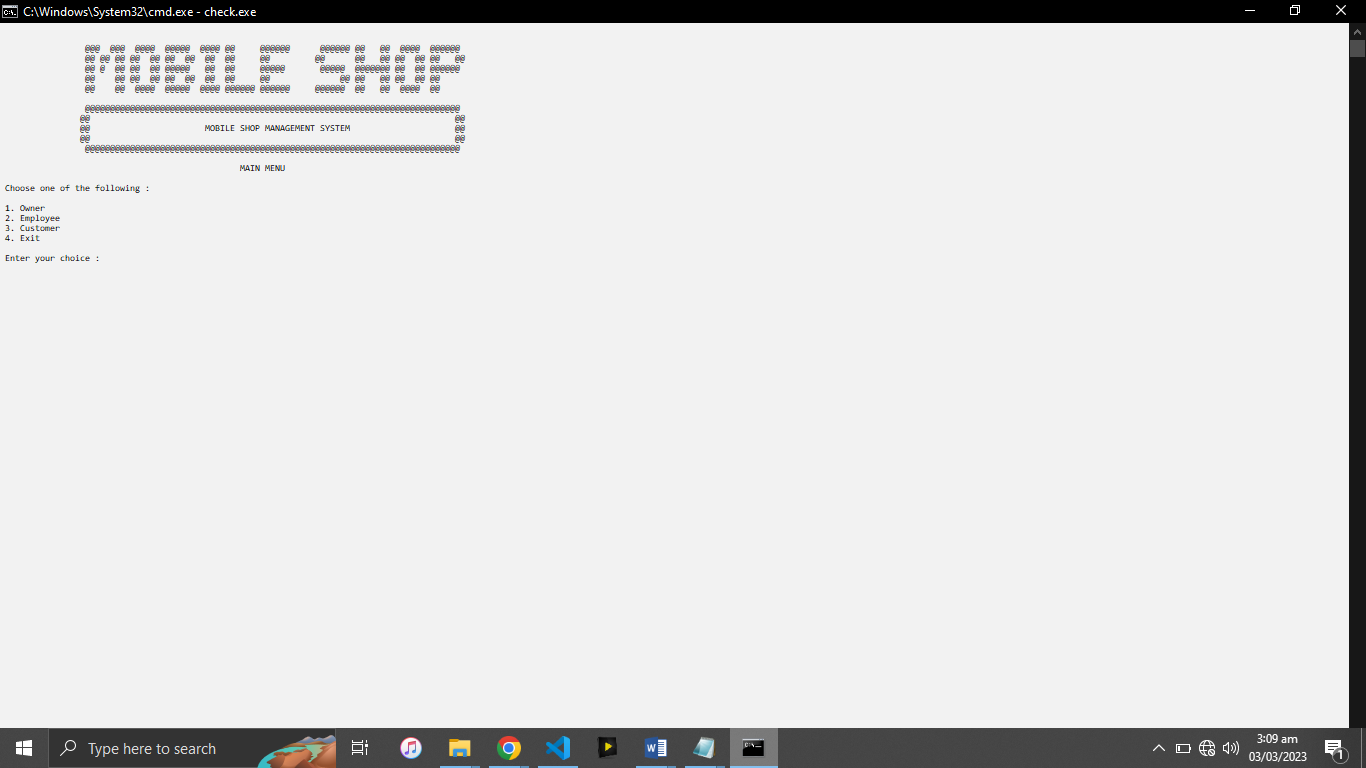
## Employee

|  |  |  |
| --- | --- | --- |
| **User Id** | **As a** | **I want to perform** |
| **2.** | **Employee** | Change Password |
| View total sale |
| View categories |
| Edit subcategories |
| Add stock |
| Remove stock |
| View attendance |
| Update price of items |
| Update attendance |
| View pay |

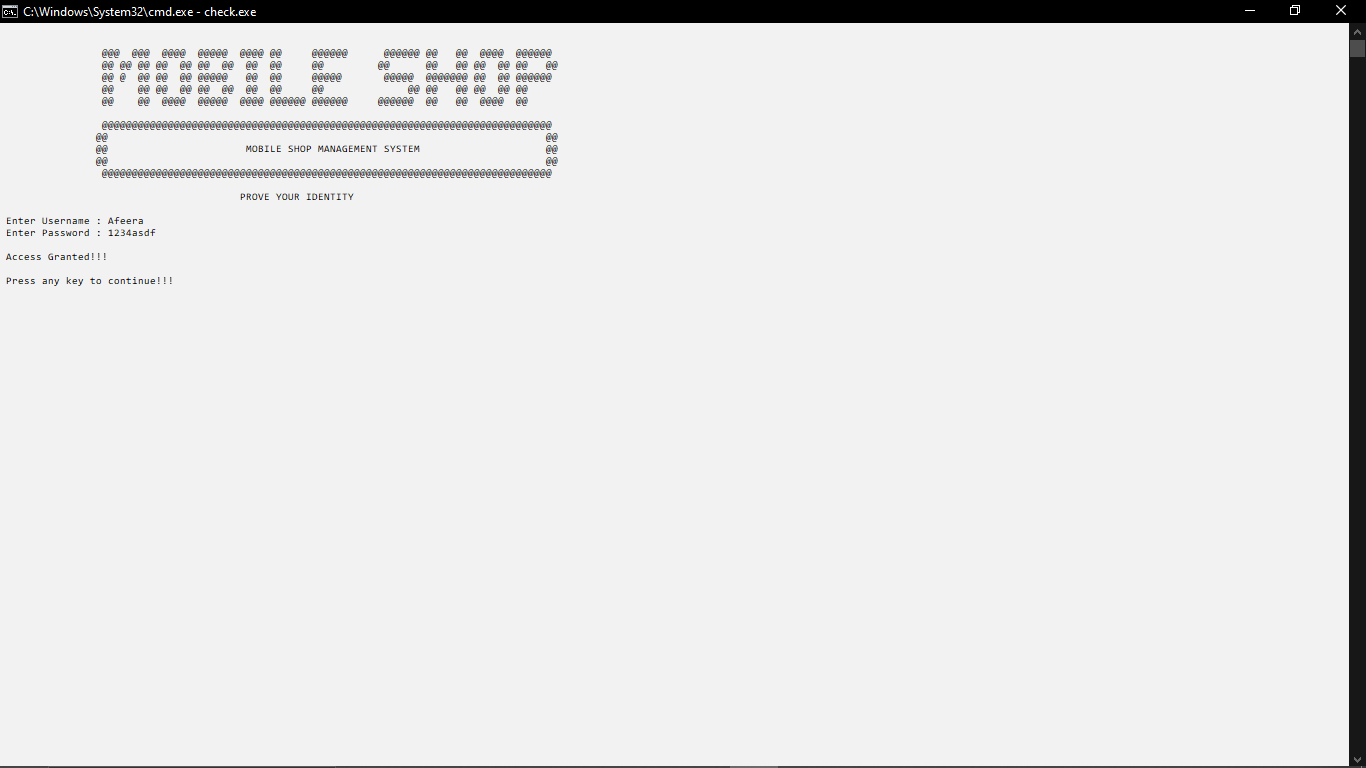
## Customer

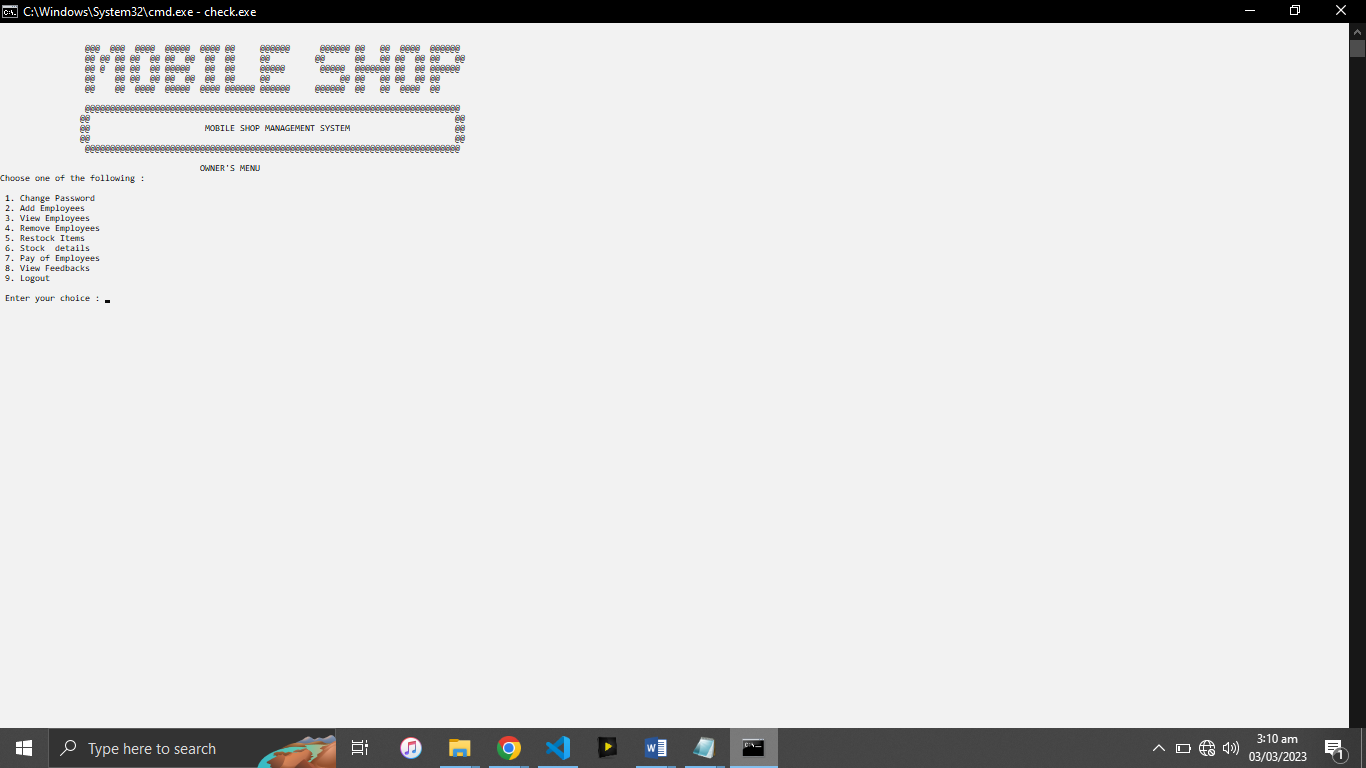
|  |  |  |
| --- | --- | --- |
| **User Id** | **As a** | **I want to perform** |
| **3.** | **Customer** | View categories of items |
| Enter his Budget |
| Search items according to budget |
| View Wallet |
| View cart |
| Calculate bill |
| Give feedback |

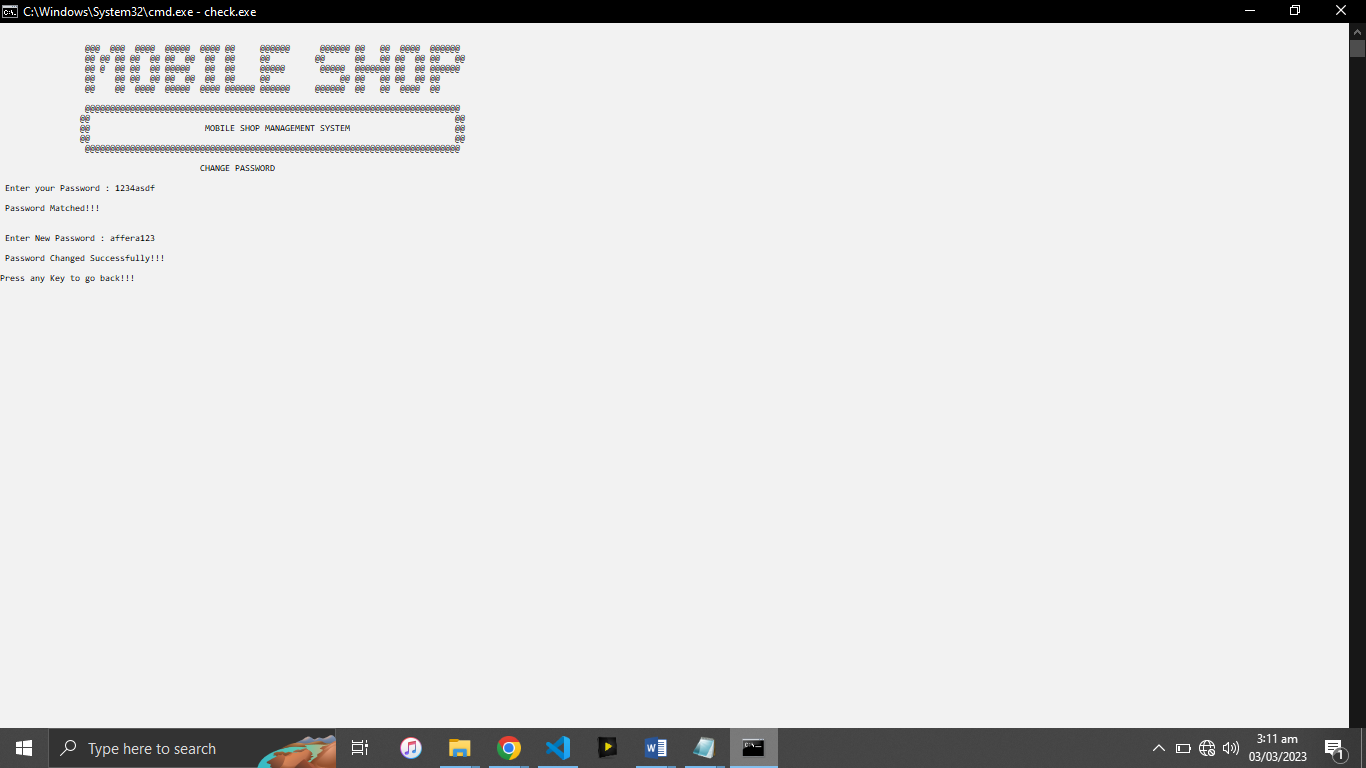
# WireFrames

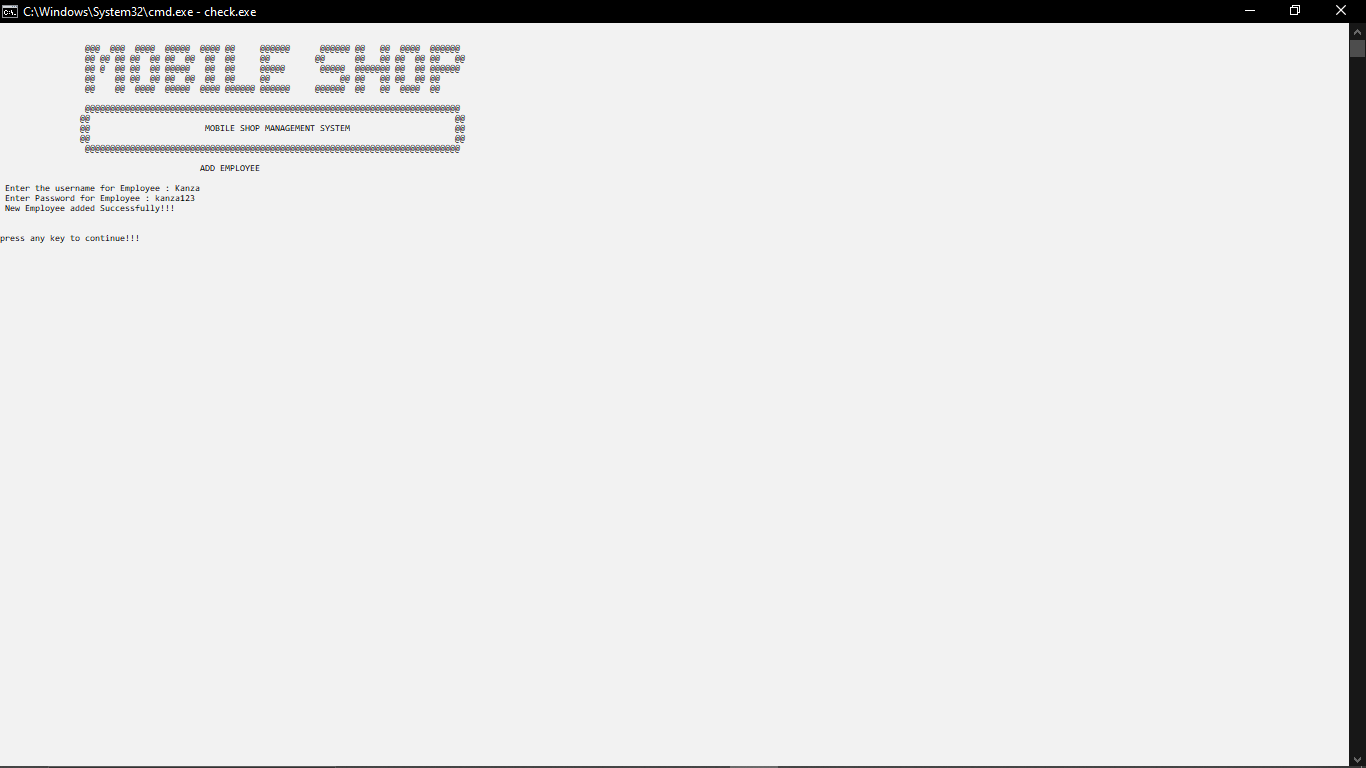


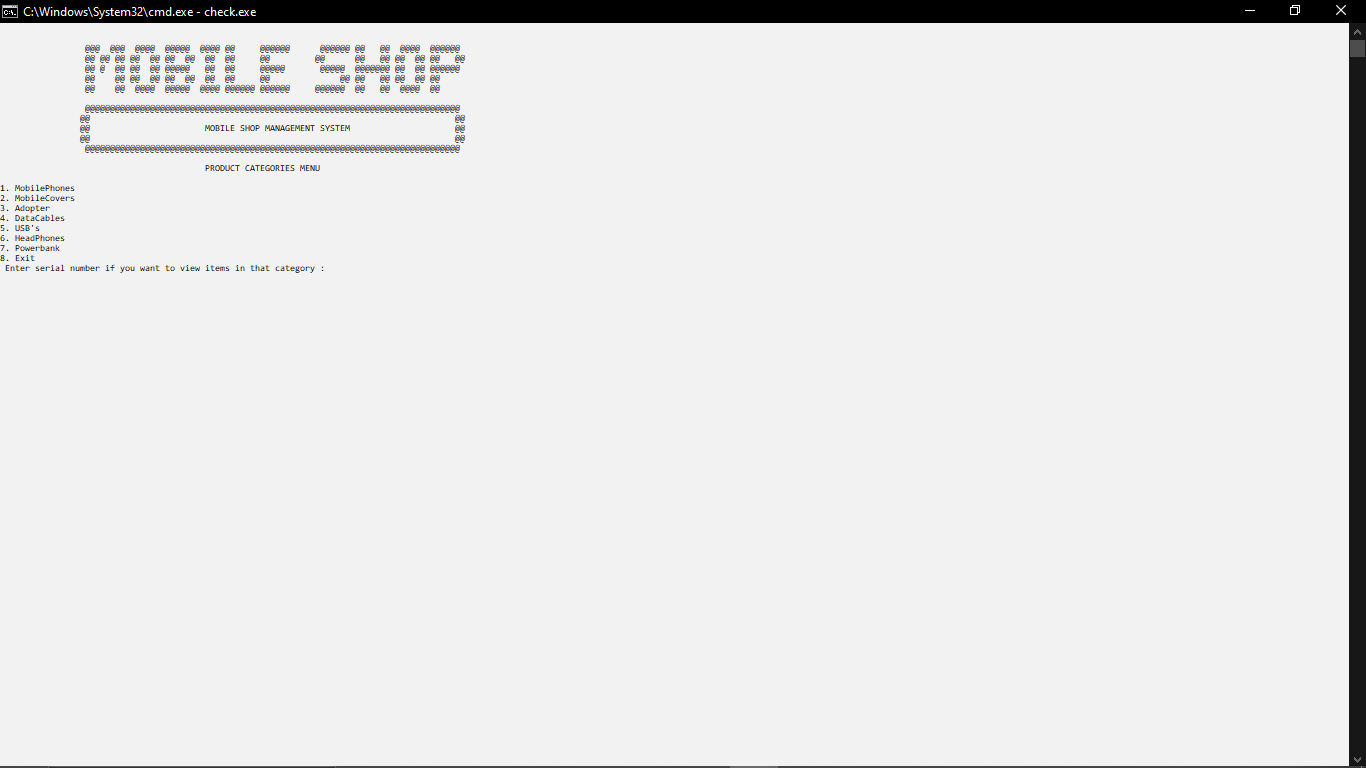
## Owner’s Wireframes

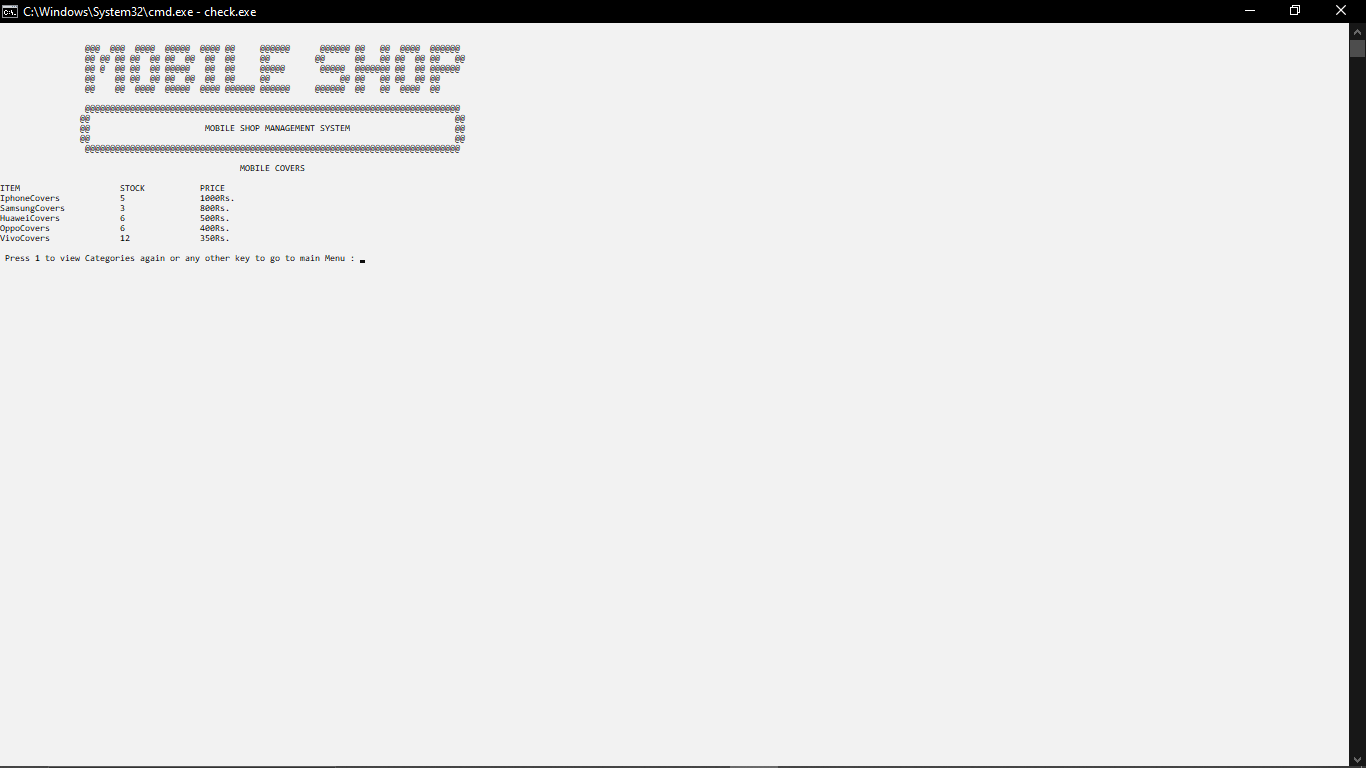


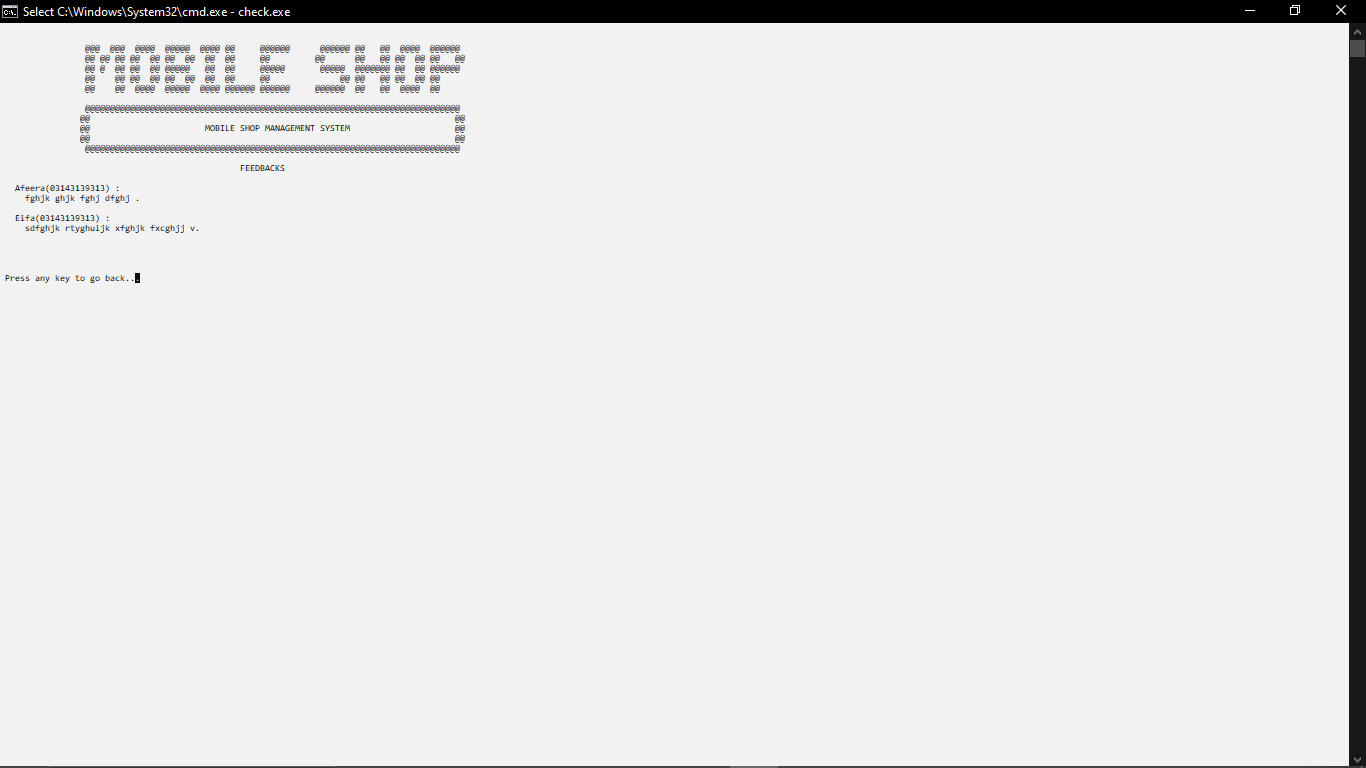


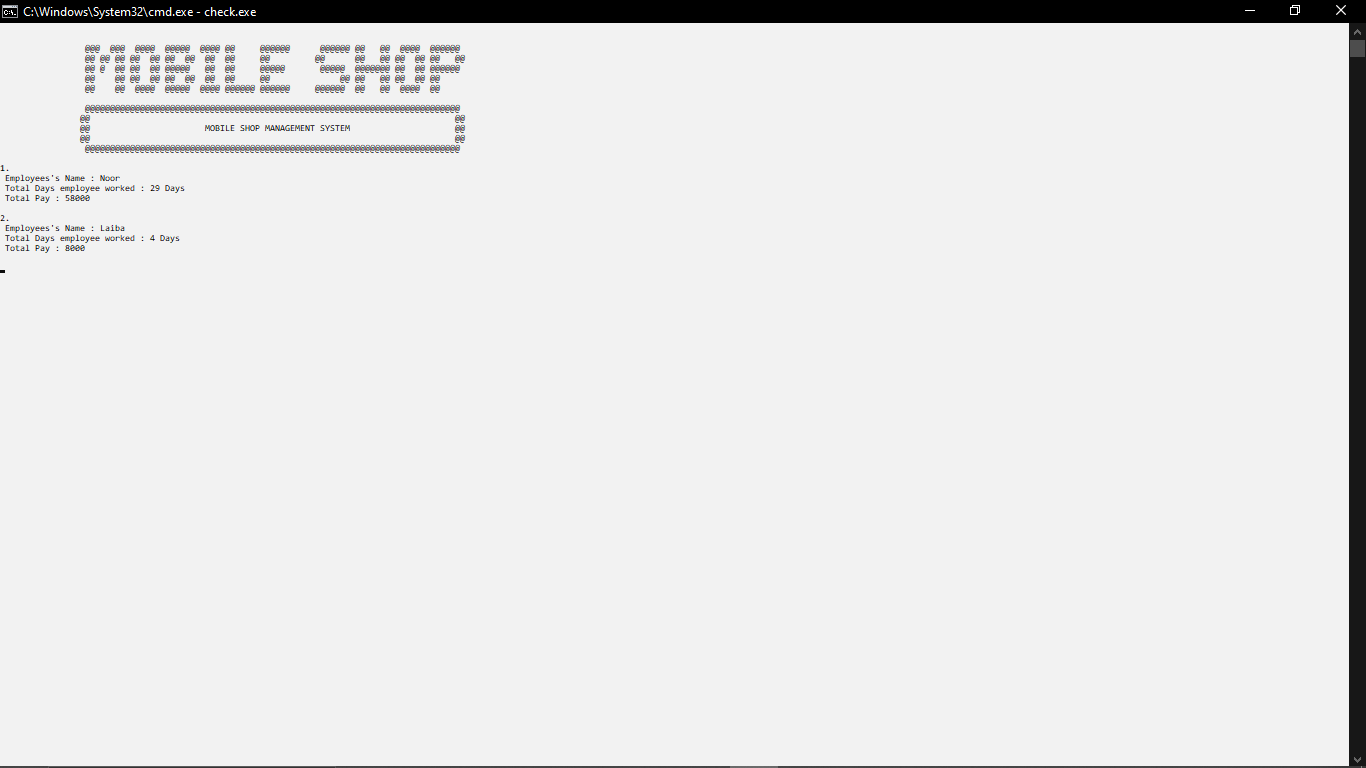




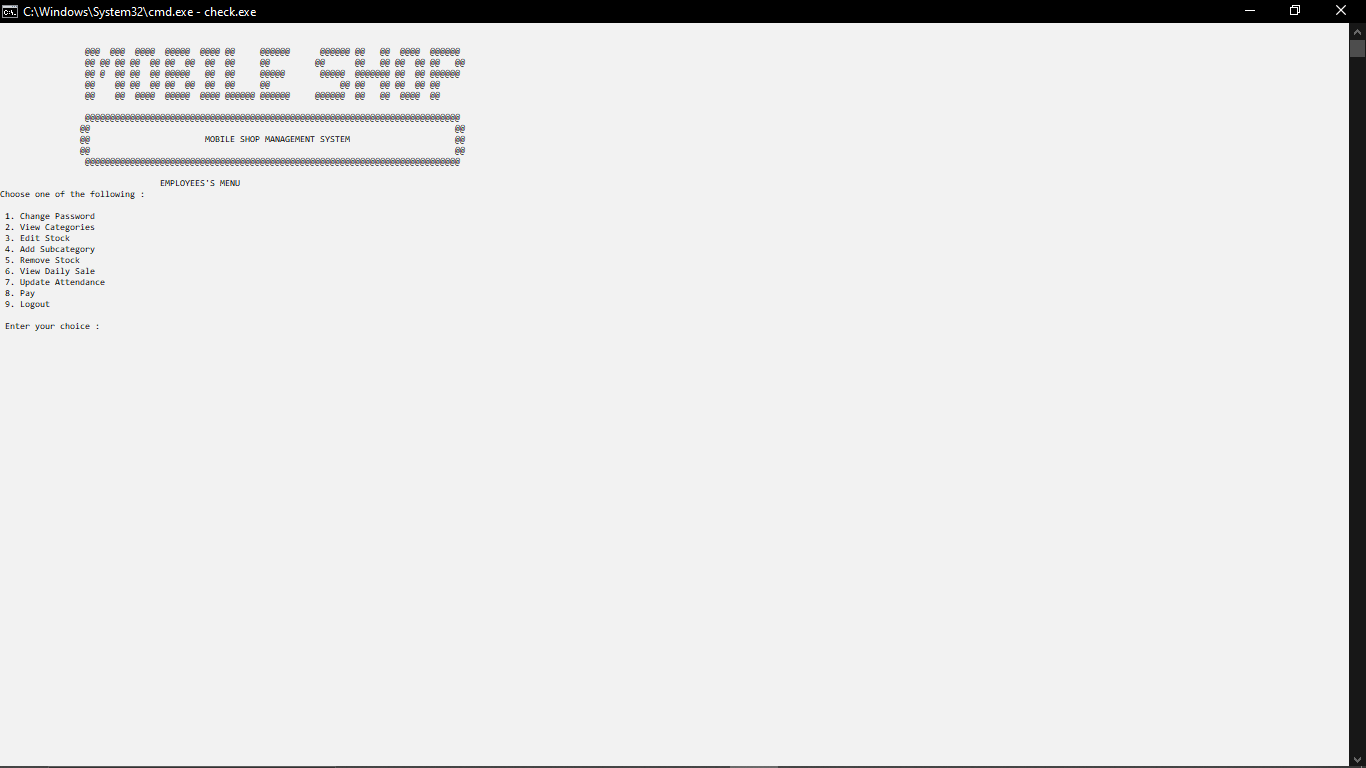


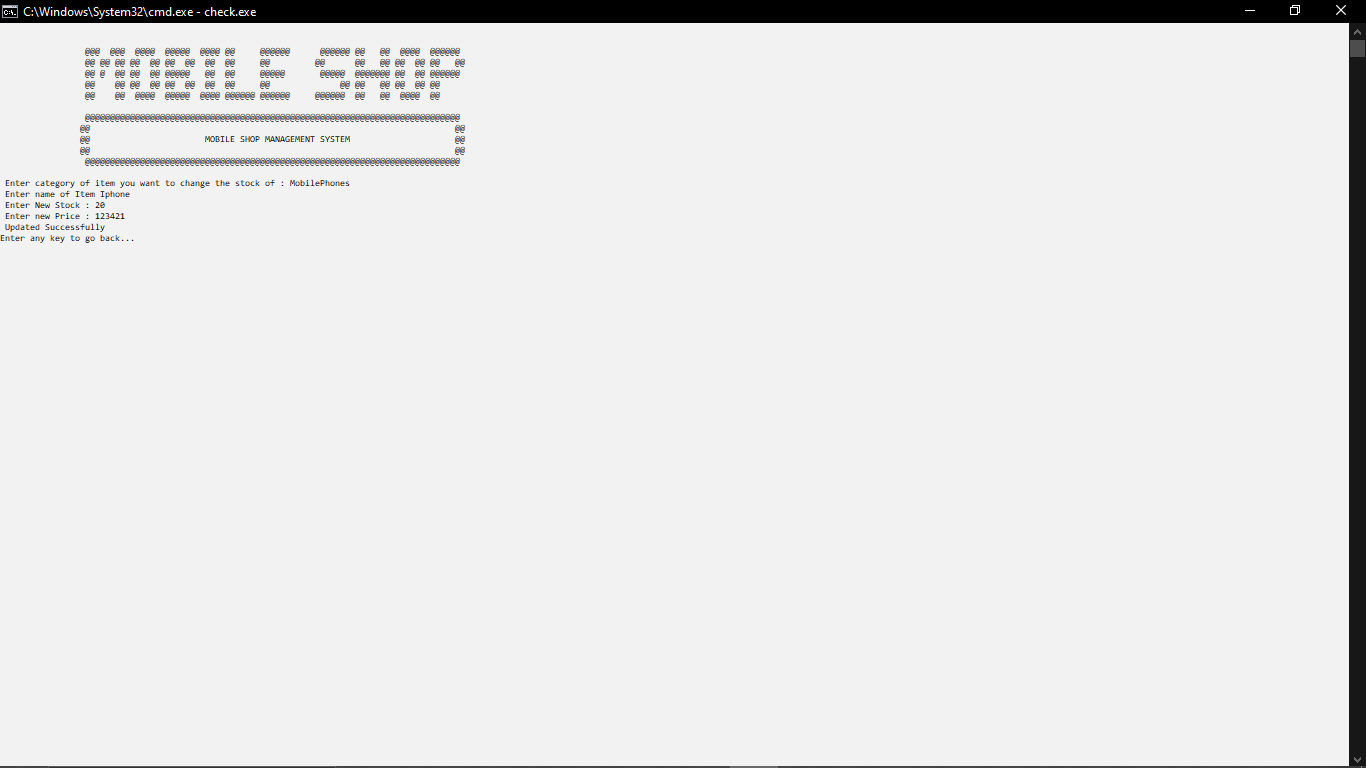
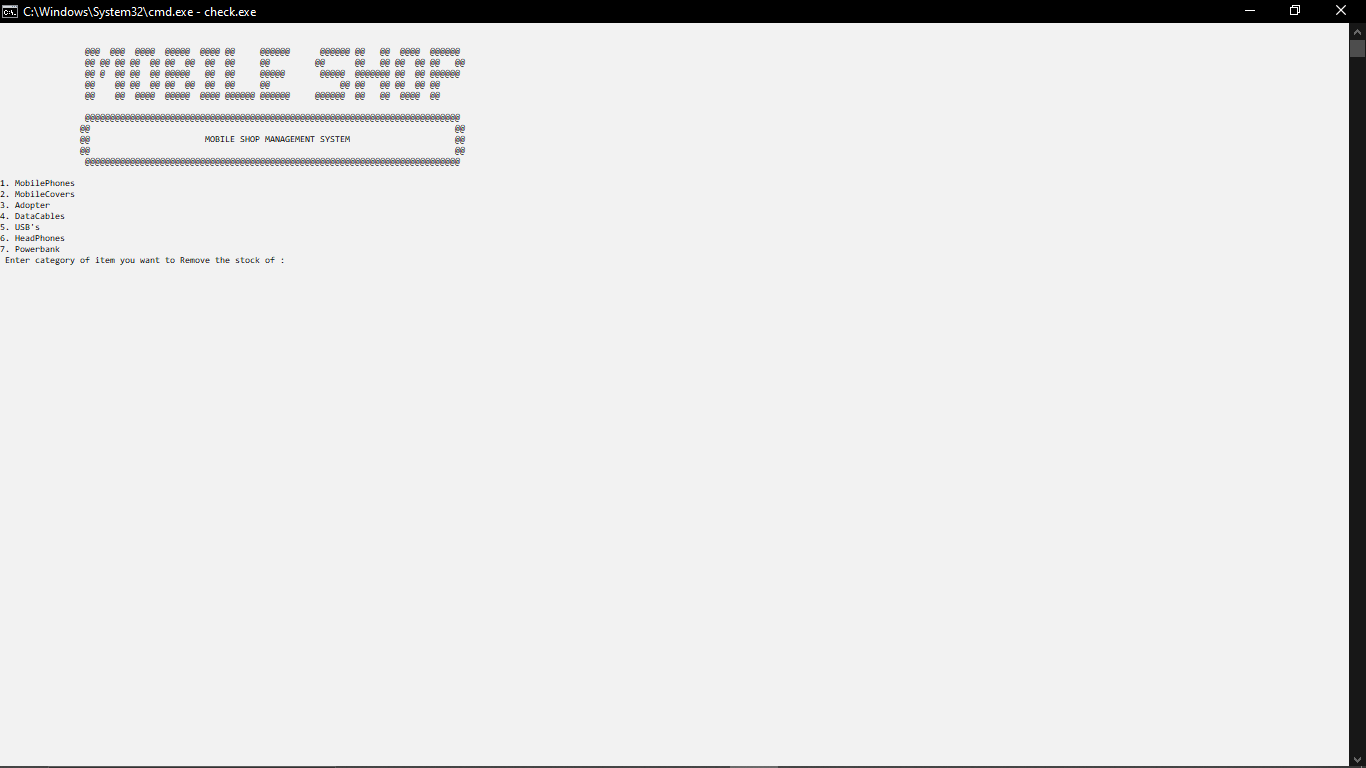


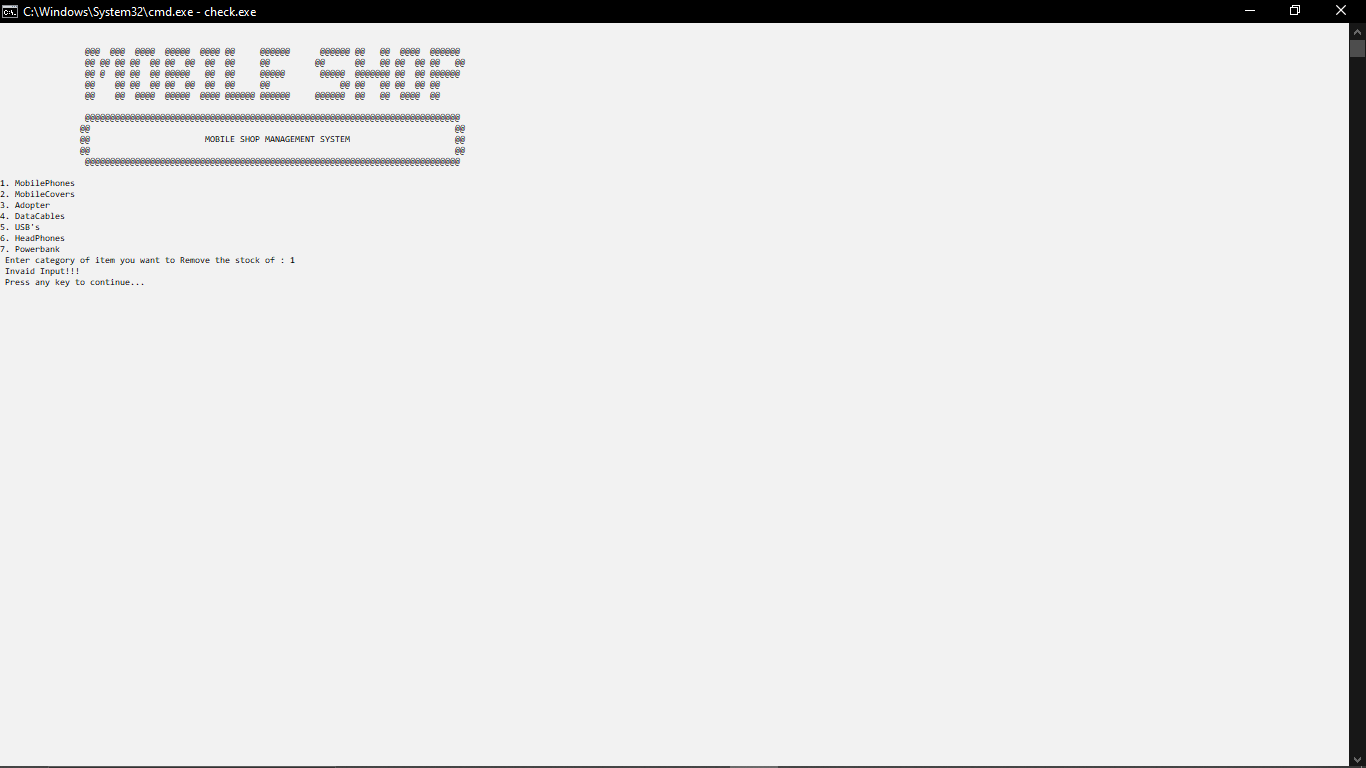


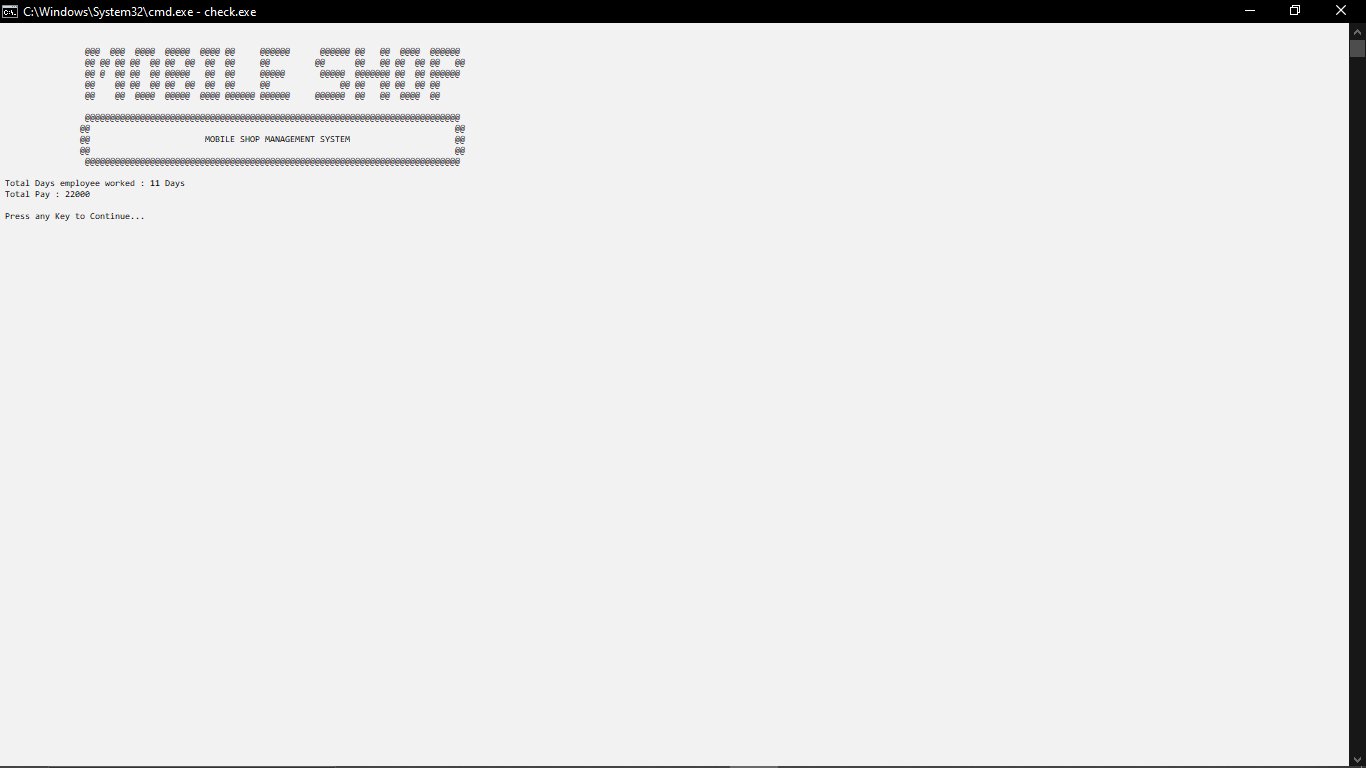


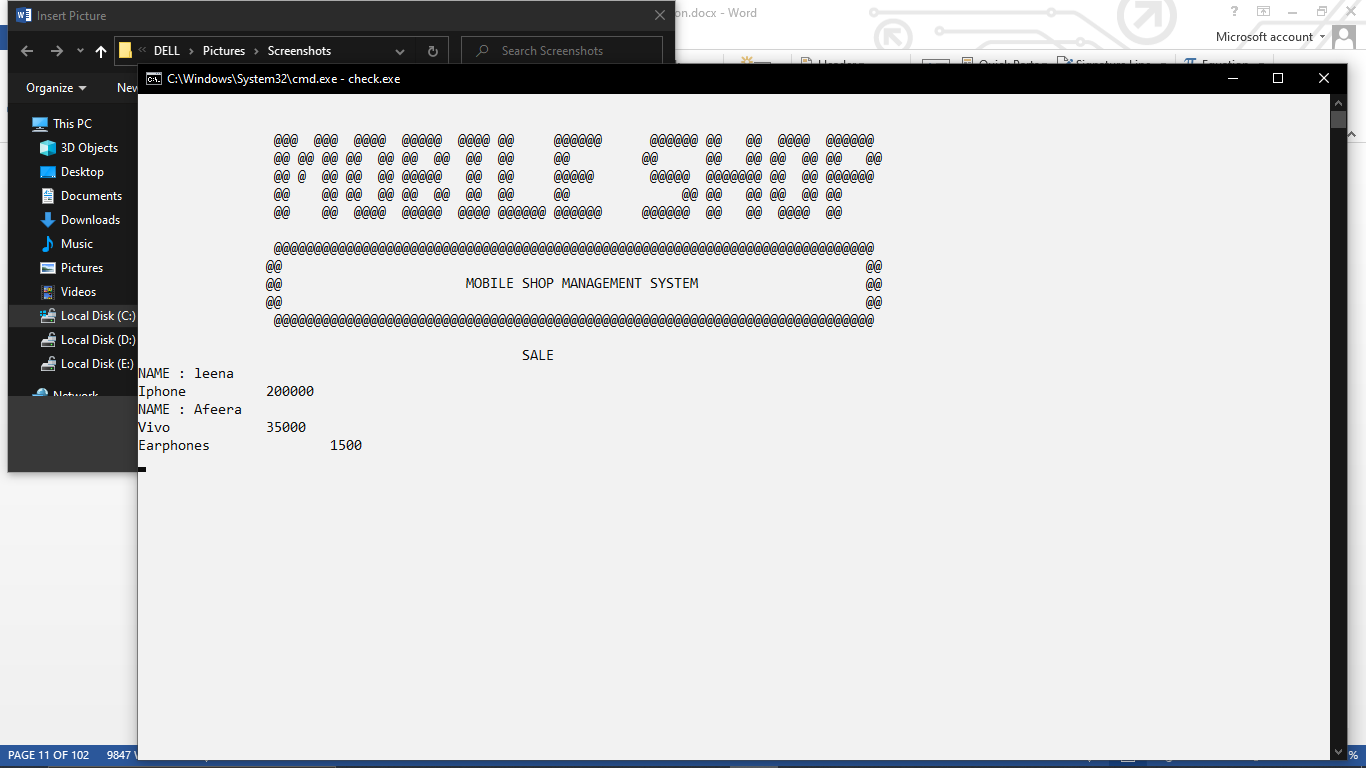
* 1. **Employee’s Wireframes**



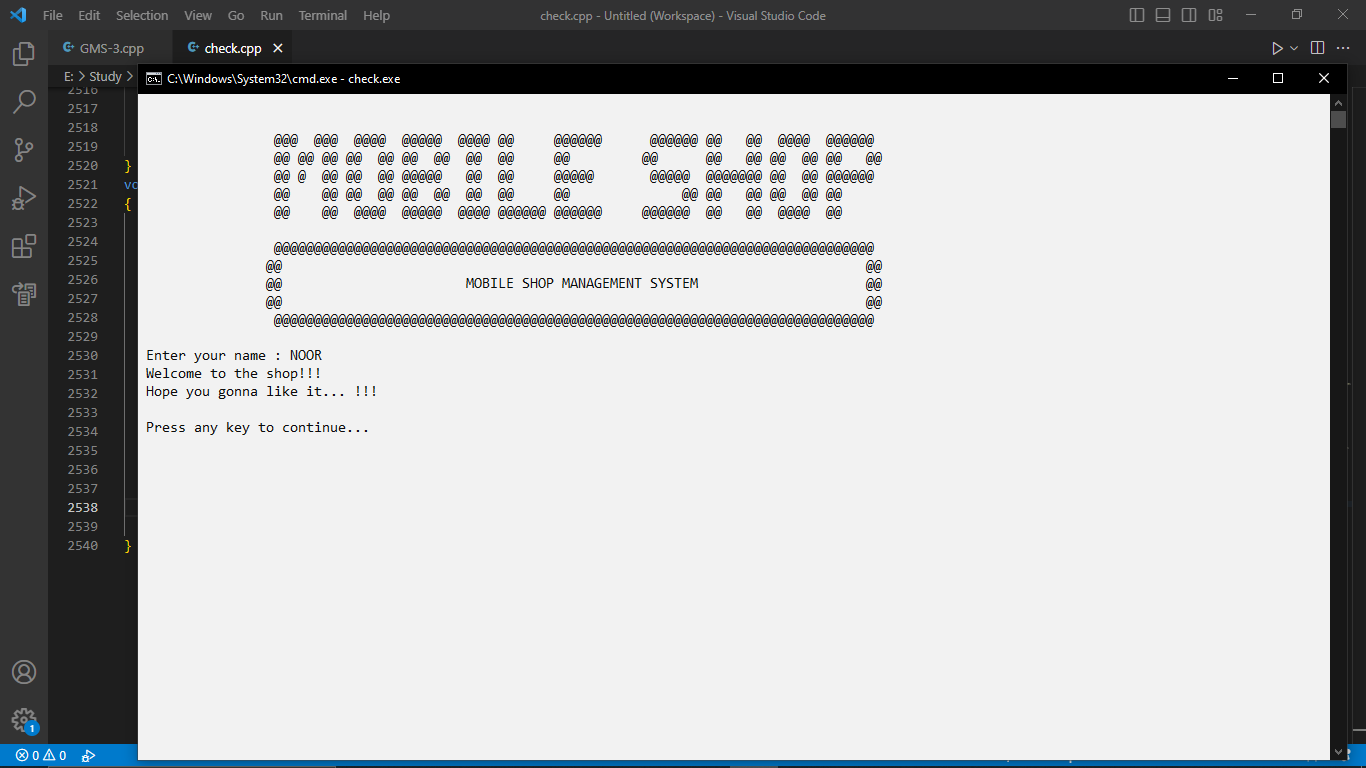


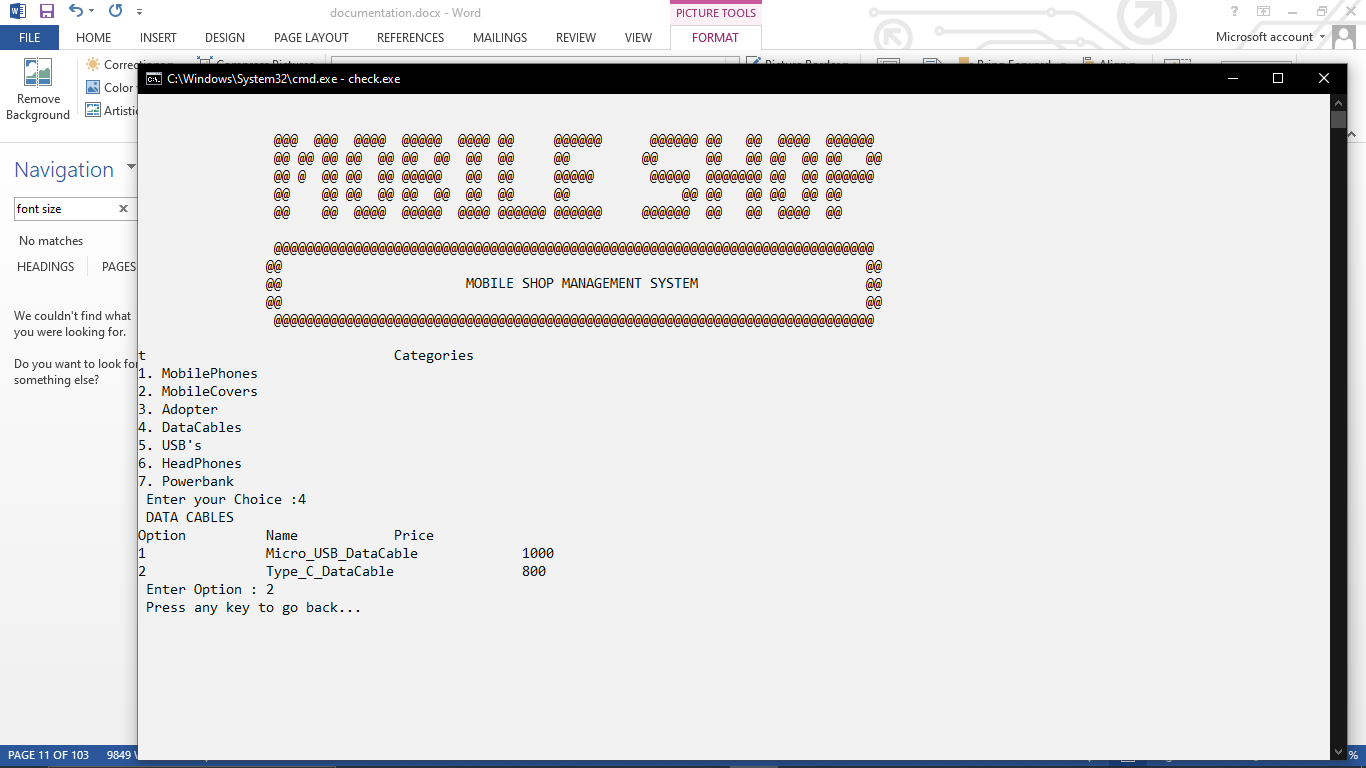


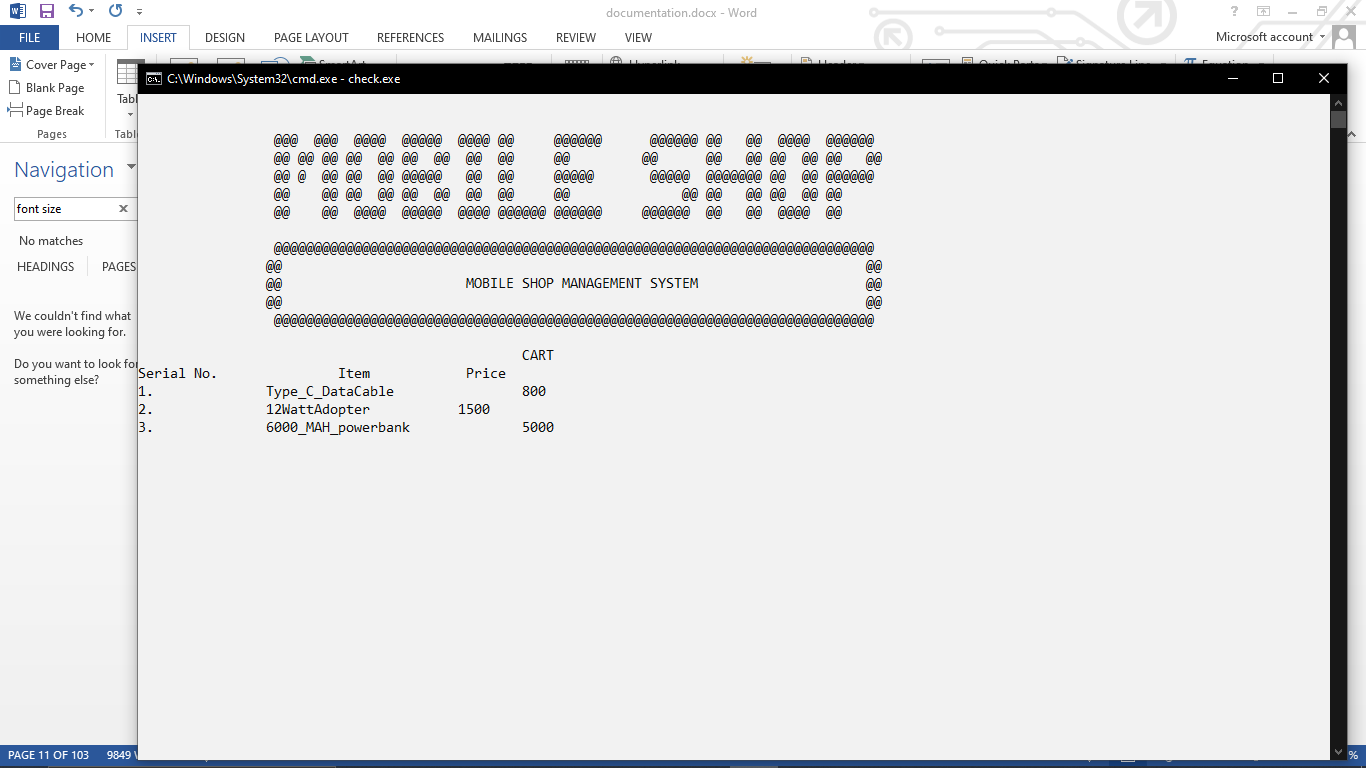
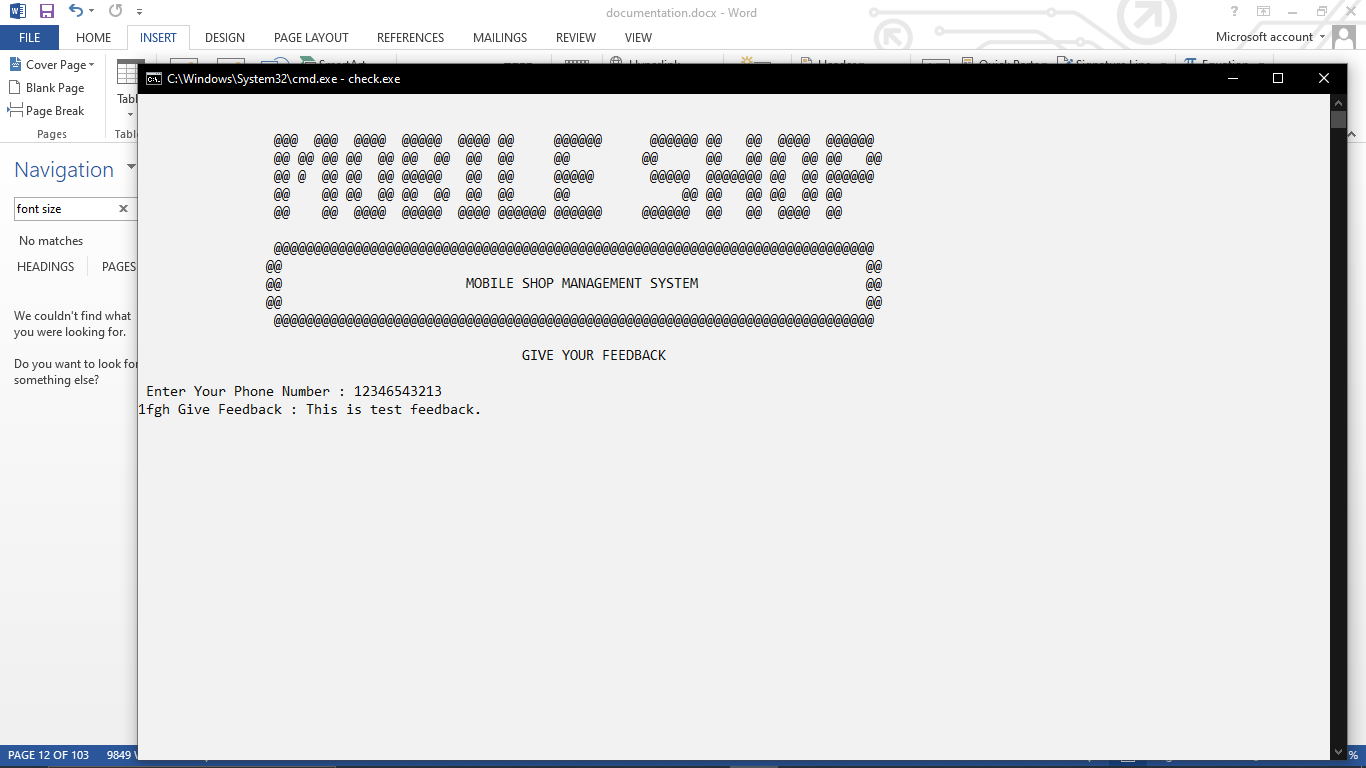


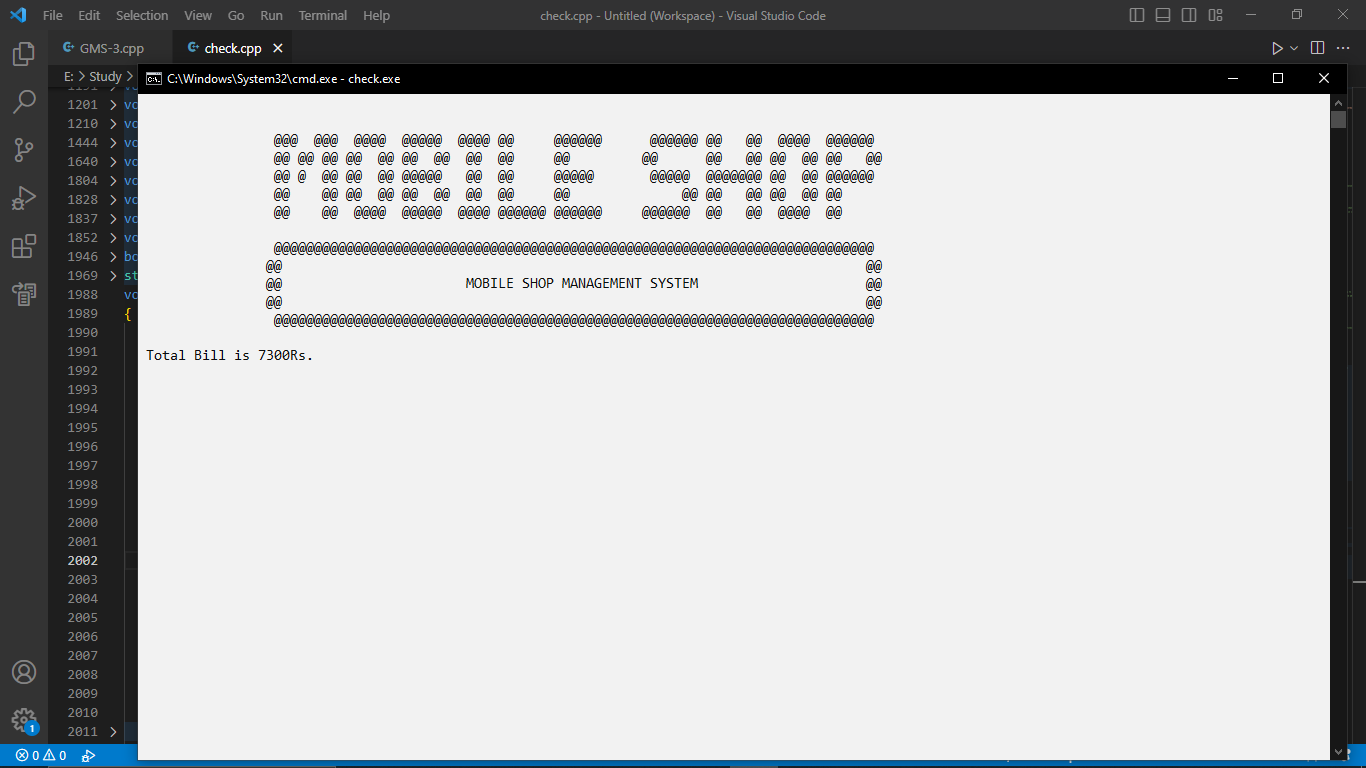


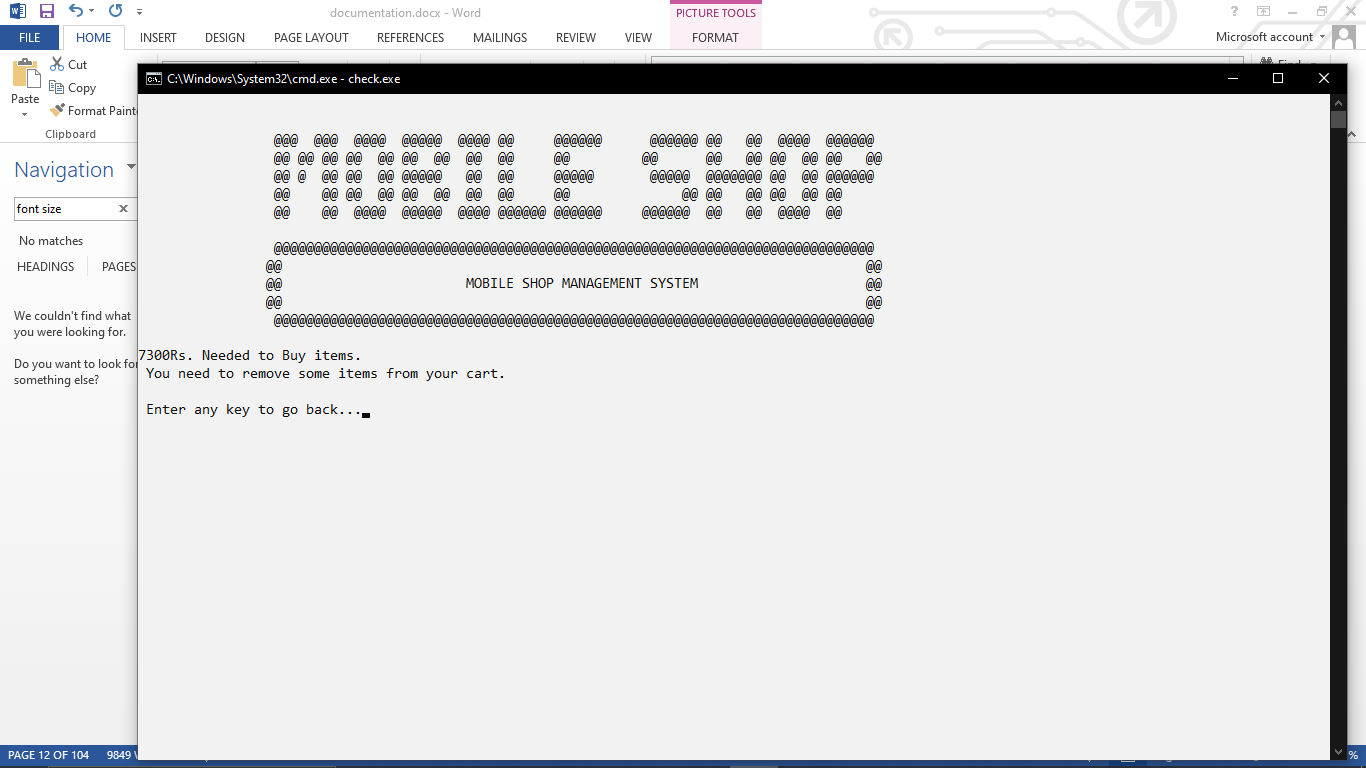
## Customer’s Wireframes











# DATA STRUCTURE

## Parallel Arrays

/\*User name and password\*/

string users[arraysize]; // username

string Passwords[arraysize]; // password

int count = 0; // it stores total number of users

/\*CATEGORIES\*/

string categories[7] = {"MobilePhones", "MobileCovers", "Adopter", "DataCables", "USB's", "HeadPhones", "Powerbank"};

///////////////////////////\*Sub Categories\*/////////////////////////

/\*MOBILES\*/

string categories\_Mobiles[arraysize];

int stock Mobiles[arraysize];

int price\_Mobiles[arraysize];

/\*COVERS\*/

string categories\_Covers[arraysize];

int stock\_Covers[arraysize];

int price\_Covers[arraysize];

/\*ADOPTERS\*/

string categories\_Adopters[arraysize];

int stock\_Adopters[arraysize];

int price\_Adopters[arraysize];

/\*CABLES\*/

int cables\_idx = 0; // total types of cables

string categories\_Cables[arraysize];

int stock\_Cables[arraysize];

int price\_Cables[arraysize];

/\*USB'S\*/

string categories\_USB[arraysize];

int stock\_USB[arraysize];

int price\_USB[arraysize];

/\*HEADPHONES\*/

string categories\_Headphones[arraysize];

int stock\_Headphones[arraysize];

int price\_Headphones[arraysize];

/\*POWERBANK\*/

string categories\_Powerbank[arraysize];

int stock\_Powerbank[arraysize];

int price\_Powerbank[arraysize];

/\*Attendance\*/

int attendance\_update[arraysize];

/\*Customer\*/

/\*CART VARIABLES\*/

string C\_Name[arraysize]; // customer's name

int C\_Budget[arraysize]; // customer's budget

string C\_Number[arraysize]; // customer's number

string C\_Comment[arraysize]; // customer's comment

string cartItem[arraysize][cart\_value]; // arraysize is same as customeridx and cartvalue store total items in cart

int cartItemPrice[arraysize][cart\_value];

int cart\_idx[arraysize];

int bill[arraysize];

## Counter Variables

const int arraysize = 50; // Array size

int count = 0; // it stores total number of users

int categories\_idx = 7; // categories idx

int mobiles\_idx = 0; // total types of mobile

int covers\_idx = 0; // total kinds of covers

int adopters\_idx = 0; // total types of adopters

int cables\_idx = 0; // total types of cables

int USB\_idx = 0; // total types of USB

int headphone\_idx = 0; // total types of headphone

int powerbank\_idx = 0; // total types of powerbank

int attendanceonlyonce = 0; // that allow employee to mark attendance only once

int employee = -10; // tempory variable that store idx of employee during employees menu to change values according to this idx

string attendance = " ";

int attendance\_idx = 10000;

int customer\_idx = 0; // customer count

int customer = 10000; // tempory variable that store idx of customer during employees menu to commit changes in his profile

const int cart\_value = 50;

# FUNCTION PROTOTYPES

## File Handling

/string getfield(string line, int field); // returns string to load functions accoriding to demand

void loadData(); // calls all the load functions

void storeData(); // calls all the store functions

void loadusers(); // load user's data(name,password,employee's attendance) from txt files

void storeusers(); // store user's data to txt files

void loadMobiles(); // load mobile's data(namr,stock,price) from txt files

void storeMobiles(); // stores mobile's data(namr,stock,price) to txt files

void loadCovers(); //...............

void storeCovers(); //...............

void loadAdopters(); //...............

void storeAdopters(); //...............

void loadCables(); //...............

void storeCables(); //...............

void loadUSB(); //...............

void storeUSB(); //...............

void loadHeadphones(); //...............

void storeHeadphones(); //...............

void loadPowerbank(); //...............

void storePowerbank(); //...............

int commaCount(string line); // count's commma's to tell us how many words are stored in line

void storeCart(); // store customer's information

void loadcart(); //load customer’s information

## All Functions

bool intCheck(string numeric); // checks whether the input value is integer

bool validnumber(string number); // checks whether the input value is 12 digitnumber

bool password\_Check(string Password); // checks whether the input value is valid for password

bool O\_identityCheck(); // Owner's identity Check function

bool nameCheck(string word); // checks whether name is valid

bool nameExists(string name); // whether user exists before

bool checkEmployees(string username, string password);

bool E\_identityCheck(); // checks employees identity

void mainHeader();

void topHeader();

void clearScreen(); // clears screen and calls topheader and mainheader

string mainMenu(); // shows main menu ans returns option entered by user to main

string ownersMenu(); // shows owner's menu and returns option entered by user

void O\_changePassword(); // to change owner's password

void addEmployees(); // allows owner to add employees

void viewEmployees(); // allows owner to view employees

void employeesPay(); // Show's Employees pay

void removeEmployees(); // allows owner to remove employees

string employeesMenu(); // shows employees's menu and returns option entered by user to main

void E\_changePassword(); // to change employee's password

void StockMobiles(); // show's stock of mobiles

void StockHeadphone(); // show's stock of headphones

void StockPowerbank(); // show's stock of powerbank

void StockUSB(); // show's stock of USB's

void StockCables(); // show's stock of Cables

void StockAdopters(); // show's stock of Adopters

void StockCovers(); // show's stock of Covers

void productCategories(); // shows all the categories shop have

void O\_employeesPay(); // shows all employee's pay and attendance to owner

void editStock(); // allows to edit stock details of subcategories

void addStock(); // allows to add stock in defined categories

void removeStock(); // allows to remove stock in defined categories

void updateAttendance(); // allow's employee to update attendance

void restock\_items(); // shows whether there is any need to restock items or not

void O\_feedback(); // shows all the customer's feedback

void C\_feedback(); // shows menu to customer to add feedbank and store's it in arrays accoring to customer's idx

string customersMenu(); // shows employees's menu and returns option entered by user to main

string customerCategoriesMenu(); // shows categories menu to customer and returns category number to view and add items in cart

void searchItems(); // search items according to budget

void customersBudget(); // add budget by customer to view whether he can buy items or not

void billStatus(); // compares price of items and budget of user

void viewCart(); // view customer's cart

void viewBill(); // view total bill to customer

void totalSale(); // show's total sale of the shop to employee

void calculatebill(); // this function calculates bill

void addtoCart(int option); // allows employee to add items in cart

bool C\_identityCheck(); // check's customer identity if he is new or not

void Employeeslist(); // shows employees name

void viewcategories(); // view all the categories

# 

# Funtions Working Flow

# Complete Code of Business Application

#include <iostream>

#include <fstream>

#include <conio.h>

#include <windows.h>

#include <string>

using namespace std;

/\*User name and password\*/

const int arraysize = 50; // Array size

string users[arraysize]; // username

string Passwords[arraysize]; // password

int count = 0; // it stores total number of users

/\*CATEGORIES\*/

int categories\_idx = 7;

string categories[7] = {"MobilePhones", "MobileCovers", "Adopter", "DataCables", "USB's", "HeadPhones", "Powerbank"};

///////////////////////////\*Sub Categories\*/////////////////////////

/\*MOBILES\*/

int mobiles\_idx = 0; // total types of mobile

string categories\_Mobiles[arraysize];

int stock\_Mobiles[arraysize];

int price\_Mobiles[arraysize];

/\*COVERS\*/

int covers\_idx = 0; // total kinds of covers

string categories\_Covers[arraysize];

int stock\_Covers[arraysize];

int price\_Covers[arraysize];

/\*ADOPTERS\*/

int adopters\_idx = 0; // total types of adopters

string categories\_Adopters[arraysize];

int stock\_Adopters[arraysize];

int price\_Adopters[arraysize];

/\*CABLES\*/

int cables\_idx = 0; // total types of cables

string categories\_Cables[arraysize];

int stock\_Cables[arraysize];

int price\_Cables[arraysize];

/\*USB'S\*/

int USB\_idx = 0; // total types of USB

string categories\_USB[arraysize];

int stock\_USB[arraysize];

int price\_USB[arraysize];

/\*HEADPHONES\*/

int headphone\_idx = 0; // total types of headphone

string categories\_Headphones[arraysize];

int stock\_Headphones[arraysize];

int price\_Headphones[arraysize];

/\*POWERBANK\*/

int powerbank\_idx = 0; // total types of powerbank

string categories\_Powerbank[arraysize];

int stock\_Powerbank[arraysize];

int price\_Powerbank[arraysize];

/\*Attendance\*/

int attendanceonlyonce = 0; // that allow employee to mark attendance only once

int employee = -10; // tempory variable that store idx of employee during employees menu to change values according to this idx

string attendance = " ";

int attendance\_idx = 10000;

int attendance\_update[arraysize];

/\*Customer\*/

/\*CART VARIABLES\*/

string C\_Name[arraysize]; // customer's name

int C\_Budget[arraysize]; // customer's budget

string C\_Number[arraysize]; // customer's number

string C\_Comment[arraysize]; // customer's comment

int customer\_idx = 0; // customer count

int customer = 10000; // tempory variable that store idx of customer during employees menu to commit changes in his profile

const int cart\_value = 50;

string cartItem[arraysize][cart\_value]; // arraysize is same as customeridx and cartvalue store total items in cart

int cartItemPrice[arraysize][cart\_value];

int cart\_idx[arraysize];

int bill[arraysize];

/\*Functions\*/

bool intCheck(string numeric); // checks whether the input value is integer

bool validnumber(string number); // checks whether the input value is 12 digitnumber

bool password\_Check(string Password); // checks whether the input value is valid for password

bool O\_identityCheck(); // Owner's identity Check function

bool nameCheck(string word); // checks whether name is valid

bool nameExists(string name); // whether user exists before

bool checkEmployees(string username, string password);

bool E\_identityCheck(); // checks employees identity

void mainHeader();

void topHeader();

void clearScreen(); // clears screen and calls topheader and mainheader

string mainMenu(); // shows main menu ans returns option entered by user to main

string ownersMenu(); // shows owner's menu and returns option entered by user

void O\_changePassword(); // to change owner's password

void addEmployees(); // allows owner to add employees

void viewEmployees(); // allows owner to view employees

void employeesPay(); // Show's Employees pay

void removeEmployees(); // allows owner to remove employees

string employeesMenu(); // shows employees's menu and returns option entered by user to main

void E\_changePassword(); // to change employee's password

void StockMobiles(); // show's stock of mobiles

void StockHeadphone(); // show's stock of headphones

void StockPowerbank(); // show's stock of powerbank

void StockUSB(); // show's stock of USB's

void StockCables(); // show's stock of Cables

void StockAdopters(); // show's stock of Adopters

void StockCovers(); // show's stock of Covers

void productCategories(); // shows all the categories shop have

void O\_employeesPay(); // shows all employee's pay and attendance to owner

void editStock(); // allows to edit stock details of subcategories

void addStock(); // allows to add stock in defined categories

void removeStock(); // allows to remove stock in defined categories

void updateAttendance(); // allow's employee to update attendance

void restock\_items(); // shows whether there is any need to restock items or not

void O\_feedback(); // shows all the customer's feedback

void C\_feedback(); // shows menu to customer to add feedbank and store's it in arrays accoring to customer's idx

string customersMenu(); // shows employees's menu and returns option entered by user to main

string customerCategoriesMenu(); // shows categories menu to customer and returns category number to view and add items in cart

void searchItems(); // search items according to budget

void customersBudget(); // add budget by customer to view whether he can buy items or not

void billStatus(); // compares price of items and budget of user

void viewCart(); // view customer's cart

void viewBill(); // view total bill to customer

void totalSale(); // show's total sale of the shop to employee

void calculatebill(); // this function calculates bill

void addtoCart(int option); // allows employee to add items in cart

bool C\_identityCheck(); // check's customer identity if he is new or not

void Employeeslist(); // shows employees name

void viewcategories(); // view all the categores

/\*File Handling\*/

string getfield(string line, int field); // returns string to load functions accoriding to demand

void loadData(); // calls all the load functions

void storeData(); // calls all the store functions

void loadusers(); // load user's data(name,password,employee's attendance) from txt files

void storeusers(); // store user's data to txt files

void loadMobiles(); // load mobile's data(namr,stock,price) from txt files

void storeMobiles(); // stores mobile's data(namr,stock,price) to txt files

void loadCovers(); //...............

void storeCovers(); //...............

void loadAdopters(); //...............

void storeAdopters(); //...............

void loadCables(); //...............

void storeCables(); //...............

void loadUSB(); //...............

void storeUSB(); //...............

void loadHeadphones(); //...............

void storeHeadphones(); //...............

void loadPowerbank(); //...............

void storePowerbank(); //...............

int commaCount(string line); // count's commma's to tell us how many words are stored in line

void storeCart(); // store customer's information

void loadcart(); // load customer's information

main()

{

loadData();

while (true)

{

clearScreen();

int option = -1;

string opt = mainMenu();

bool intoption = intCheck(opt); // checks whether the input is int

if (intoption)

{

option = stoi(opt); // changes string to int

}

if (option == 1)

{

clearScreen();

bool check = O\_identityCheck(); // owner's identity Check

if (check == true)

{

while (true)

{

int option = 0;

clearScreen();

string opt = ownersMenu();

bool intoption = intCheck(opt); // checks whether the input is int

if (intoption)

{

option = stoi(opt); // changes string to int

}

if (option == 1)

{

clearScreen();

O\_changePassword(); // change password

}

else if (option == 2)

{

clearScreen();

addEmployees(); // add employees

}

else if (option == 3)

{

clearScreen();

viewEmployees(); // view employees

}

else if (option == 4)

{

clearScreen();

removeEmployees(); // remove employees

}

else if (option == 5)

{

clearScreen();

restock\_items(); // shows items that's stock is less that or equal to zero

}

else if (option == 6)

{

clearScreen();

productCategories(); // total categories (fixed)

}

else if (option == 7)

{

clearScreen();

O\_employeesPay(); // shows pay to the employee

}

else if (option == 8)

{

clearScreen();

O\_feedback(); // view feedback's of customer's in owner's menu

}

else if (option == 9)

{

break; // back to main menu

}

}

}

}

else if (option == 2)

{

clearScreen();

bool check = E\_identityCheck(); // Emplyees identity check

if (check == true)

{

while (true)

{

clearScreen();

int option = -10;

string opt = employeesMenu();

bool intoption = intCheck(opt); // checks input is integer or not

if (intoption)

{

option = stoi(opt); // changes string to int

}

if (option == 1)

{

clearScreen();

E\_changePassword(); // Employee's change password

}

if (option == 2)

{ // SetConsoleTextAttribute(a,)

clearScreen();

productCategories(); // view categories

}

if (option == 3)

{

clearScreen();

editStock(); // edit stock details by entering category name and subcategory name

}

if (option == 4)

{

clearScreen();

addStock(); // add subcategory

}

if (option == 5)

{

clearScreen();

removeStock(); // remove subcatgory

}

if (option == 6)

{

clearScreen();

totalSale(); // total items added by customers to their cart

}

if (option == 7)

{

clearScreen();

cout << "Employee" << employee << endl;

updateAttendance();

}

if (option == 8)

{

clearScreen();

employeesPay(); // pay of that employee whose usernamr and password was entered

}

if (option == 9)

{

break; // back to main menu

}

}

}

}

else if (option == 3)

{

clearScreen();

bool run = C\_identityCheck();

while (run)

{

clearScreen();

int option = -10;

string opt = customersMenu(); // goto customer's menu return's option entered by user in string

bool intoption = intCheck(opt); // checks input is integer or not

if (intoption)

{

option = stoi(opt); // changes string to int

}

if (option == 1)

{

clearScreen();

customersBudget(); // Enter Budget

}

else if (option == 2)

{

clearScreen();

int option;

string opt = customerCategoriesMenu();

bool checkint = intCheck(opt); // checks input is integer or not

if (checkint)

{

option = stoi(opt); // changes string to int

}

addtoCart(option);

calculatebill();

cout << " Press any key to go back... ";

getch();

}

else if (option == 3)

{

clearScreen();

searchItems(); // shows item according to budget(just view not add)

}

else if (option == 4)

{

clearScreen();

viewCart(); // view cart of customer

}

else if (option == 5)

{

clearScreen();

viewBill(); // total bill of customer

}

else if (option == 6)

{

clearScreen();

billStatus(); // total bill exceeds budget or not

}

else if (option == 7)

{

clearScreen();

C\_feedback(); // give feedback

}

else if (option == 8)

{

break; // back to main menu

}

}

}

else if (option == 4)

{

break; // exit programms

}

}

storeData();

}

string getfield(string line, int field) // line read by load function and field is word separated by comma's

{

int commacount = 1;

string item = "";

for (int i = 0; i < line.length(); i++)

{

if (line[i] == ',')

{

commacount++;

}

else if (commacount == field)

{

item = item + line[i];

}

}

return item;

}

void loadData() // load all the data from files when programm starts to the desired arrays

{

loadusers();

loadMobiles();

loadCables();

loadCovers();

loadAdopters();

loadUSB();

loadHeadphones();

loadPowerbank();

loadcart();

}

void storeData() // stores all the data from arrays to files

{

storeusers();

storeMobiles();

storeCovers();

storeCables();

storeAdopters();

storeUSB();

storeHeadphones();

storePowerbank();

storeCart();

}

void loadusers()

{ // load users data

string line;

fstream file;

file.open("users.txt", ios::in);

while (getline(file, line))

{

users[count] = getfield(line, 1);

Passwords[count] = getfield(line, 2);

attendance\_update[count] = stoi(getfield(line, 3));

count++;

}

file.close();

}

void storeusers()

{

fstream file;

int i = 0;

file.open("users.txt", ios::out);

while (i < count)

{

file << users[i] << ",";

file << Passwords[i] << ",";

file << attendance\_update[i] << endl;

i++;

}

file.close();

}

void loadMobiles()

{ // load Mobiles

string line;

fstream file;

file.open("mobiles.txt", ios::in);

while (getline(file, line))

{

categories\_Mobiles[mobiles\_idx] = getfield(line, 1);

stock\_Mobiles[mobiles\_idx] = stoi(getfield(line, 2));

price\_Mobiles[mobiles\_idx] = stoi(getfield(line, 3));

mobiles\_idx++;

}

file.close();

}

void storeMobiles()

{

fstream file;

int i = 0;

file.open("mobiles.txt", ios::out);

while (i < mobiles\_idx)

{

file << categories\_Mobiles[i] << ",";

file << stock\_Mobiles[i] << ",";

file << price\_Mobiles[i] << endl;

i++;

}

file.close();

}

void loadCovers()

{ // load covers

string line;

fstream file;

file.open("covers.txt", ios::in);

while (getline(file, line))

{

categories\_Covers[covers\_idx] = getfield(line, 1);

stock\_Covers[covers\_idx] = stoi(getfield(line, 2));

price\_Covers[covers\_idx] = stoi(getfield(line, 3));

covers\_idx++;

}

file.close();

}

void storeCovers()

{

fstream file;

int i = 0;

file.open("covers.txt", ios::out);

while (i < covers\_idx)

{

file << categories\_Covers[i] << ",";

file << stock\_Covers[i] << ",";

file << price\_Covers[i] << endl;

i++;

}

file.close();

}

void loadAdopters()

{ // load adopters

string line;

fstream file;

file.open("adopters.txt", ios::in);

while (getline(file, line))

{

categories\_Adopters[adopters\_idx] = getfield(line, 1);

stock\_Adopters[adopters\_idx] = stoi(getfield(line, 2));

price\_Adopters[adopters\_idx] = stoi(getfield(line, 3));

adopters\_idx++;

}

file.close();

}

void storeAdopters()

{

fstream file;

int i = 0;

file.open("adopters.txt", ios::out);

while (i < adopters\_idx)

{

file << categories\_Adopters[i] << ",";

file << stock\_Adopters[i] << ",";

file << price\_Adopters[i] << endl;

i++;

}

file.close();

}

void loadCables()

{

string line;

fstream file;

file.open("cables.txt", ios::in);

while (getline(file, line))

{

categories\_Cables[cables\_idx] = getfield(line, 1);

stock\_Cables[cables\_idx] = stoi(getfield(line, 2));

price\_Cables[cables\_idx] = stoi(getfield(line, 3));

cables\_idx++;

}

file.close();

}

void storeCables()

{

fstream file;

int i = 0;

file.open("cables.txt", ios::out);

while (i < cables\_idx)

{

file << categories\_Cables[i] << ",";

file << stock\_Cables[i] << ",";

file << price\_Cables[i] << endl;

i++;

}

file.close();

}

void loadUSB()

{

string line;

fstream file;

file.open("USB.txt", ios::in);

while (getline(file, line))

{

categories\_USB[USB\_idx] = getfield(line, 1);

stock\_USB[USB\_idx] = stoi(getfield(line, 2));

price\_USB[USB\_idx] = stoi(getfield(line, 3));

USB\_idx++;

}

file.close();

}

void storeUSB()

{

fstream file;

int i = 0;

file.open("USB.txt", ios::out);

while (i < USB\_idx)

{

file << categories\_USB[i] << ",";

file << stock\_USB[i] << ",";

file << price\_USB[i] << endl;

i++;

}

file.close();

}

void storeCart()

{

fstream file;

int i = 0;

file.open("cart.txt", ios::out);

while (i < customer\_idx)

{

file << C\_Name[i] << ",";

file << C\_Budget[i] << ",";

file << C\_Number[i] << ",";

file << C\_Comment[i] << ",";

for (int j = 0; j < cart\_idx[i]; j++)

{

file << cartItem[i][j] << ",";

file << cartItemPrice[i][j] << ",";

}

file << endl;

i++;

}

file.close();

}

void loadcart()

{

string line;

fstream file;

file.open("cart.txt", ios::in);

while (getline(file, line))

{

C\_Name[customer\_idx] = getfield(line, 1);

C\_Budget[customer\_idx] = stoi(getfield(line, 2));

C\_Number[customer\_idx] = getfield(line, 3);

C\_Comment[customer\_idx] = getfield(line, 4);

int comma = commaCount(line);

comma = comma - 4;

comma = comma / 2;

int a = 5;

int b = 6;

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

{

cartItem[customer\_idx][cart\_idx[customer\_idx]] = getfield(line, a);

cartItemPrice[customer\_idx][cart\_idx[customer\_idx]] = stoi(getfield(line, b));

a = a + 2;

b = b + 2;

cart\_idx[customer\_idx]++;

}

customer\_idx++;

}

file.close();

}

void loadHeadphones()

{

string line;

fstream file;

file.open("headphones.txt", ios::in);

while (getline(file, line))

{

categories\_Headphones[headphone\_idx] = getfield(line, 1);

stock\_Headphones[headphone\_idx] = stoi(getfield(line, 2));

price\_Headphones[headphone\_idx] = stoi(getfield(line, 3));

headphone\_idx++;

}

file.close();

}

void storeHeadphones()

{

fstream file;

int i = 0;

file.open("headphones.txt", ios::out);

while (i < headphone\_idx)

{

file << categories\_Headphones[i] << ",";

file << stock\_Headphones[i] << ",";

file << price\_Headphones[i] << endl;

i++;

}

file.close();

}

void loadPowerbank()

{

string line;

fstream file;

file.open("powerbank.txt", ios::in);

while (getline(file, line))

{

categories\_Powerbank[powerbank\_idx] = getfield(line, 1);

stock\_Powerbank[powerbank\_idx] = stoi(getfield(line, 2));

price\_Powerbank[powerbank\_idx] = stoi(getfield(line, 3));

powerbank\_idx++;

}

file.close();

}

void storePowerbank()

{

fstream file;

int i = 0;

file.open("powerbank.txt", ios::out);

while (i < powerbank\_idx)

{

file << categories\_Powerbank[i] << ",";

file << stock\_Powerbank[i] << ",";

file << price\_Powerbank[i] << endl;

i++;

}

file.close();

}

void clearScreen()

{

system("cls");

cout << "\33[34m";

mainHeader();

topHeader();

Sleep(90);

}

string mainMenu()

{

cout << "\t\t\t\t\t\tMAIN MENU " << endl

<< endl;

string choice;

cout << " Choose one of the following : " << endl;

cout << endl;

cout << " 1. Owner " << endl;

cout << " 2. Employee" << endl;

cout << " 3. Customer " << endl;

cout << " 4. Exit " << endl;

cout << endl;

cout << " Enter your choice : ";

cin >> choice;

return choice;

}

string ownersMenu()

{

Sleep(90);

cout << "\t\t\t\t\tOWNER'S MENU " << endl;

string choice;

cout << "Choose one of the following : " << endl;

cout << endl;

cout << " 1. Change Password " << endl;

cout << " 2. Add Employees" << endl;

cout << " 3. View Employees" << endl;

cout << " 4. Remove Employees" << endl;

cout << " 5. Restock Items" << endl;

cout << " 6. Stock details " << endl;

cout << " 7. Pay of Employees" << endl;

cout << " 8. View Feedbacks" << endl;

cout << " 9. Logout" << endl;

cout << endl;

cout << " Enter your choice : ";

cin >> choice;

return choice;

}

bool O\_identityCheck() // checks owners identity

{

string username;

string Password;

cout << "\t\t\t\t\tPROVE YOUR IDENTITY " << endl

<< endl;

cout << " Enter Username : ";

cin >> username;

cout << " Enter Password : ";

cin >> Password;

cout << endl;

if (username == users[0] && Password == Passwords[0])

{

cout << " Access Granted!!!" << endl;

cout << endl;

cout << " Press any key to continue!!!" << endl;

getch();

return true;

}

cout << " Incorrect Username or Password !!!" << endl;

cout << endl;

cout << " Press any key to continue!!!" << endl;

getch();

return false;

}

void O\_changePassword() // change owner password

{

cout << "\t\t\t\t\tCHANGE PASSWORD " << endl

<< endl;

string Password;

cout << " Enter your Password : ";

cin >> Password;

if (Password == Passwords[0])

{

cout << " \n Password Matched!!!" << endl;

cout << " \n\n Enter New Password : ";

cin >> Password;

bool valid = password\_Check(Password);

if (valid)

{

Passwords[0] = Password;

cout << "\n Password Changed Successfully!!!" << endl;

}

}

else

{

cout << " Incorrect Password!!!" << endl;

}

cout << " \nPress any Key to go back!!!";

getch();

}

bool password\_Check(string Password)

{

int i = 0;

if (Password.length() > 7)

{

for (int idx = 0; idx < count; idx++)

{

if (Password == Passwords[idx])

{

cout << "\n Please Choose a strong Password...!!!\n ";

return false;

}

}

while (i < Password.length())

{

if ((Password[i] > 63 && Password[i] < 91) || (Password[i] > 96 && Password[i] < 123) || (Password[i] > 47 && Password[i] < 58))

{

i++;

}

else

{

cout << "\n Invalid Input!!!\n Password only consists of numeric values and Alphabets...\n";

return false;

}

}

}

else

{

cout << " \n Invalid Input!!!\n Password must be consist of atleast 8 characters.\n ";

return false;

}

return true;

}

void addEmployees()

{

string password; // temporary variable

string name; // temporary variable

cout << "\t\t\t\t\tADD EMPLOYEE " << endl

<< endl;

cout << " Enter the username for Employee : ";

cin >> name; // temporary variable

bool exist = nameExists(name);

if (exist == false)

{

bool validname = nameCheck(name);

if (validname == true)

{

cout << " Enter Password for Employee : ";

cin >> password;

bool check = password\_Check(password);

if (check == true)

{

users[count] = name;

Passwords[count] = password;

attendance\_update[count] = 0;

count++;

cout << " New Employee added Successfully!!!" << endl;

cout << endl;

}

else

{

cout << " Invalid Password " << endl;

}

}

}

else

{

cout << "\n User ALready Exists" << endl;

}

cout << " \npress any key to continue!!! ";

getch();

}

void Employeeslist()

{

cout << " Employee's list is as follow : " << endl;

for (int idx = 1; idx < count; idx++)

{

cout << " Employee " << idx << " : " << users[idx] << endl;

}

}

void viewEmployees()

{

cout << "\t\t\t\t\t\tEMPLOYEE'S LIST " << endl

<< endl;

if (count > 1)

{

cout << " Total Employees : " << count - 1 << endl;

cout << endl;

Employeeslist();

}

else

{

cout << "\n\n No Employees Added yet!!!\n";

}

cout << endl;

cout << " Press any key to continue....";

getch();

}

bool nameExists(string name)

{

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

{

if (name == users[i])

{

return true;

}

}

return false;

}

bool nameCheck(string word)

{

int i = 0;

while (i < word.length())

{

if ((word[i] > 64 && word[i] < 91) || (word[i] > 96 && word[i] < 123))

{

i++;

}

else

{

cout << "\n Invalid Input!!!";

cout << "\n\n Name only contains aphabets..!!!\n ";

return false;

}

}

return true;

}

void removeEmployees()

{

int idx = -1;

cout << "\t\t\t\tREMOVE EMPLOYEES" << endl

<< endl;

if (count > 1)

{

Employeeslist();

string name;

cout << "\n Enter Name of Employee you want to remove : ";

cin >> name;

for (int i = 1; i < count; i++)

{

if (name == users[i])

{

idx = i;

}

}

if (idx != -1)

{

count--;

users[idx] = users[count]; // replace last user with that user you want to remove

Passwords[idx] = Passwords[count];

cout << " Employee removed Successfully!!!";

}

else

{

cout << "User not found!!!" << endl;

}

}

cout << "\nPress any key to continue...";

getch();

}

bool E\_identityCheck() // E symbolizes Employee

{

bool flag;

string username;

string Password;

cout << "\t\t\t\t\tPROVE YOUR IDENTITY " << endl

<< endl;

cout << " Enter Username : ";

cin >> username;

cout << " Enter Password : ";

cin >> Password;

cout << endl;

bool Check = checkEmployees(username, Password);

if (Check == true)

{

flag = true;

cout << " Access Granted!!!" << endl;

cout << endl;

}

else

{

flag = false;

cout << " Incorrect Password !!!"

<< endl;

}

Sleep(90);

cout << " Press any key to continue!!!";

getch();

return flag;

}

void E\_changePassword()

{

bool flag = false;

string username;

string Password;

cout << " Enter your username : ";

cin >> username;

cout << " Enter your Password : ";

cin >> Password;

bool check = checkEmployees(username, Password);

if (check == true)

{

cout << " Password Matched!!!" << endl;

cout << " Enter New Password : ";

cin >> Password;

bool valid = password\_Check(Password);

if (valid)

{

cout << " Password Changed Successfully!!!" << endl;

}

else if (!valid)

{

cout << " Invalid Input!!!" << endl;

}

}

else

{

cout << " Incorrect Username or Password!!!" << endl;

}

cout << " Press any key to continue!!!";

getch();

}

bool checkEmployees(string username, string password)

{

bool flag = false;

for (int idx = 1; idx < count; idx++)

{

if (username == users[idx] && password == Passwords[idx])

{

employee = idx;

attendanceonlyonce = employee;

flag = true;

}

}

return flag;

}

string employeesMenu()

{

string choice;

cout << "\t\t\t\tEMPLOYEES'S MENU " << endl;

cout << "Choose one of the following : " << endl;

cout << endl;

cout << " 1. Change Password " << endl;

cout << " 2. View Categories " << endl;

cout << " 3. Edit Stock " << endl;

cout << " 4. Add Subcategory " << endl;

cout << " 5. Remove Stock " << endl;

cout << " 6. View Daily Sale " << endl;

cout << " 7. Update Attendance " << endl;

cout << " 8. Pay " << endl;

cout << " 9. Logout " << endl;

cout << endl;

cout << " Enter your choice : ";

cin >> choice;

return choice;

}

void viewcategories()

{

for (int idx = 0; idx < categories\_idx; idx++)

{

cout << idx + 1 << ". " << categories[idx] << endl;

}

}

void productCategories() // cout all the categories

{

a:

clearScreen();

cout << "\t\t\t\t\t PRODUCT CATEGORIES MENU " << endl;

cout << endl;

viewcategories();

cout << "8. Exit" << endl;

string option;

cout << " Enter serial number if you want to view items in that category : ";

cin >> option;

if (option[0] > 48 && option[0] < 56)

{

if (option == "1")

{

clearScreen();

StockMobiles();

}

else if (option == "2")

{

clearScreen();

StockCovers();

}

else if (option == "3")

{

clearScreen();

StockAdopters();

}

else if (option == "4")

{

clearScreen();

StockCables();

}

else if (option == "5")

{

clearScreen();

StockUSB();

}

else if (option == "6")

{

clearScreen();

StockHeadphone();

}

else if (option == "7")

{

clearScreen();

StockPowerbank();

}

string option;

cout << endl;

cout << " Press 1 to view Categories again or any other key to go to main Menu : ";

cin >> option;

if (option == "1")

{

goto a;

}

}

}

void StockMobiles() // cout all covers

{

cout << "\t\t\t\t\t\tMOBILE PHONES " << endl

<< endl;

cout << "ITEM\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < mobiles\_idx; j++)

{

cout << categories\_Mobiles[j] << "\t\t " << stock\_Mobiles[j] << "\t\t" << price\_Mobiles[j] << "Rs." << endl;

}

}

void StockCovers()

{

cout << "\t\t\t\t\t\tMOBILE COVERS" << endl

<< endl;

cout << "ITEM\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < covers\_idx; j++)

{

cout << categories\_Covers[j] << "\t\t" << stock\_Covers[j] << "\t\t" << price\_Covers[j] << "Rs." << endl;

}

}

void StockAdopters()

{

cout << "\t\t\t\t\t\tADOPTERS " << endl

<< endl;

cout << "ITEM\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < adopters\_idx; j++)

{

cout << categories\_Adopters[j] << "\t\t" << stock\_Adopters[j] << "\t\t" << price\_Adopters[j] << "Rs." << endl;

}

}

void StockCables()

{

cout << "\t\t\t\t\t\tDATA CABLES " << endl

<< endl;

cout << "ITEM\t\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < cables\_idx; j++)

{

cout << categories\_Cables[j] << "\t\t" << stock\_Cables[j] << "\t\t" << price\_Cables[j] << "Rs." << endl;

}

}

void StockUSB()

{

cout << "\t\t\t\t\t\tUSB'S " << endl

<< endl;

cout << "ITEM\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < USB\_idx; j++)

{

cout << categories\_USB[j] << "\t\t" << stock\_USB[j] << "\t\t" << price\_USB[j] << "Rs." << endl;

}

}

void StockHeadphone()

{

cout << "\t\t\t\t\t\tHEADPHONES" << endl

<< endl;

cout << "ITEM\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < headphone\_idx; j++)

{

cout << categories\_Headphones[j] << "\t\t" << stock\_Headphones[j] << "\t\t" << price\_Headphones[j] << "Rs." << endl;

}

}

void StockPowerbank()

{

cout << "\t\t\t\t\t\tPOWERBANK" << endl;

cout << "ITEM\t\t\tSTOCK\t\tPRICE" << endl;

for (int j = 0; j < powerbank\_idx; j++)

{

cout << categories\_Powerbank[j] << "\t\t" << stock\_Powerbank[j] << "\t\t" << price\_Powerbank[j] << "Rs." << endl;

}

}

void editStock()

{

int i = -10;

int j;

string category;

string item;

cout << " Enter category of item you want to change the stock of : ";

cin >> category;

for (int idx = 0; idx < categories\_idx; idx++)

{

if (category == categories[idx])

{

i = idx;

}

}

if (i != -10)

{

cout << " Enter name of Item ";

cin >> item;

if (i == 0)

{

for (int idx = 0; idx < mobiles\_idx; idx++)

{

if (item == categories\_Mobiles[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Mobiles[j] = stoi(stock);

price\_Mobiles[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 1)

{

for (int idx = 0; idx < covers\_idx; idx++)

{

if (item == categories\_Covers[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Covers[j] = stoi(stock);

price\_Covers[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 2)

{

for (int idx = 0; idx < adopters\_idx; idx++)

{

if (item == categories\_Adopters[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Adopters[j] = stoi(stock);

price\_Adopters[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 3)

{

for (int idx = 0; idx < cables\_idx; idx++)

{

if (item == categories\_Cables[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Cables[j] = stoi(stock);

price\_Cables[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 4)

{

for (int idx = 0; idx < USB\_idx; idx++)

{

if (item == categories\_USB[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_USB[j] = stoi(stock);

price\_USB[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 5)

{

for (int idx = 0; idx < headphone\_idx; idx++)

{

if (item == categories\_Headphones[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Headphones[j] = stoi(stock);

price\_Headphones[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

else if (i == 6)

{

for (int idx = 0; idx < powerbank\_idx; idx++)

{

if (item == categories\_Powerbank[idx])

{

j = idx;

break;

}

}

string stock;

string price;

cout << " Enter New Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter new Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Powerbank[j] = stoi(stock);

price\_Powerbank[j] = stoi(price);

cout << " Updated Successfully";

}

}

}

}

cout << endl;

cout << "Enter any key to go back...";

getch();

}

void addStock()

{

int i = -10;

string category;

string item;

viewcategories(); // cout all the categories

cout << " \n Enter category of item you want to add the stock of : ";

cin >> category;

for (int idx = 0; idx < categories\_idx; idx++)

{

if (category == categories[idx])

{

i = idx;

}

}

if (i != -10)

{

if (i == 0)

{

StockMobiles();

cout << " Enter Name of New Item : ";

cin >> categories\_Mobiles[mobiles\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Mobiles[mobiles\_idx] = stoi(stock);

price\_Mobiles[mobiles\_idx] = stoi(price);

mobiles\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 1)

{

StockCovers();

cout << " Enter Name of New Item : ";

cin >> categories\_Covers[covers\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Covers[covers\_idx] = stoi(stock);

price\_Covers[covers\_idx] = stoi(price);

covers\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 2)

{

StockAdopters();

cout << " Enter Name of New Item : ";

cin >> categories\_Adopters[adopters\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Adopters[adopters\_idx] = stoi(stock);

price\_Adopters[adopters\_idx] = stoi(price);

adopters\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 3)

{

StockCables();

cout << " Enter Name of New Item : ";

cin >> categories\_Cables[cables\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Cables[cables\_idx] = stoi(stock);

price\_Cables[cables\_idx] = stoi(price);

cables\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 4)

{

StockUSB();

cout << " Enter Name of New Item : ";

cin >> categories\_USB[USB\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_USB[USB\_idx] = stoi(stock);

price\_USB[USB\_idx] = stoi(price);

USB\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 5)

{

StockHeadphone();

cout << " Enter Name : ";

cin >> categories\_Headphones[headphone\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Headphones[headphone\_idx] = stoi(stock);

price\_Headphones[headphone\_idx] = stoi(price);

headphone\_idx++;

cout << " Added Successfully";

}

}

}

else if (i == 6)

{

cout << " Enter Name : ";

cin >> categories\_Powerbank[powerbank\_idx];

string stock;

string price;

cout << " Enter Stock : ";

cin >> stock;

bool intstock = intCheck(stock);

if (intstock)

{

cout << " Enter Price : ";

cin >> price;

bool intprice = intCheck(price);

if (intprice)

{

stock\_Powerbank[powerbank\_idx] = stoi(stock);

price\_Powerbank[powerbank\_idx] = stoi(price);

powerbank\_idx++;

cout << " Added Successfully";

}

}

}

}

else

{

cout << " invalid Input " << endl;

}

cout << endl;

cout << " Press any key to continue!!!";

getch();

}

void removeStock()

{

int check = -10; // to save index

int change = 0; // whether the item removed or not

string category;

string item; //

viewcategories();

cout << " Enter category of item you want to Remove the stock of : ";

cin >> category;

for (int idx = 0; idx < categories\_idx; idx++)

{

if (category == categories[idx])

{

check = idx;

}

}

if (check != -10)

{

string name;

if (check == 0)

{

StockMobiles();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < mobiles\_idx; j++)

{

if (name == categories\_Mobiles[j])

{

mobiles\_idx--;

change++;

categories\_Mobiles[j] = categories\_Mobiles[mobiles\_idx];

stock\_Mobiles[j] = stock\_Mobiles[mobiles\_idx];

price\_Mobiles[j] = price\_Mobiles[mobiles\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 1)

{

StockCovers();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < covers\_idx; j++)

{

if (name == categories\_Covers[j])

{

covers\_idx--;

change++;

categories\_Covers[j] = categories\_Covers[mobiles\_idx];

stock\_Covers[j] = stock\_Covers[mobiles\_idx];

price\_Covers[j] = price\_Covers[mobiles\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 2)

{

StockAdopters();

cout << " \nEnter Name : ";

cin >> name;

for (int j = 0; j < adopters\_idx; j++)

{

if (name == categories\_Adopters[j])

{

adopters\_idx--;

change++;

categories\_Adopters[j] = categories\_Adopters[adopters\_idx];

stock\_Adopters[j] = stock\_Adopters[adopters\_idx];

price\_Adopters[j] = price\_Adopters[adopters\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 3)

{

StockCables();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < cables\_idx; j++)

{

if (name == categories\_Cables[j])

{

cables\_idx--;

change++;

categories\_Cables[j] = categories\_Cables[cables\_idx];

stock\_Cables[j] = stock\_Cables[cables\_idx];

price\_Cables[j] = price\_Cables[cables\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 4)

{

StockUSB();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < USB\_idx; j++)

{

if (name == categories\_USB[j])

{

USB\_idx--;

change++;

categories\_USB[j] = categories\_USB[cables\_idx];

stock\_USB[j] = stock\_USB[cables\_idx];

price\_USB[j] = price\_USB[cables\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 5)

{

StockHeadphone();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < headphone\_idx; j++)

{

if (name == categories\_Headphones[j])

{

headphone\_idx--;

change++;

categories\_Headphones[j] = categories\_Headphones[headphone\_idx];

stock\_Headphones[j] = stock\_Headphones[headphone\_idx];

price\_Headphones[j] = price\_Headphones[headphone\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

else if (check == 6)

{

StockPowerbank();

cout << "\n Enter Name : ";

cin >> name;

for (int j = 0; j < powerbank\_idx; j++)

{

if (name == categories\_Powerbank[j])

{

powerbank\_idx--;

change++;

categories\_Powerbank[j] = categories\_Powerbank[powerbank\_idx];

stock\_Powerbank[j] = stock\_Powerbank[powerbank\_idx];

price\_Powerbank[j] = price\_Powerbank[powerbank\_idx];

cout << "\n Removed Successfully!!!\n";

break;

}

}

}

if (change == 0)

{

cout << " \n Update Failed!!!\n You must enter Valid Name!!!!\n";

}

}

else

{

cout << " Invaid Input!!!" << endl;

}

cout << " Press any key to continue...";

getch();

}

void updateAttendance()

{

if (attendanceonlyonce == employee) // employee is employee idx that is stored when he signin

{

cout << "Press 'p' to mark attendance for today ";

cin >> attendance;

if (attendance == "p" || attendance == "P")

{

cout << " Updated!!!";

attendanceonlyonce++;

attendance\_update[employee]++;

}

else

{

cout << " Update Failed!!!";

}

}

else

{

cout << " \n You already marked your attendance!!!\n ";

}

cout << "\n Press any key to go back...";

getch();

}

void employeesPay()

{

cout << " Total Days employee worked : " << attendance\_update[employee] << " Days " << endl;

cout << " Total Pay : " << attendance\_update[employee] \* 2000 << endl;

cout << endl

<< " Press any Key to Continue...\n";

getch();

}

void O\_employeesPay()

{

for (int i = 1; i < count; i++)

{

cout << i;

cout << ".\n Employees's Name : " << users[i] << endl;

cout << " Total Days employee worked : " << attendance\_update[i] << " Days " << endl;

cout << " Total Pay : " << attendance\_update[i] \* 2000 << endl;

cout << endl;

getch();

}

cout << " Press any Key to go back..";

getch();

}

void restock\_items()

{

cout << "\t\t\t\t\tRESTOCK ITEMS" << endl

<< endl;

int count = 0;

for (int idx = 0; idx < categories\_idx; idx++)

{

if (idx == 0)

{

for (int i = 0; i < mobiles\_idx; i++)

{

if (stock\_Mobiles[i] == 0)

{

cout << categories\_Mobiles[i] << endl;

count++;

}

}

}

if (idx == 1)

{

for (int i = 0; i < covers\_idx; i++)

{

if (stock\_Covers[i] <= 0)

{

cout << categories\_Covers[i] << endl;

count++;

}

}

}

if (idx == 2)

{

for (int i = 0; i < adopters\_idx; i++)

{

if (stock\_Adopters[i] <= 0)

{

cout << categories\_Adopters[i] << endl;

count++;

}

}

}

if (idx == 3)

{

for (int i = 0; i < cables\_idx; i++)

{

if (stock\_Cables[i] <= 0)

{

cout << categories\_Cables[i] << endl;

count++;

}

}

}

if (idx == 4)

{

for (int i = 0; i < USB\_idx; i++)

{

if (stock\_USB[i] <= 0)

{

cout << categories\_USB[i] << endl;

count++;

}

}

}

if (idx == 5)

{

for (int i = 0; i < headphone\_idx; i++)

{

if (stock\_Headphones[i] <= 0)

{

cout << categories\_Headphones[i] << endl;

count++;

}

}

}

if (idx == 6)

{

for (int i = 0; i < powerbank\_idx; i++)

{

if (stock\_Powerbank[i] <= 0)

{

cout << categories\_Powerbank[i] << endl;

count++;

}

}

}

}

if (count == 0)

{

cout << " No need to restock any item..." << endl

<< endl;

}

cout << " Press any key to go back!!!";

getch();

}

bool validnumber(string number)

{

int length = number.length();

if (length == 11)

{

for (int i = 0; i < number.length(); i++)

{

if (!(number[i] > 47 && number[i] < 58))

{

cout << "\n Invalid Input!!!!\n Entered value is not numeric...;)\n ";

getch();

return false;

}

}

}

else

{

cout << " \n Number only consist of 11 characters and numeric values.\n Please enter a valid number!!!\n";

return false;

}

return true;

}

string customersMenu()

{

cout << "\t\t\t\t\tWELCOME TO THE SHOP " << endl;

string choice;

cout << " \n Choose one of the following : " << endl;

cout << endl;

cout << " 1. Enter Budget " << endl;

cout << " 2. Add items in Cart " << endl;

cout << " 3. Search Items according to Budget " << endl;

cout << " 4. View cart " << endl;

cout << " 5. Caculate Bill " << endl;

cout << " 6. View Wallet" << endl;

cout << " 7. Give Feedback " << endl;

cout << " 8. Exit " << endl;

cout << endl;

cout << " Enter your choice : ";

cin >> choice;

return choice;

}

void C\_feedback()

{

string number;

string option;

cout << " \t\t\t\t\t\tGIVE YOUR FEEDBACK";

if (C\_Number[customer] == "1")

{

cin.ignore();

cout << "\n\n Enter Your Phone Number : ";

getline(cin, number);

bool numbercheck = validnumber(number);

cout << numbercheck;

if (numbercheck)

{

cout << "fgh";

C\_Number[customer] = number;

cout << " Give Feedback : ";

getline(cin, C\_Comment[customer]);

cout << endl;

cout << " Response added successfully!!!" << endl;

cout << endl;

}

}

else

{

cout << "\n You have added your response before...";

}

cout << "\n Press any key to go back...\n";

getch();

}

void O\_feedback()

{

int count = 0;

cout << "\t\t\t\t\t\tFEEDBACKS\n\n";

for (int idx = 0; idx < customer\_idx; idx++)

{

if (C\_Number[idx] == "1")

{

continue;

}

cout << " " << C\_Name[idx] << "(" << C\_Number[idx] << ") : " << endl;

cout << " " << C\_Comment[idx] << endl;

cout << endl;

getch();

count++;

}

if (count == 0)

{

cout << "\n\n There are no Feedbacks!!!...";

}

cout << endl;

cout << "\n\n Press any key to go back...";

getch();

}

string customerCategoriesMenu()

{

cout << "t\t\t\t\tCategories" << endl;

string choice;

for (int idx = 0; idx < categories\_idx; idx++)

{

cout << idx + 1 << ". " << categories[idx] << endl;

}

cout << " Enter your Choice :";

cin >> choice;

return choice;

}

void searchItems() //

{

cout << "\t\t\tItems According to Budget" << endl;

int count = 1;

cout << "\nMobile Phones" << endl;

for (int idx = 0; idx < mobiles\_idx; idx++)

{

if (price\_Mobiles[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_Mobiles[idx] << "\t" << price\_Mobiles[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Mobile Phones...";

cout << "\n Press any key to continue...";

getch();

cout << "\nMobile Covers" << endl;

for (int idx = 0; idx < covers\_idx; idx++)

{

if (price\_Covers[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_Covers[idx] << "\t" << price\_Covers[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Mobile Covers...";

cout << " \n Press any key to continue...";

getch();

cout << "\nAdopters" << endl;

for (int idx = 0; idx < adopters\_idx; idx++)

{

if (price\_Adopters[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_Adopters[idx] << "\t" << price\_Adopters[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Adopters...";

cout << " \n Press any key to continue...";

getch();

cout << "\n Data Cables" << endl;

for (int idx = 0; idx < cables\_idx; idx++)

{

if (price\_Cables[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_Cables[idx] << "\t" << price\_Cables[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Adopters...";

cout << " \n Press any key to continue...";

getch();

cout << "\n USB's" << endl;

for (int idx = 0; idx < USB\_idx; idx++)

{

if (price\_USB[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_USB[idx] << "\t" << price\_USB[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for USB's...";

cout << " \n Press any key to continue...";

getch();

cout << "\n Headphones's" << endl;

for (int idx = 0; idx < headphone\_idx; idx++)

{

if (price\_Headphones[idx] < C\_Budget[customer])

{

cout<<count<< .\t" << categories\_Headphones[idx] << "\t" << price\_Headphones[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Headphones...";

cout << " \n Press any key to continue...";

getch();

cout << "\n Powerbank" << endl;

for (int idx = 0; idx < powerbank\_idx; idx++)

{

if (price\_Powerbank[idx] < C\_Budget[customer])

{

cout << count << ".\t" << categories\_Powerbank[idx] << "\t" << price\_Powerbank[idx] << endl;

count++;

}

}

cout << " You are all Caught Up for Powerbank...";

cout << " \n Press any key to go back....";

getch();

}

void customersBudget()

{

string bud;

cout << "\t\t\t\t\t\t BUDGET MENU" << endl

<< endl;

cout << " Enter money you have in Rupees : ";

cin >> bud; // budget

bool budgetint = intCheck(bud);

if (budgetint)

{

int budget = stoi(bud);

C\_Budget[customer] = budget;

cout << endl;

cout << " UPDATED SUCCESSFULLY!!!" << endl;

cout << endl;

}

cout << " \n Press any key to continue...";

getch();

}

void addtoCart(int option)

{

if (option == 1)

{

int cart = 0;

string input; // temporary variable to add item in cart

cout << " \t\t\t\t\t\tMOBILES " << endl;

cout << " Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < mobiles\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Mobiles[idx] << "\t\t" << price\_Mobiles[idx] << endl;

}

cout << " Enter Option : ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < mobiles\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Mobiles[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Mobiles[cart];

cart\_idx[customer]++;

stock\_Mobiles[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

cout << cartItem[customer][0];

getch();

}

}

else if (option == 2)

{

string input;

int cart; // temporary variable to add item in cart

cout << "\t\t\t\t\t\tMOBILE COVERS" << endl;

cout << "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < covers\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Covers[idx] << "\t\t" << price\_Covers[idx] << endl;

}

cout << " Enter Option : ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < covers\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Covers[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Covers[cart];

cart\_idx[customer]++;

stock\_Covers[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else if (option == 3)

{

string input;

int cart; // temporary variable to add item in cart

cout << " ADOPTER" << endl;

cout << "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < adopters\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Adopters[idx] << "\t\t" << price\_Adopters[idx] << endl;

}

cout << " Enter Option : ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < adopters\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Adopters[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Adopters[cart];

cart\_idx[customer]++;

stock\_Adopters[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else if (option == 4)

{

string input;

int cart; // temporary variable to add item in cart

cout << " DATA CABLES" << endl;

cout

<< "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < cables\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Cables[idx] << "\t\t" << price\_Cables[idx] << endl;

}

cout << " Enter Option : ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < cables\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Cables[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Cables[cart];

cart\_idx[customer]++;

stock\_Cables[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else if (option == 5)

{

string input;

int cart; // temporary variable to add item in cart

cout << " USB's" << endl;

cout

<< "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < USB\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_USB[idx] << "\t\t" << price\_USB[idx] << endl;

}

cout << " Enter Option to add in your cart: ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < adopters\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_USB[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_USB[cart];

cart\_idx[customer]++;

stock\_USB[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else if (option == 6)

{

string input;

int cart; // temporary variable to add item in cart

cout << " HEADPHONES " << endl;

cout

<< "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < headphone\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Headphones[idx] << "\t\t" << price\_Headphones[idx] << endl;

}

cout << " Enter Option to add in your cart: ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < headphone\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Headphones[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Headphones[cart];

cart\_idx[customer]++;

stock\_Headphones[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else if (option == 7)

{

string input;

int cart; // temporary variable to add item in cart

cout << " HEADPHONES " << endl;

cout

<< "Option\t\tName\t\tPrice" << endl;

for (int idx = 0; idx < powerbank\_idx; idx++)

{

cout << idx + 1 << "\t\t" << categories\_Powerbank[idx] << "\t\t" << price\_Powerbank[idx] << endl;

}

cout << " Enter Option to add in your cart: ";

cin >> input;

bool cartint = intCheck(input);

if (cartint)

{

cart = stoi(input);

if (cart < powerbank\_idx + 1)

{

cart = cart - 1;

cartItem[customer][cart\_idx[customer]] = categories\_Powerbank[cart];

cartItemPrice[customer][cart\_idx[customer]] = price\_Powerbank[cart];

cart\_idx[customer]++;

stock\_Powerbank[cart]--;

}

else

{

cout << " Invalid Input!!!" << endl;

}

}

}

else

{

cout << " Invalid Input!!!";

}

}

void viewCart()

{

cout << "\t\t\t\t\t\tCART" << endl;

if (cart\_idx[customer] > 0)

{

cout << "Serial No."

<< "\t\t Item"

<< "\t\t Price" << endl;

for (int idx = 0; idx < cart\_idx[customer]; idx++)

{

cout << idx + 1 << ".\t\t" << cartItem[customer][idx] << "\t\t" << cartItemPrice[customer][idx] << endl;

}

}

else

{

cout << " Cart is Empty" << endl;

}

getch();

}

void viewBill()

{

calculatebill();

if (cart\_idx[customer] > 0)

{

cout

<< " Total Bill is " << bill[customer] << "Rs. ";

getch();

}

else

{

cout << "EMPTY CART" << endl;

}

}

void calculatebill()

{

bill[customer] = 0;

for (int idx = 0; idx < cart\_idx[customer]; idx++)

{

bill[customer] = bill[customer] + cartItemPrice[customer][idx];

}

}

void billStatus()

{

calculatebill();

if (bill[customer] < C\_Budget[customer])

{

cout << C\_Budget[customer] - bill[customer] << "Rs. Left in your Wallet. " << endl;

cout << " You Still have capacity to buy items. ";

}

else if (bill[customer] == C\_Budget[customer])

{

cout << " 0 Rs. Left in your Wallet.\n You don't have capacity to buy items." << endl;

}

else if (bill[customer] > C\_Budget[customer])

{

cout << bill[customer] - C\_Budget[customer] << "Rs. Needed to Buy items." << endl;

cout << " You need to remove some items from your cart. " << endl;

}

cout << endl;

cout << " Enter any key to go back...";

getch();

}

bool C\_identityCheck()

{

bool flag = false;

int check = 0;

string name;

cout << " Enter your name : ";

getline(cin.ignore(), name);

bool valname = nameCheck(name); // name valid

if (valname)

{

for (int i = 0; i < customer\_idx; i++)

{

if (name == C\_Name[i])

{

customer = i;

cout << "\n WELCOME BACK!!!" << endl;

check = 1;

flag = true;

break;

}

}

if (check == 0)

{

cout << " Welcome to the shop!!!\n Hope you gonna like it... !!!\n";

C\_Name[customer\_idx] = name;

C\_Budget[customer\_idx] = 0;

C\_Number[customer\_idx] = "1";

C\_Comment[customer\_idx] = "1";

customer = customer\_idx;

customer\_idx++;

flag = true;

}

}

cout << " \n Press any key to continue...";

getch();

return flag;

}

int commaCount(string line)

{

int count = 0;

for (int i = 0; i < line.length(); i++)

{

if (line[i] == ',')

{

count++;

}

}

return count;

}

bool intCheck(string numeric)

{

for (int i = 0; i < numeric.length(); i++)

{

if (!(numeric[i] > 47 && numeric[i] < 58))

{

cout << "\n Invalid Input!!!!\n Entered value is not numeric...;)\n ";

getch();

return false;

}

}

return true;

}

void totalSale()

{

cout << "\t\t\t\t\t\tSALE" << endl;

for (int i = 0; i < customer\_idx; i++)

{

if (cart\_idx[i] > 0)

{

cout << "NAME : " << C\_Name[i] << endl;

for (int j = 0; j < cart\_idx[i]; j++)

{

cout << cartItem[i][j] << "\t\t" << cartItemPrice[i][j] << endl;

}

getch();

}

}

}

# Weakness in Business Application

1. No Sorting in the System.
2. No proper salary system for Employee.
3. No validation of spaces.
4. Customer’s menu need modifications.
5. It is not that presentable.

# Future Directions

1. Owner will be able to give bonus and deduct salary according to employee’s attendance and Proper salary system.
2. Add Recommendations according to most sold items.
3. Employee will be able to add new Categories
4. Proper Cart system for Customer.
5. Proper billing System for Customer.
6. Proper Attendance System.