General Store

Session 2023 - 2027

Submitted by:

Muhammad Talha 2023-CS-169

Supervised by:

Mam Maida Mirza

Course:

CSC-102 Programming Fundamentals

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Short Description of General Store

- General Store software for stock management, sales tracking, and customer transaction for small businesses. An efficient way for the customer and the owner of the shop to keep record, buy items in an order.

Users of Application (Customer & Manager)

General Store includes the following users:

- Manager: He/she is basically the owner of the store, who sells projects, manages and set the prices of items, keep sales record and track the activity of user.
- Customer : He/she is the user of Departmental Store, who can buy, check rate lists and also can review the store.

Functional Requirements:

User Story ID	***As a***	***I want to perform***	***So that I can***
1)	**Manager**	Update Rates	Update tax rate and items rate
		Edit Stock	Add and remove any item in the stock
		Show all Users	To check all users and their roles.
		Remove user	To delete a user other than a manager
		Check Listed Items	Remove, add and update the listed items along with their prices.
		Check Revenue	See the total revenue of all sales
		Print Sales Record	To see list all items sales
		Change Admin Code	To change the special admin code.
		See Complains	To see Complains submitted by the users.
		App Settings	To change theme and password
		See Ratings & Reviews	To check reviews / feedback of users
2)	**User**	See Listed Items	So that user can see prices and items.
		Buy items	Buy items and add them to cart.

	Check my cart	Check the items bought and total bill.
	Empty Cart	Resets my cart
	Pay Bill	Pay the total bill of all items bought.
	App Settings	Change Theme and Password
	Customer Support Menu	To navigate on Customer Support Page
	Write Complains	Write complains to manager
	Contact Page	To see phone number and email of General Store
	Write a Review	Review and rate the service
	Check Reviews	Check ratings & reviews.

Figure 1: Start Menu

Figure 2: Sign Up Page

Figure 3: Sign In Page

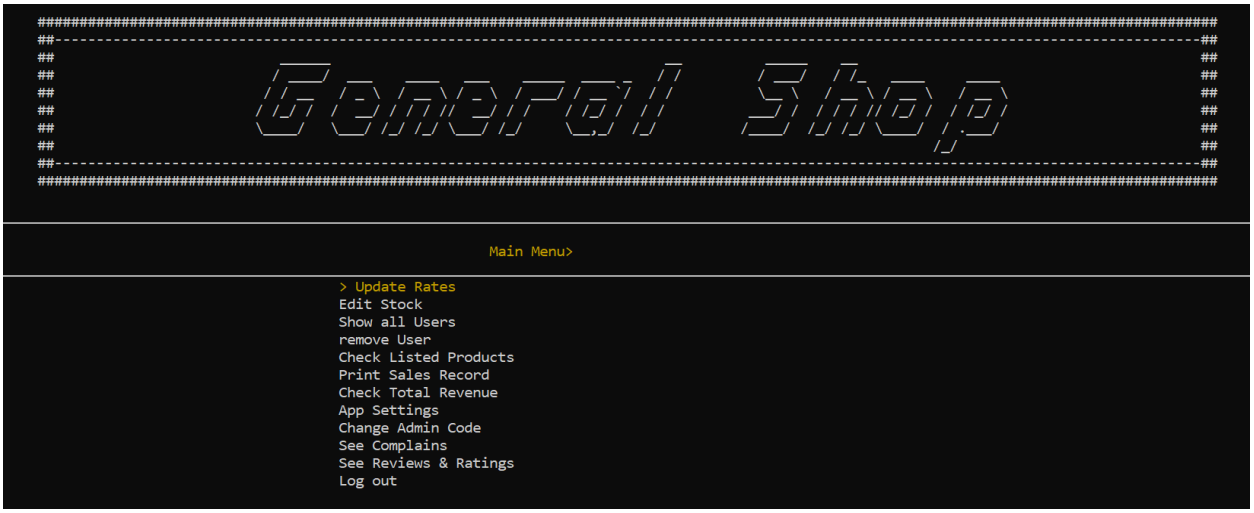


Figure 4: Manager Menu

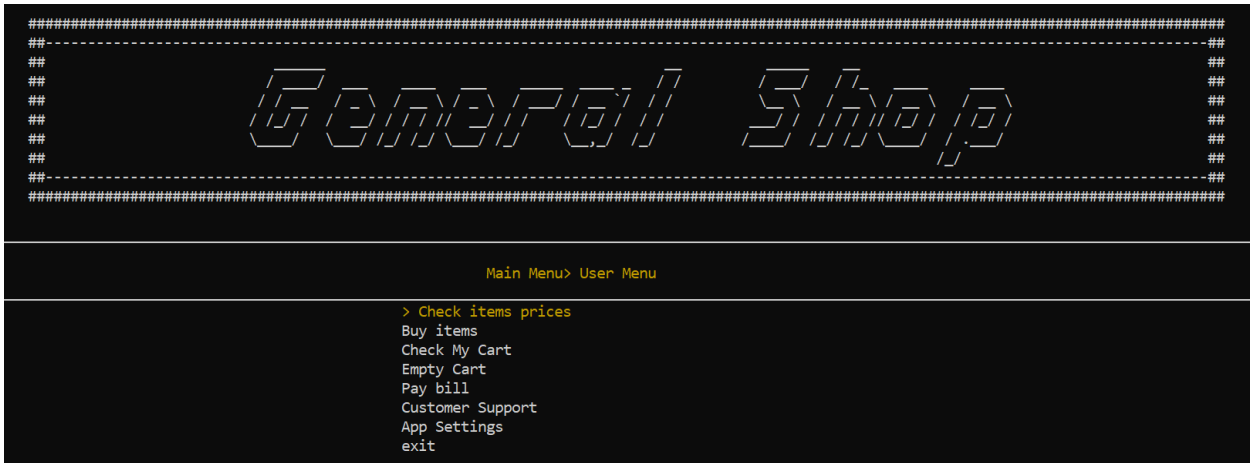


Figure 5: User Menu

Figure 6: Print Sales Menu

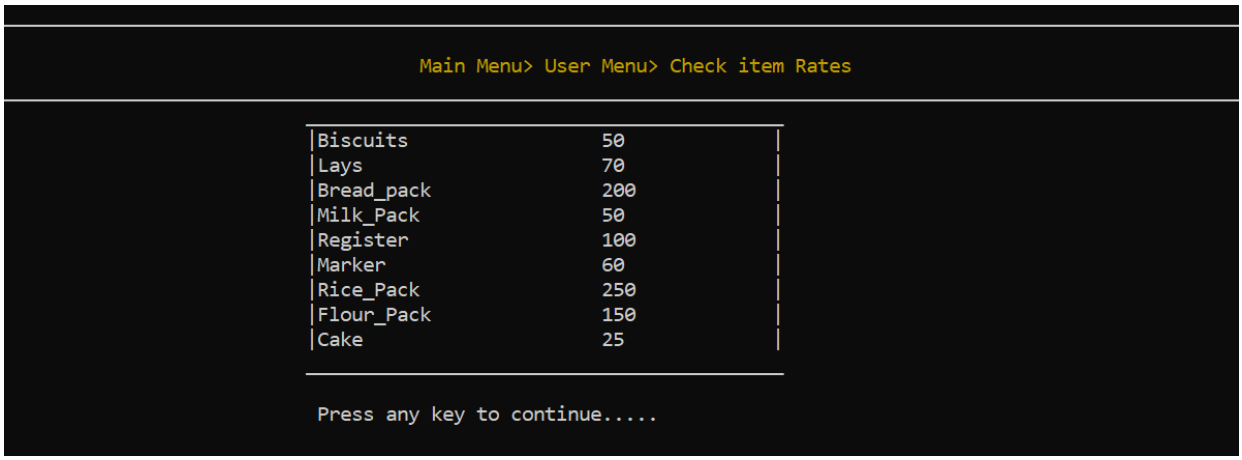


Figure 7: Listed Items Prices

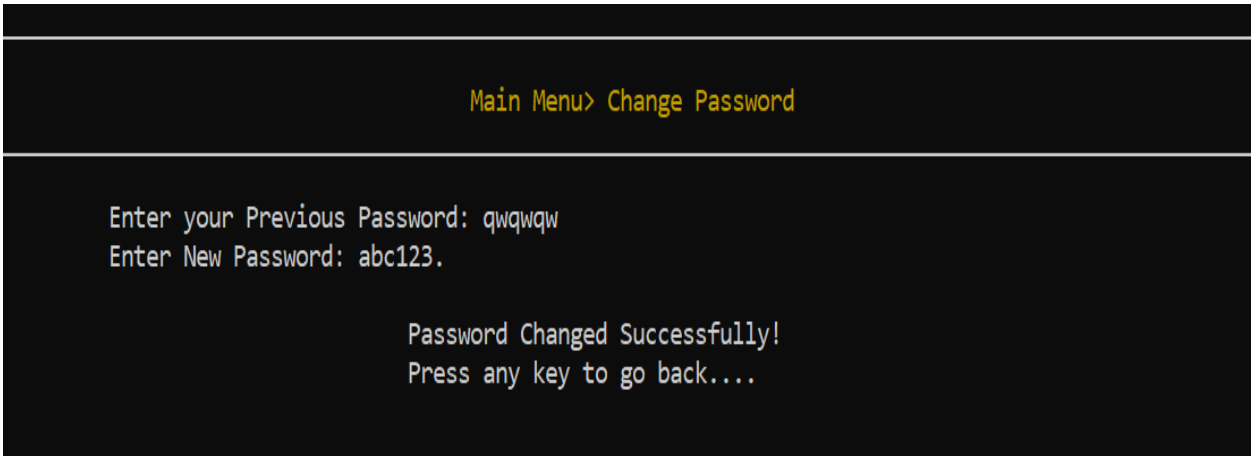


Figure 8: Change Password

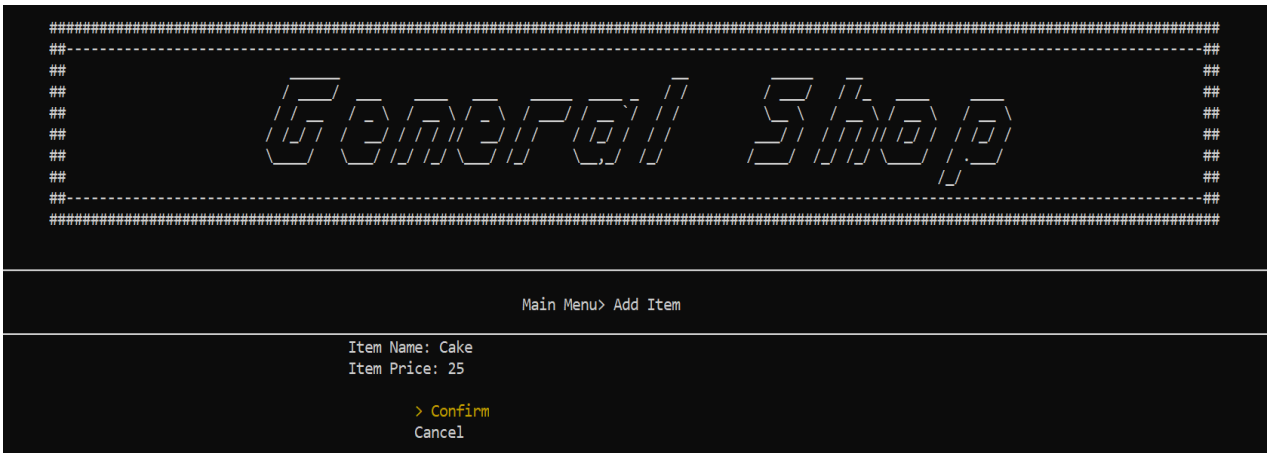


Figure 9: Add Item

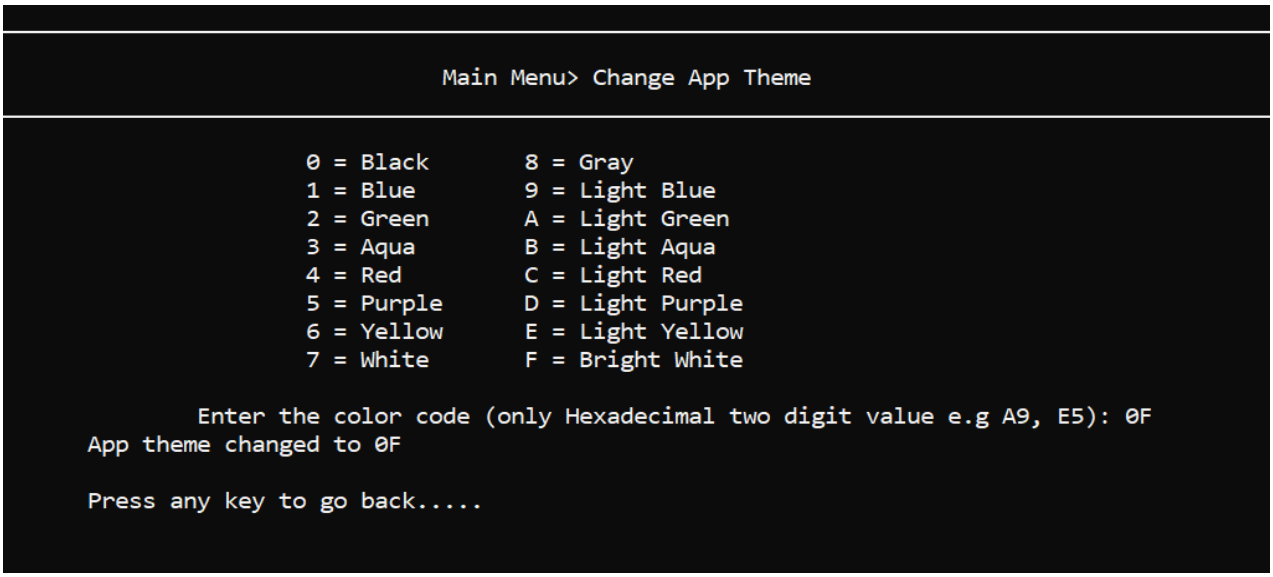


Figure 10: Change App Theme

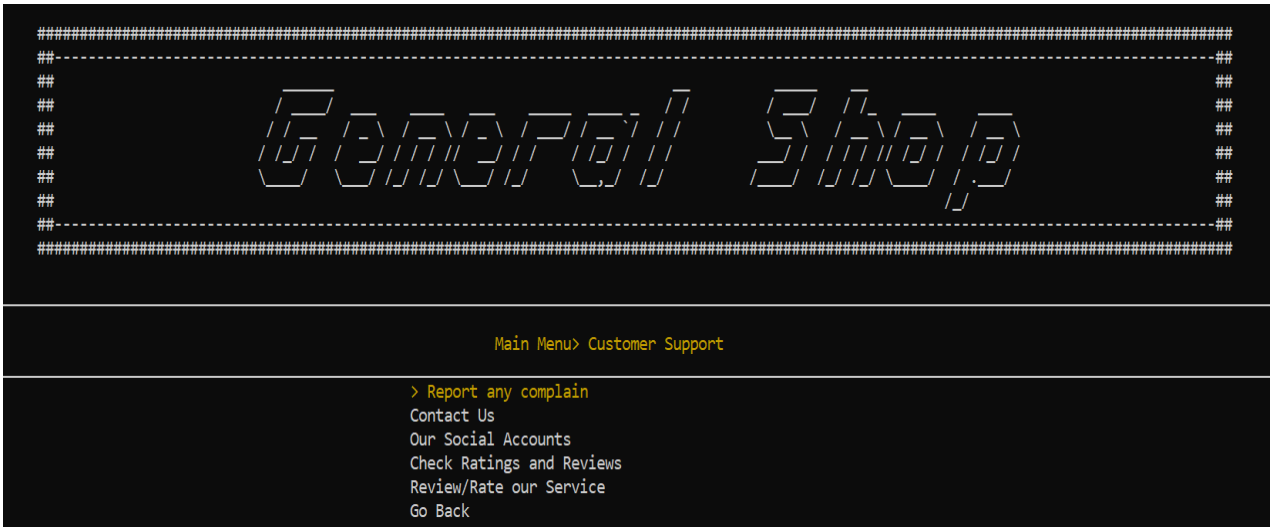


Figure 11: Customer Support Menu

2 Data Structures (Parallel Arrays)

- Parallel Arrays and variables used in General Store.

```
string products[30];
int ProductPrice[30];
int items_bought_by_user[30];
int total_bought_items[30];
 string user_Names_data[30];
string paswords_data[30];
string roles_data[30];
string review[30];
int ratings[30];
int review_index = 0;
```

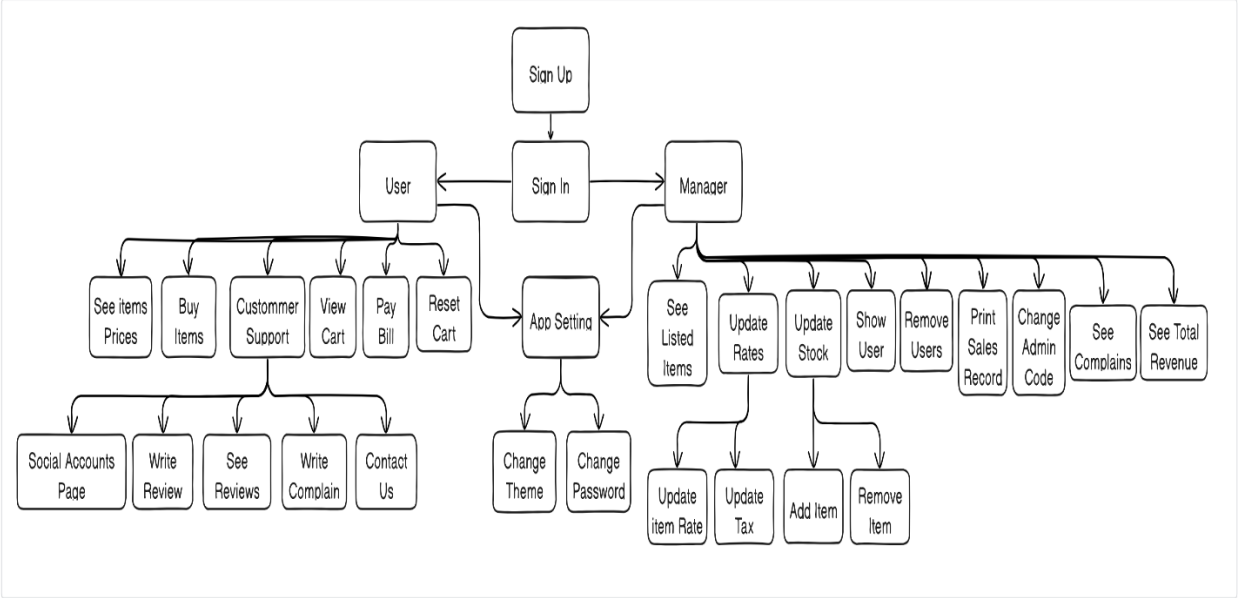
Function Prototypes

```
void start_menu(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
void smallMenu(string);
void tryAgain_Goback(int&,string);
void printHeader();
void SignIn_Header();
void SignUp_Header();
void contact_page();
void Social_Accounts_page();
// mutual functions ,both user and manager can use
void changeSystemColor(string &, string[], int &, int &);
void app_settings(string &, string, string[], string[], int &, string[], int &);
void change_Password(string, string[], string[], int &);
// SignIn Sign Up Functions
void sign_in(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
string signIn(string, string[], string, string[], string[], int &, string &);
string SignUp(string, string[], string, string[], string, string[], int &, string &);
void sign_Up_in(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
bool doesUserNameExists(string, string[], int &);
```

```
int userNameIndex(string, string[], int &);
// Validation Functions
bool isValid_UserName(string);
bool password_vald(string);
bool is_valid_color_code(string color_code);
static void gotoxy(int , int );
void Color(int);
void hideAndVisibleCursor(bool isShow);
void user(string[], int[], int[], int[], int &, int &, int &, string[], string &, string &,
string[], int &, string[], string[], int &, string[], int[], int &, int &);
void print_item_rates(string[], int[], int &);
void fn_buy_items(string[], int[], int &);
int bill_calculator(int[], int[], int &, int &);
int tax_calculator(int[], int[], int &, int &);
void View_Cart(string[], int[], int[], int &, int &);
void empty_cart(int[], int &);
void fn_paybill(int[], int[], int[], int &, int &, int &);
void customerSupport(string[], int &, string[], int[], int &);
void write_complain(string[], int &);
void review_rating(string[], int[], int &);
// manager
void manager(string[], int[], int[], int[], int &, int &, int &, string[], string[], string[],
int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
void printSalesRecord(string[], int[], int[], int &, int &);
void checkTotalRavenue(int &);
void updateRecord(string[], int[], int &, int &);
void CheckListedProducts(string[], int &);
void updateRates(string[], int[], int &);
void UpdateTaxRate(int &);
void showUsers(string[], string[], int &);
void removeUser(string[], string[], string[], int &);
void editStock(string[], int[], int[], int[], int &);
void addProduct(string[], int[], int[], int[], int &);
void remove_item(string[], int[], int[], int[], int &);
bool checkProduct(string[], string, int &);
void change_Admin_Code(string &);
void show_complains(string[], int);
void show_reviews(string[], int[], int);
```

```
// File handling
void load_users_data(string [],string [],string roles_data[], int &);
void save_users_data(string [],string [],string roles_data[], int &);
string parseData(string record, int field);
void loadProducts(string [],int [],int [],int [],int &);
void saveProducts(string [],int [],int [],int [],int);
void save_reviews(string [], int [],int);
void load_reviews(string [], int[],int &);
void save_complains(string [], int  );
void load_complains(string [], int & );
void load_admin_code(string admin_code);
void save_admin_code(string admin_code);
```

▯ Functions Working Flow



Complete Code of General Store:

```
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <windows.h>
#include <limits>
#include <fstream>
using namespace std;
// For Startup
```

```
void start_menu(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
void smallMenu(string);
void tryAgain_Goback(int&,string);
void printHeader();
void SignIn_Header();
void SignUp_Header();
void contact_page();
void Social_Accounts_page();

// mutual functions ,both user and manager can use
void changeSystemColor(string &, string[], int &, int &);
void app_settings(string &, string, string[], string[], int &, string[], int &);
void change_Password(string, string[], string[], int &);

// sign in and sign up
void sign_in(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
string signIn(string, string[], string, string[], string[], int &, string &);
string SignUp(string, string[], string, string[], string, string[], int &, string &);
void sign_Up_in(string[], string[], string[], int &, string[], int[], int[], int[], int &, int &, int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
bool doesUserNameExists(string, string[], int &);
int userNameIndex(string, string[], int &);

// Validation Functions
bool isValid_UserName(string);
bool password_vald(string);
bool is_valid_color_code(string color_code);

static void gotoxy(int , int );
void Color(int);
void hideAndVisibleCursor(bool isShow);
```

```
// functions for user
void user(string[], int[], int[], int[], int &, int &, int &, string[], string &, string &,
string[], int &, string[], string[], int &, string[], int[], int &, int &);
void print_item_rates(string[], int[], int &);
void fn_buy_items(string[], int[], int &);
int bill_calculator(int[], int[], int &, int &);
int tax_calculator(int[], int[], int &, int &);
void View_Cart(string[], int[], int[], int &, int &);
void empty_cart(int[], int &);
void fn_paybill(int[], int[], int[], int &, int &, int &);
void customerSupport(string[], int &, string[], int[], int &);
void write_complain(string[], int &);
void review_rating(string[], int[], int &);


// manager
void manager(string[], int[], int[], int[], int &, int &, int &, string[], string[], string[],
int &, string &, string &, string &, string[], string[], int &, string[], int[], int &, int &);
void printSalesRecord(string[], int[], int[], int &, int &);
void checkTotalRavenue(int &);
void Update_Rates(string[], int[], int &, int &);
void CheckListedProducts(string[], int &);
void updateRates(string[], int[], int &);
void UpdateTaxRate(int &);
void showUsers(string[], string[], int &);
void removeUser(string[], string[], string[], int &);
void editStock(string[], int[], int[], int[], int &);
void add_item(string[], int[], int[], int[], int &);
void remove_item(string[], int[], int[], int[], int &);
bool checkProduct(string[], string, int &);
void change_Admin_Code(string &);
void show_complains(string[], int);
void show_reviews(string[], int[], int);


// File handling
void load_users_data(string [],string [],string roles_data[], int &);
void save_users_data(string [],string [],string roles_data[], int &);
```

```
string parseData(string record, int field);
void load_items(string [],int [],int [],int [],int &);
void save_items(string [],int [],int [],int [],int);
void save_reviews(string [], int [],int);
void load_reviews(string [], int[],int &);
void save_complains(string [], int  );
void load_complains(string [], int & );
void load_admin_code(string admin_code);
void save_admin_code(string admin_code);


// Convert Integer string to integer number
int convertStoInt(string );

//main
main()
{

   int TotalRevenue = 0;
   string products[30];
   int ProductPrice[30];
   int items_bought_by_user[30];
   int total_bought_items[30];
   int productIndex = 0;
   int taxRate = 10;
   int totalSales = 0;

   for (int i = 0; i < productIndex; i++)          // Reset the items when the program
starts
   {
      total_bought_items[i] = 0;
      items_bought_by_user[i] = 0;
   }
   // DataBases
   string user_Names_data[30];
   string paswords_data[30];
   string roles_data[30];
   int userName_index = 0;
```

```
    // Complains System

    string complains[30];
    int complains_index = 0;

    // Admin Code:
    string admin_code;

    // Username variable for entered in sign in username, to be used in change
password
    string runing_username = "";

    // theme (app settings)
    int theme_index = 0;
    string themes_data[30];
    string theme_color = themes_data[theme_index];

    // Review System
    string review[30];
    int ratings[30];
    int review_index = 0;
    // Loading
    load_users_data(user_Names_data,paswords_data,roles_data,userName_index);

load_items(products,ProductPrice,items_bought_by_user,total_bought_items,product
Index);
    load_reviews(review,ratings,review_index);
    load_complains(complains, complains_index);
    load_admin_code(admin_code);

    // hide cursor
    hideAndVisibleCursor(false);

    // Starting Point  >> Sign In >> Sign Up Functions
    start_menu(user_Names_data, paswords_data, roles_data, userName_index,
products, ProductPrice, items_bought_by_user, total_bought_items, taxRate,
productIndex, TotalRevenue, admin_code, theme_color, runing_username,
```

```
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);

   //Saving
   save_users_data(user_Names_data,paswords_data,roles_data,userName_index);

save_items(products,ProductPrice,items_bought_by_user,total_bought_items,product
Index);
   save_reviews(review,ratings,review_index);
   save_complains(complains, complains_index);
   save_admin_code(admin_code);

}
        // Start Menu whcih contain Sign In Sign Up
void start_menu(string user_Names_data[], string paswords_data[], string
roles_data[], int &userName_index, string products[], int productPrice[], int
items_bought_by_user[], int total_bought_items[], int &taxRate, int &productIndex,
int &totalRevenue, string &admin_code, string &theme_color, string
&runing_username, string themes_data[], string complains[], int &complains_index,
string review[], int ratings[], int &review_index, int &theme_index)
{

   int opt_length = 3;
   string options[opt_length] = {"Sign In", "Sign Up", "Exit"};
   int current_opt = 0;
   char key;

   while (true)
   {
      printHeader();
      for(int i = 0; i < opt_length; i++){          // for colored keys
         if(i == current_opt){
            Color(06);
            cout <<"\t\t\t\t\t\t\t"<<"> "<<options[i] <<endl;
            Color(07);
         }
         else
         {
```

```
            cout <<"\t\t\t\t\t\t\t"<< options[i]<<endl;
        }
    }


    key = 0;
    while(!(key == 13 || key == 80 || key == 72))
    {
        key = _getch();
    }           // User can only use these three keys,   1.Enter     2.Up Key       3.
Down key

    if (key == 80)      // Down Key
    {
        if (current_opt < opt_length - 1)
        {
            current_opt++;
        }
    }
    else if (key == 72)     // Up Key
    {
        if (current_opt > 0)
        {
            current_opt--;
        }
    }
    else if (key == 13)         // Enter Key
    {
        if (current_opt == 0)
        {
            sign_in(user_Names_data, paswords_data, roles_data, userName_index,
products, productPrice, items_bought_by_user, total_bought_items, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_username,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
        else if (current_opt == 1)
        {
```

```
            sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_username,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
        else if (current_opt == 2)
        {
           break;        // Exit
        }
      }


   }
}

// Sign In Function that takes input
void sign_in(string user_Names_data[], string paswords_data[], string roles_data[],
int &userName_index, string products[], int ProductPrice[], int
items_bought_by_user[], int totalitems_bought_by_user[], int &taxRate, int
&productIndex, int &totalRevenue, string &admin_code, string &theme_color, string
&runing_username, string themes_data[], string complains[], int &complains_index,
string review[], int ratings[], int &review_index, int &theme_index)
{
   system("cls");
   SignIn_Header();
   int errorChoice=0;
   string signInOption;
   string userName;
   string password;

   cin.clear();
   cin.sync();
   cout <<"\t\t\t\t\t\t"<< "Enter userName: ";
   getline(cin, userName);
   cout <<"\t\t\t\t\t"<< "Enter Password: ";
   cin.clear();
   cin.sync();
```

```
    getline(cin, password);
    runing_username = userName;     // To be used in change Password

    signInOption = signIn(userName, user_Names_data, password, paswords_data,
roles_data, userName_index, admin_code);

    if (signInOption == "user")
    {
        user(products, ProductPrice, items_bought_by_user, totalitems_bought_by_user,
productIndex, taxRate, totalRevenue, user_Names_data, theme_color,
runing_username, paswords_data, userName_index, themes_data, complains,
complains_index, review, ratings, review_index, theme_index);
    }
    else if (signInOption == "manager")
    {
        manager(products, ProductPrice, items_bought_by_user,
totalitems_bought_by_user, productIndex, taxRate, totalRevenue, user_Names_data,
roles_data, paswords_data, userName_index, admin_code, theme_color,
runing_username, themes_data, complains, complains_index, review, ratings,
review_index, theme_index);
    }
    else if (signInOption == "Invalid Password")
    {
        cout <<"\t\t\t\t\t\t"<< "Invalid Password \n";
        Sleep(1500);
        tryAgain_Goback(errorChoice,"signIn");
        if (errorChoice == 0)
        {
            sign_in(user_Names_data, paswords_data, roles_data, userName_index,
products, ProductPrice, items_bought_by_user, totalitems_bought_by_user, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_username,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
        else if (errorChoice == 1)
        {
            start_menu(user_Names_data, paswords_data, roles_data, userName_index,
products, ProductPrice, items_bought_by_user, totalitems_bought_by_user, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_username,
```

```
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
    }
    else if (signInOption == "Not Found")
    {
        cout <<"\t\t\t\t\t"<< "Invalid userName.Rather Sign Up \n";
        Sleep(1500);
        tryAgain_Goback(errorChoice,"signIn");
        if (errorChoice == 0)
        {
            sign_in(user_Names_data, paswords_data, roles_data, userName_index,
products, ProductPrice, items_bought_by_user, totalitems_bought_by_user, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_username,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
        else if (errorChoice == 1)
        {
            start_menu(user_Names_data, paswords_data, roles_data, userName_index,
products, ProductPrice, items_bought_by_user, totalitems_bought_by_user, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_username,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
        }
    }
}
// Sign In function that checks and returns
string signIn(string userName, string user_Names_data[], string password, string
paswords_data[], string roles_data[], int &userName_index, string &admin_code)
{
    int index;
    string result;
    bool isAuser;
    isAuser = doesUserNameExists(userName, user_Names_data, userName_index);
// Check if username already exists
    if (isAuser)
    {
```

```
        index = userNameIndex(userName, user_Names_data, userName_index);
    // Finds index of entered username
        if (password == paswords_data[index])
        {
            if (roles_data[index] == "user")
            {
                result = "user";
            }
            else if (roles_data[index] == "manager")
            {
                result = "manager";
            }
        }
        else
        {
            result = "Invalid Password";
        }
    }
    else
    {
        result = "Not Found";
    }
    return result;
}
// Sign Up Function for user input
void sign_Up_in(string user_Names_data[], string paswords_data[], string
roles_data[], int &userName_index, string products[], int productPrice[], int
items_bought_by_user[], int total_bought_items[], int &taxRate, int &productIndex,
int &totalRevenue, string &admin_code, string &theme_color, string
&runing_usernme, string themes_data[], string complains[], int &complains_index,
string review[], int ratings[], int &review_index, int &theme_index)
{
    if(getch() == 27){
        return;
    }

    string adminCode;
    system("cls");
```

```
    SignUp_Header();
    string userName, password, role, signUp_op;
    int errorChoice;
    cout <<"\t\t\t\t\t\t"<< "Enter User Name: ";
    getline(cin, userName);
    if (isValid_UserName(userName))     // Username Validation
    {
       cin.clear();
       cin.sync();
       cout <<"\t\t\t\t\t"<< "Enter Password (min 6 characters && No Space && No
Comma): ";
       getline(cin, password);

       if(password_vald(password)){           // Password Validation

          int current_opt = 0, option_length = 2;
          string role_keys[option_length] = {"Manager", "User"};
          char key;

          while(true){
             system("cls");
             SignUp_Header();
             cout <<"\t\t\t\t\t"<< "Select Your Role  "<<endl;
             for(int i = 0; i <2; i++){
                if(i == current_opt){
                   Color(06);
                   cout <<"\t\t\t\t\t\t\t"<<"> "<<role_keys[i] <<endl;
                   Color(07);
                   }
                else
                {
                   cout <<"\t\t\t\t\t\t\t"<< role_keys[i]<<endl;
                }
             }

             key = 0;
             while(!(key == 13 || key == 80 || key == 72)){
             key = _getch();
```

```
            }

            if(key == 80){
                if(current_opt < option_length - 1){
                    current_opt++;
                }
            }
            else if(key == 72){
                if(current_opt > 0){
                    current_opt--;
                }
            }
            else if(key == 13){
                if(current_opt == 0){
                    role = "manager";
                break;
                }
            else if(current_opt == 1){
                role = "user";
                break;
            }
        }
        }

    signUp_op = SignUp(userName, user_Names_data, password, paswords_data,
role, roles_data, userName_index, admin_code);

        if (signUp_op == "user")
        {
            user_Names_data[userName_index] = userName;
            paswords_data[userName_index] = password;
            roles_data[userName_index] = role;
            userName_index++;
            cout <<"\t\t\t\t\t\t"<< "Signed Up Successfully...";
            Sleep(2000);
        }
        else if (signUp_op == "manager")
        {
```

```
            cin.clear();
            cin.sync();
            cout <<"\t\t\t\t\t\t\t"<< "Enter admin code: ";     // Special Code to Sign Up
as a Manager
            getline(cin ,adminCode);
            if(adminCode == admin_code)
            {
               user_Names_data[userName_index] = userName;
               paswords_data[userName_index] = password;
               roles_data[userName_index] = role;
               userName_index++;
            }
            else
            {
               cout <<"\t\t\t\t\t\t\t"<< "Admin code is incorrect   \n";
               Sleep(1500);
               tryAgain_Goback(errorChoice,"signUp");
               if (errorChoice == 0)
               {
                  sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
               }
               else if (errorChoice == 1)
               {
               start_menu(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
               }
            }
         }
         else if (signUp_op == "incorrect Password")
         {
```

```
            cout <<"\t\t\t\t\t\t"<< "Password length must at least be 6 characters long
\n\n";
            Sleep(1500);
            tryAgain_Goback(errorChoice,"signUp");
            if (errorChoice == 0)
            {
                sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
            else if (errorChoice == 1)
            {
                start_menu(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
        }
        else if (signUp_op == "password not acceptable")
        {
            cout <<"\t\t\t\t"<< "Password is inValid, It must include one alphabet, No
space and No Comma \n\n";
            Sleep(1500);
            tryAgain_Goback(errorChoice,"signUp");
            if (errorChoice == 0)
            {
                sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
            else if (errorChoice == 1)
            {
                start_menu(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
```

```
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
        }
        else if ("user already Exists")
        {
            cout <<"\t\t\t\t\t"<< "User already exists...";
            Sleep(1500);
            tryAgain_Goback(errorChoice,"signUp");
            if (errorChoice == 0)
            {
                sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
            else if (errorChoice == 1)
            {
                start_menu(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
        }

    }
    else
    {
        cout <<"\t\t\t"<< "Password is inValid, It must include one alphabet, No space
and No Comma \n\n";
        Sleep(1500);
        tryAgain_Goback(errorChoice,"signUp");
        if (errorChoice == 0)
        {
            sign_Up_in(user_Names_data, paswords_data, roles_data,
userName_index, products, productPrice, items_bought_by_user, total_bought_items,
```

```
taxRate, productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
            else if (errorChoice == 1)
            {
                start_menu(user_Names_data, paswords_data, roles_data, userName_index,
products, productPrice, items_bought_by_user, total_bought_items, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
            }
        }
    }
    else
    {
        cout <<"\t\t\t\t\t\t"<< "Username must include one alphabet and no space..\n\n";
        cout <<"\t\t\t\t\t\t"<< "Press any character to try again..\n";
        getch();
        sign_Up_in(user_Names_data, paswords_data, roles_data, userName_index,
products, productPrice, items_bought_by_user, total_bought_items, taxRate,
productIndex, totalRevenue, admin_code, theme_color, runing_usernme,
themes_data, complains, complains_index, review, ratings, review_index,
theme_index);
    }
}
//  Sign Up that checks and returns value according to sign up fn
string SignUp(string userName, string user_Names_data[], string password, string
paswords_data[], string role, string roles_data[], int &userName_index, string
&admin_code)
{
    system("cls");
    SignUp_Header();
    string result;
    bool isAlreadyUser;
    isAlreadyUser = doesUserNameExists(userName, user_Names_data,
userName_index);
    if (!isAlreadyUser)
    {
```

```
        if (password.length() >= 6)
        {
            if (password_vald(password))
            {
                if (role == "user" || role == "manager")
                {
                    if (role == "manager")
                    {
                            result = "manager";
                    }
                    else
                    {
                        result = "user";
                    }
                }
                else
                {
                    result = "incorrect Role";
                }
            }
            else
            {
                result = "password not acceptable";
            }
        }
        else
        {
            result = "incorrect Password";
        }
    }
    else
    {
        result = "user already Exists";
    }
    return result;
}
void tryAgain_Goback(int &errorChoice,string header){       // To be called in when
invalid Input in Sign in & Sign Up
```

```
string choice[2] = {"Try Again", "Go Back"};
char key;
int current_opt = 0;

while(true){
   system("cls");
   if(header == "signIn"){
      SignIn_Header();
   }
   else if(header == "signUp"){
      SignUp_Header();
   }

   cout <<"\t\t\t\t\t\t"<< "Try again or Go back\n";
   for(int i = 0; i <2; i++){
      if(i == current_opt){
         Color(06);
         cout <<"\t\t\t\t\t\t"<<"> "<<choice[i] <<endl;
         Color(07);
      }
      else
      {
         cout <<"\t\t\t\t\t\t"<< choice[i]<<endl;
      }
   }
// Keys
   key = 0;
   while(!(key == 13 || key == 80 || key == 72)){
      key = _getch();
   }

   if(key == 80){
      if(current_opt < 1){
         current_opt++;
      }
   }
```

```
        else if(key == 72){
            if(current_opt > 0){
                current_opt--;
            }
        }
        else if(key == 13){
            errorChoice = current_opt;
            break;
        }
    }
}
// User Name Checker
bool doesUserNameExists(string userName, string user_Names_data[], int
&userName_index)
{
    for (int i = 0; i < userName_index; i++)
    {
        if (user_Names_data[i] == userName)
        {
            return true;
        }
    }
    return false;
}
// To obtain username Index
int userNameIndex(string userName, string user_Names_data[], int
&userName_index)
{
    for (int i = 0; i < userName_index; i++)
    {
        if (user_Names_data[i] == userName)
        {
            return i;
        }
    }
}

// Manager Menu that contains all options
```

```
void manager(string products[], int ProductPrice[], int items_bought_by_user[], int total_bought_items[], int &productIndex, int &taxRate, int &totalRavenue, string user_Names_data[], string roles_data[], string passwords_data[], int &userName_index, string &admin_code, string &theme_color, string &runing_username, string themes_data[], string complains[], int &complains_index, string review[], int ratings[], int &review_index, int &theme_index)
{

   int opt_length = 12;
   string mangr_options[opt_length] = {"Update Rates", "Edit Stock", "Show all Users", "remove User", "Check Listed Products", "Print Sales Record", "Check Total Revenue", "App Settings", "Change Admin Code", "See Complains", "See Reviews & Ratings", "Log out"};
   int current_option = 0;
   char key;

   while (true)
   {
      printHeader();
      smallMenu("");
      for(int i = 0; i < opt_length; i++){
         if(i == current_option){
            Color(06);
            cout <<"\t\t\t\t\t\t"<<"> "<<mangr_options[i] <<endl;
            Color(07);
         }
         else
         {
            cout <<"\t\t\t\t\t\t"<< mangr_options[i]<<endl;
         }
      }
      // Keys
      key = 0;
      while(!(key == 13 || key == 80 || key == 72)){
         key = _getch();
      }

      if (key == 80)
```

```
        {
            if (current_option < opt_length - 1)
            {
                current_option++;
            }
        }
        else if (key == 72)
        {
            if (current_option > 0)
            {
                current_option--;
            }
        }
        else if (key == 13)
        {
            if (current_option == 0)
            {
                Update_Rates(products, ProductPrice, productIndex, taxRate);
            }
            else if (current_option == 1)
            {
                editStock(products, ProductPrice, items_bought_by_user,
total_bought_items, productIndex);
            }
            else if (current_option == 2)
            {
                showUsers(user_Names_data, roles_data, userName_index);
            }
            else if (current_option == 3)
            {
                removeUser(user_Names_data, passwords_data, roles_data,
userName_index);
            }
            else if (current_option == 4)
            {
                CheckListedProducts(products, productIndex);
            }
            else if (current_option == 5)
```

```
            {
                printSalesRecord(products, ProductPrice, total_bought_items, taxRate,
productIndex);
            }
            else if (current_option == 6)
            {
                checkTotalRavenue(totalRavenue);
            }
            else if (current_option == 7)
            {
                app_settings(theme_color, runing_username, passwords_data,
user_Names_data, userName_index, themes_data, theme_index);
            }
            else if (current_option == 8)
            {
                change_Admin_Code(admin_code);
            }
            else if (current_option == 9)
            {
                show_complains(complains, complains_index);
            }
            else if (current_option == 10)
            {
                show_reviews(review, ratings, review_index);
            }
            else if (current_option == 11)
            {
                break;
            }
        }
    }
}
// Show User functions to be used by manager
void showUsers(string user_Names_data[], string roles_data[], int
&userName_index)     // Show all users to Manager
{
    system("cls");
    printHeader();
```

```
    smallMenu("Check Users List");
    cout << endl;
    cout <<"\t\t\t\t\t\t\t"<< "Sr   " << setw(10) << "User Names" << setw(10) <<
"Roles" << endl;
    for (int i = 0; i < userName_index; i++)
    {
        cout <<"\t\t\t\t\t\t\t"<< i + 1 << " " << setw(10) << user_Names_data[i] <<
setw(15) << roles_data[i] << "\n";
    }
    cout <<"\n\t\t\t\t\t\t\t"<< "press any key to go back...";
    getch();
}
// Remove user Function to be used by manager
void removeUser(string user_Names_data[], string passwords_data[], string
roles_data[], int &userName_index)    // Remove User Function
{
    string user;
    int index;
    printHeader();
    smallMenu("Remove Users");
    cout <<"\t\t\t\t\t\t"<< "Users are: \n\n";
    cout <<"\t\t\t\t\t\t\t"<< "Sr   " << setw(10) << "User Names" << setw(10) <<
"Roles" << endl;
    for (int i = 0; i < userName_index; i++)
    {
        cout <<"\t\t\t\t\t\t\t"<< i + 1 << " " << setw(10) << user_Names_data[i] <<
setw(15) << roles_data[i] << "\n";
    }
    cout <<"\n\t\t\t\t\t\t\t"<< "Enter an index to delete: ";
    // To check wether the input is valid integer or not. If not ask again
    while(true){
        if(cin >> index)
        {
            break;
        }
        else
        {
            cout <<"\t\t\t\t\t\t\t" << "invalid index, Enter a valid Value: ";
```

```
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
        }
    }
    if(index < 0)       // Entered Index Validation
    {
        cout <<"\n\t\t\t\t\t\t\t"<< "Index must be positive! Try Again.";
        Sleep(1300);
        removeUser(user_Names_data,passwords_data,roles_data,userName_index);
    }
    if(index > userName_index)   // Entered Index Validation
    {
        cout << "\n\t\t\t\t\t\tIndex is out of range! Please enter a value within index: ";
        // Validator for index
        while(true){
            if(cin >> index)
            {
                break;
            }
            else
            {
                cout <<"\t\t\t\t\t\t" << "invalid index, Enter a valid Value: ";
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(),'\n');
            }
        }
    }
    else if (roles_data[index - 1] == "manager")    // Only Users can be deleted,
manager can't be
    {
        cout <<"\n\t\t\t\t\t\t\t"<< "You cannot delete any manager or yourself..\n\n";
        cout <<"\n\t\t\t\t\t\t\t"<< "Press any key to go back..\n";
        getch();
    }
    else
    {
        user = user_Names_data[index - 1];                          // Takes the Selected
index in "user"
```

```
        for (int i = index - 1; i < userName_index - 1; i++)    // Makes the selected index
data equal to next index
        {
           user_Names_data[i] = user_Names_data[i + 1];
           passwords_data[i] = passwords_data[i + 1];
           roles_data[i] = roles_data[i + 1];
        }
        user_Names_data[userName_index - 1] = "";          // Places null at the selected
index
        passwords_data[userName_index - 1] = "";
        roles_data[userName_index - 1] = "";
        userName_index--;
        cout <<"\n\t\t\t\t\t\t\t"<< user << " is removed\n";
        cout <<"\n\t\t\t\t\t\t\t"<< "press any key to continue....";
        getch();
        return;
    }

}
// Check Total Revenue For manger
void checkTotalRavenue(int &totalRavenue)              // Displays Total Revenue
Generated from sales
{
    system("cls");
    printHeader();
    smallMenu("Check Total Revenue");
    cout << endl;
    cout <<"\t\t\t\t\t\t"<< "     Total Ravenue Generated: " << totalRavenue << "\n";
    cout <<"\t\t\t\t\t\t"<< "_____\n\n";
    cout <<"\t\t\t\t\t\t"<< "press any key to continue....";
    getch();
}
// Manger can see all sales record from this menu
void printSalesRecord(string products[], int productPrice[], int total_bought_items[],
int &taxRate, int &productIndex)
{
    system("cls");
    printHeader();                  // Displays list of total sales of all products
```

```
    smallMenu("Sales menu");
    cout  << endl;
    cout <<"\t\t\t\t\t\t" <<
"_____\n";
    cout <<"\t\t\t\t\t\t" << "|----------------------------------------------|\n";
    cout <<"\t\t\t\t\t\t" << "| " << setw(12) << left << "index" << setw(15) << left <<
"item" << setw(15) << "sold"
         << "    |\n";
    for (int i = 0; i < productIndex; i++)
    {
       cout <<"\t\t\t\t\t\t" << "| " << setw(12) << left << i + 1 << setw(15) << left <<
products[i] << setw(13) << total_bought_items[i] << "      |\n";
    }
    cout <<"\t\t\t\t\t\t" << "|----------------------------------------------|\n";
    cout <<"\t\t\t\t\t\t" <<
"_____\n\n";
    cout <<"\t\t\t\t\t\t" << "press any key to go back";
    getch();
}
// Total Sales Calculator
int calculateTotalSale(int productPrice[], int total_bought_items[], int &taxRate, int
&productIndex)
{
    int bill = 0;                       // Total Sales Calculator
    for (int i = 0; i < productIndex; i++)
    {
       bill += productPrice[i] * total_bought_items[i];
    }
    bill += (bill * taxRate) / 100;
    return bill;
}
// Update Record where manager
void Update_Rates(string products[], int productPrice[], int &productIndex, int
&taxRate)
{
                              // A sub-Function where manager can Update item rates and
tax rate.
    int options_length = 3;
```

```
string record[options_length] = {"Update Rates", "Update Tax", "Go back"};
int current_opt = 0;
char key;

while (true)
{
   printHeader();
   smallMenu("Update Record");
   for(int i = 0; i <3; i++)
   {
      if(i == current_opt){              // Key Colors
         Color(06);
         cout <<"\t\t\t\t\t\t\t"<<"> "<<record[i] <<endl;
         Color(07);
      }
      else
      {
         cout <<"\t\t\t\t\t\t\t"<< record[i]<<endl;
      }
   }
               // Keys
   key = 0;
   while(!(key == 13 || key == 80 || key == 72)){
      key = _getch();
   }
   if (key == 80)
   {
      if (current_opt < options_length - 1)
      {
         current_opt++;
      }
   }
   else if (key == 72)
   {
      if (current_opt > 0)
      {
         current_opt--;
```

```
            }
        }
        else if (key == 13)
        {
            if (current_opt == 0)
            {
                updateRates(products, productPrice, productIndex);
            }
            else if (current_opt == 1)
            {
                UpdateTaxRate(taxRate);
            }
            else if (current_opt == 2)
            {
                break;
            }
        }
    }
}
// Check listed items in the store for manager
void CheckListedProducts(string products[], int &productIndex)
{
    system("cls");
    printHeader();
    smallMenu("Listed Items");
    cout <<"\t\t\t\t\t\t\t"<< "Listed Products are \n\n";
    for (int i = 0; i < productIndex; i++)
    {
        cout <<"\t\t\t\t\t\t\t"<< i + 1 << ": " << products[i] << "\n";
    }
    cout <<"\n\t\t\t\t\t\t\t"<< "press any key to go back...";
    getch();
}
// Tax updator Function
void UpdateTaxRate(int &taxRate)
{
    int newtax;
```

```
    cout << endl;
    cout <<"\t\t\t\t\t\t\t"<< "Previous tax rate was " << taxRate <<"\n\t\t\t\t\t\t\t"<<
"Enter new tax rate: ";
    // Input Validator
    while(true){
        if(cin >> newtax)
        {
            break;
        }
        else
        {
            cout <<"\t\t\t\t\t\t\t" << "invalid input Try an integer Value: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
        }
    }

    taxRate = newtax;
    cout << endl;
    cout <<"\t\t\t\t\t\t\t"<< "Tax Rate changed to > " << taxRate <<endl <<endl;
    cout <<"\t\t\t\t\t\t\t"<< "Enter any key to go back....";
    getch();
}
// Update Rates Function for manager to update tax and

void updateRates(string products[], int productPrice[], int &productIndex)
{                       // Update item rates Function
    system("cls");
    printHeader();
    smallMenu("Update Item Rates");
    int number, price;
    cout << endl;
    for (int i = 0; i < productIndex; i++)          // Displays items listed in the store
    {
        cout <<"\t\t\t\t\t\t\t"<< i + 1 << ":  " << products[i] << "\n";
    }
    cout <<"\t\t\t\t\t\t\t"<< productIndex + 1 << ": Done\n\n";
    while (true)
```

```
    {
        cout <<"\t\t\t\t\t\t\t"<< "select item from list to change rate: ";
        cin >> number;
        if (number <= productIndex)
        {
            cout <<"\t\t\t\t\t\t\t"<< "Last rate of " << products[number - 1] << " was " <<
productPrice[number - 1]<< endl << "\t\t\t\t\t\t\t enter new price ";
            // Input Validator
            while(true)
            {
                if(cin >> price)
                {
                    break;
                }
                else
                {
                cout <<"\t\t\t\t\t\t\t" << "invalid input Try an integer Value: ";
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(),'\n');
                }
            }

            productPrice[number - 1] = price;       // Sets the new price
        }
        else if (number == productIndex + 1)        // Done Option
        {
            for (int i = 0; i < productIndex; i++)          // Displays the products price
            {
                cout <<"\t\t\t\t\t\t\t"<< productPrice[i] << endl;
                Sleep(1000);
            }
            break;          // Exit from function
        }
    }
}
// Edit Stock Mw=enu where manager can add and remove item
void editStock(string products[], int ProductPrice[], int items_bought_by_user[], int
total_bought_items[], int &productIndex)
```

```
{
                        // Edits items in the stock
    int options_length = 3;
    string editStock[options_length] = {"Add Item", "Remove Item", "Go back"};
    int current_opt = 0;
    char key;

    while (true)
    {
        printHeader();
        smallMenu("Edit Record");
        for(int i = 0; i <3; i++)          // Keys Color
        {
            if(i == current_opt){
                Color(06);
                cout <<"\t\t\t\t\t\t\t" << "> "<<editStock[i] <<endl;
                Color(07);
            }
            else
            {
                cout <<"\t\t\t\t\t\t\t"<< editStock[i]<<endl;
            }
        }
    // Keys
        key = 0;
        while(!(key == 13 || key == 80 || key == 72)){
            key = _getch();
        }
        if (key == 80)
        {
            if (current_opt < options_length - 1)
            {
                current_opt++;
            }
        }
        else if (key == 72)
        {
```

```
            if (current_opt > 0)
            {
                current_opt--;
            }
        }
        else if (key == 13)
        {
            if (current_opt == 0)
            {
                add_item(products, ProductPrice, items_bought_by_user,
total_bought_items, productIndex);
            }
            else if (current_opt == 1)
            {
                remove_item(products, ProductPrice, items_bought_by_user,
total_bought_items, productIndex);
            }
            else if (current_opt == 2)
            {
                break;
            }
        }
    }
}
// Manager Can add any item in this menu
void add_item(string products[], int ProductPrice[], int items_bought_by_user[], int
total_bought_items[], int &productIndex)
{                       // Function to add item in stock
    system("cls");
    printHeader();
    smallMenu("Add Item");
    string product_name;
    int price;
    cout <<"\t\t\t\t\t\t"<< "Enter the name of Item: ";
    cin >> product_name;
    cout <<"\t\t\t\t\t\t"<< "Enter Price: ";
    // Validator
    while(true){
```

```
    if(cin >> price)
    {
        break;
    }
    else
    {
        cout <<"\t\t\t\t\t\t\t" << "invalid input Try an integer Value: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(),'\n');
    }
}
if(price < 0){
    price *= -1;
}

string choices[2] = {"Confirm", "Cancel"};
char key;
int current_opt = 0;

while(true){
    printHeader();
    smallMenu("Add Item");
    cout <<"\t\t\t\t\t\t"<< "Item Name: " << product_name << "\n";
    cout <<"\t\t\t\t\t\t"<< "Item Price: " << price << "\n\n";            // Confirmation
    for(int i = 0; i <2; i++)
    {
        if(i == current_opt){
            Color(06);
            cout <<"\t\t\t\t\t\t\t"<<"> "<<choices[i] <<endl;
            Color(07);
        }
        else
        {
            cout <<"\t\t\t\t\t\t\t"<< choices[i]<<endl;
        }
    }
    key = 0;
```

```
        while(!(key == 13 || key == 80 || key == 72)){
           key = _getch();
        }

        if(key == 80){
           if(current_opt < 1){
              current_opt++;
           }
        }
        else if(key == 72){
           if(current_opt > 0){
              current_opt--;
           }
        }
        else if(key == 13)
        {
           if(current_opt == 0){
              if (!checkProduct(products, product_name, productIndex))           // Check
if it already present in stock
              {
                 products[productIndex] = product_name;          // Add the item
                 ProductPrice[productIndex] = price;
                 items_bought_by_user[productIndex] = 0;
                 total_bought_items[productIndex] = 0;
                 productIndex++;
                 cout << endl;
                 cout <<"\t\t\t\t\t\t"<< product_name << " is added successfully to the list
.....\n";
                 cout <<"\t\t\t\t\t\t\t"<< "Enter any key to continue";
                 getch();
                 break;
              }
              else
              {
                 cout << product_name << " already exists....";
                 cout << "Enter any key to continue";
                 getch();
              }
```

```
            }
        else if(current_opt == 1)
        {
            break;      // Go Back Key
        }
        }
    }

}
// Manager can Remove any item from this menu
void remove_item(string products[], int productPrice[], int items_bought_by_user[],
int total_bought_items[], int &productIndex)
{
    system("cls");                  // Removes any item from stock
    printHeader();
    smallMenu("Remove X item ");
    string product;
    int index;
    cout <<"\t\t\t\t\t\t\t"<< "Products are: \n\n";
    for (int i = 0; i < productIndex; i++)
    {
        cout <<"\t\t\t\t\t\t\t"<< i + 1 << setw(10) << left << " " << products[i] << "\n";
    }
    cout <<"\t\t\t\t\t\t\t"<< "Enter an index to delete: ";
    // Validator
    while(true){
        if(cin >> index)
        {
            break;
        }
        else
        {
            cout <<"\t\t\t\t\t\t\t" << "invalid input Try an integer Value: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
        }
    }
```

```
    if(index > productIndex)
    {
        cout <<"\t\t\t\t\t\t\t" << "Index does not exists, Please enter valid index from
above:";
        // validator
        while(true)
        {
        if(cin >> index)
        {
            break;
        }
        else
        {
            cout <<"\t\t\t\t\t\t\t" << "Again invalid input Try an integer Value: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),'\n');
        }
    }

    }
    else
    {
        product = products[index - 1]; // takes the selected index name in variable
        for (int i = index - 1; i < productIndex - 1; i++)     // loop above from the selected
value to end of array
        {
            products[i] = products[i + 1];                    // Increments the index of each
item by one
            productPrice[i] = productPrice[i + 1];
            items_bought_by_user[i] = items_bought_by_user[i + 1];
            total_bought_items[i] = total_bought_items[i + 1];
        }
        products[productIndex - 1] = "";                 // Places null at name and 0 at it's
price
        productPrice[productIndex - 1] = 0;
        items_bought_by_user[productIndex - 1] = 0;
        total_bought_items[productIndex - 1] = 0;
        productIndex--;
```

```
        cout <<"\t\t\t\t\t\t\t"<< product << " removed\n";
        cout <<"\t\t\t\t\t\t\t"<< "press any key to continue....";
        getch();
    }
}
// Check Wether the product present in stock
bool checkProduct(string products[], string product, int &productIndex)
{
    for (int i = 0; i < productIndex; i++)
    {
        if (products[i] == product)
        {
            return true;
        }
    }
    return false;
}
// Manager can Change Special Admin Code
void change_Admin_Code(string &admin_code)
{
    printHeader();
    smallMenu("Change Admin Code");
    string new_admin_code;
    cin.clear();
    cin.sync();
    cout <<"\t\t\t\t\t\t\t"<< "Enter new ADMIN CODE : ";
    getline(cin ,new_admin_code);
    admin_code = new_admin_code;
    cout <<"\t\t\t\t\t\t\t"<< "Admin code is now changed...\n";
    cout <<"\t\t\t\t\t\t\t"<< "press any key to go back....\n";
    getch();
}

// User Menu
void user(string products[], int productPrice[], int items_bought_by_user[], int
total_bought_items[], int &productIndex, int &taxRate, int &totalRevenue, string
usernames_data[], string &theme_color, string &runing_username, string
passwords_data[], int &userName_index, string themes_data[], string complains[], int
```

```
&complains_index, string review[], int ratings[], int &review_index, int
&theme_index)
{


   int options_length = 8;
   string user[options_length] = {"Check items prices", "Buy items", "Check My
Cart", "Empty Cart", "Pay bill", "Customer Support", "App Settings", "exit"};
   int current_opt = 0;
   char key;

   while (true)
   {
      printHeader();
      smallMenu("User Menu");

      for(int i = 0; i < options_length; i++)   // Keys Colors
      {
         if(i == current_opt){
            Color(06);
            cout <<"\t\t\t\t\t\t\t"<<"> "<<user[i] <<endl;
            Color(07);
         }
         else
         {
            cout <<"\t\t\t\t\t\t\t"<< user[i]<<endl;
         }
      }
                  //Keys
      key = 0;
      while(!(key == 13 || key == 80 || key == 72)){
         key = _getch();
      }


      if (key == 80)
      {
         if (current_opt < options_length - 1)
```

```
            {
                current_opt++;
            }
        }
        else if (key == 72)
        {
            if (current_opt > 0)
            {
                current_opt--;
            }
        }
        else if (key == 13)     // Enter Key
        {
            if (current_opt == 0)
            {
                print_item_rates(products, productPrice, productIndex);
            }
            else if (current_opt == 1)
            {
                fn_buy_items(products, items_bought_by_user, productIndex);
            }
            else if (current_opt == 2)
            {
                View_Cart(products, productPrice, items_bought_by_user, productIndex,
taxRate);
            }
            else if (current_opt == 3)
            {
                empty_cart(items_bought_by_user, productIndex);
            }
            else if (current_opt == 4)
            {
                fn_paybill(items_bought_by_user, total_bought_items, productPrice,
productIndex, taxRate, totalRevenue);
            }
            else if (current_opt == 5)
            {
```

```
            customerSupport(complains, complains_index, review, ratings,
review_index);
        }
        else if (current_opt == 6)
        {
            app_settings(theme_color, runing_username, passwords_data,
usernames_data, userName_index, themes_data, theme_index);
        }
        else if (current_opt == 7)
        {
            break;          // Exit/Logout from user menu
        }
      }
   }
}
// See item rates menu for user
void print_item_rates(string products[], int ProductPrice[], int &productIndex)
{
   system("cls");
   printHeader();                                  // Displays item Rates to user with price
   smallMenu("User Menu> Check item Rates");
   cout <<"\t\t\t\t\t\t\t"<< "_____________________________________________\n";

   for (int i = 0; i < productIndex; i++)
   {
      cout <<"\t\t\t\t\t\t\t"<< "|" << setw(25) <<left<< products[i] << setw(15) <<
ProductPrice[i] << setw(20) << "|"<<"\n";
   }
   cout <<"\t\t\t\t\t\t\t"<< "_____________________________________________\n\n";
   cout <<"\t\t\t\t\t\t\t"<< " Press any key to continue.....";  //Goes Backs
   getch();
}
// buy items menu for user
void fn_buy_items(string products[], int items_bought_by_user[], int &productIndex)
{                                       // Buy items from store
   system("cls");
   printHeader();
   smallMenu("User Menu> Buy Items");
```

```
cout << endl;
for (int i = 0; i < productIndex; i++)              //  Again Clears All items bougth
{
   items_bought_by_user[i] = 0;
}
int option, number;
cout <<"\t\t\t\t\t\t\t"<< "-----------------------------------" << endl;
cout <<"\t\t\t\t\t\t\t"<< "          Sr"<< "    Product Name    " << endl;
cout <<"\t\t\t\t\t\t\t"<< "-----------------------------------" << endl;
for (int i = 0; i < productIndex; i++)      // Displays product
{
   cout <<"\t\t\t\t\t\t\t"<< "          " << i + 1 << ":       " << products[i] << "\n";
}
cout <<"\t\t\t\t\t\t\t"<< "    " << productIndex + 1 << ": Done.....\n\n";
while (true)
{
   cout <<"\t\t\t\t\t\t\t"<< "    What do you want to buy: ";
   // Validator
   while(true){
      if( cin >> option)
      {
         break;
      }
      else
      {
         cout <<"\t\t\t\t\t\t"<< "Please enter a valid int value: ";
         cin.clear();
         cin.ignore(numeric_limits<streamsize>::max(),'\n');
      }
   }

   if(option == productIndex+1)
   {
      break;          // Done Option
   }
   if(option > productIndex+1)
   {
```

```
            cout <<"\t\t\t\t\t\t\t"<< "invalid option....\n";
            Sleep(950);
            fn_buy_items(products,items_bought_by_user,productIndex);
        }
        else          //  Item Quantity
        {
            cout <<"\t\t\t\t\t\t\t"<< "Enter number: ";
            while(true){
                if( cin >> number)
                {
                    break;
                }
                else
                {
                    cout <<"\t\t\t\t\t\t"<< "Please enter a valid int value: ";
                    cin.clear();
                    cin.ignore(numeric_limits<streamsize>::max(),'\n');
                }
        }
            items_bought_by_user[option-1] += number;
        }
    }
}
// Empty cart menu for user
void empty_cart(int items_bought_by_user[], int &productIndex)
{                   // Clears all items from cart
    printHeader();
    smallMenu("Empty Cart");

    cout <<"\t\t\t\t\t\t\t"<< "Press any key to empty cart..\n";
    getch();
    for (int i = 0; i < productIndex; i++)
    {
        items_bought_by_user[i] = 0;
    }
    cout <<"\t\t\t\t\t\t\t"<< "Your cart is empty now..\n";
    cout <<"\t\t\t\t\t\t\t"<< "Press any key to go back..\n";
```

```
    getch();

}
// View cart where user can see all items he bought  with bill and tax
void View_Cart(string products[], int productPrice[], int items_bought_by_user[], int
&productIndex, int &taxRate)
{
    system("cls");
    printHeader();
    smallMenu("User Menu> Check Bill");
    int bill, tax, total;
    bill = bill_calculator(items_bought_by_user, productPrice, productIndex, taxRate);
    tax = tax_calculator(items_bought_by_user, productPrice, productIndex, taxRate);
    total = bill + tax;
    cout <<"\t\t\t\t\t\t"<<
"_____\n
";
    cout <<"\t\t\t\t\t\t"<< "#                                                    #\n";
    cout <<"\t\t\t\t\t\t"<< "#  " << setw(10) << left << "Index" << setw(12) <<
"Product" << setw(12) << "Price" << setw(12) << "Quantity" << setw(11) << "Total"
         << "      #\n";
    for (int i = 0; i < productIndex; i++)
    {
        cout <<"\t\t\t\t\t\t"<< "#     " << setw(8) << left << i + 1 << setw(12) << left <<
products[i] << setw(12) << productPrice[i] << setw(12) << items_bought_by_user[i]
<< setw(11) << items_bought_by_user[i] * productPrice[i] << "     #\n";
    }
    cout <<"\t\t\t\t\t\t"<< "#                                                    #\n";
    cout <<"\t\t\t\t\t\t"<< "# ";
    Color(03);
    cout << setw(14) << left << "Bill: " << setw(9) << left << bill << setw(9) << right
<< "Tax: " << setw(9) << left << tax << setw(9) << right << "Total: " << setw(11) <<
left << total;
    Color(07);
    cout  << "   #\n";
    cout <<"\t\t\t\t\t\t"<< "#                                                    #\n";
```

```
    cout <<"\t\t\t\t\t\t\t"<<
"#_____________________________________________________________________#\
n";

    cout << endl;
    cout <<"\t\t\t\t\t\t\t"<< " press any key to go back.....";
    getch();
}
// Bill calculator for buoght items
int bill_calculator(int items_bought_by_user[], int productPrice[], int &productIndex,
int &taxRate)
{
    int bill = 0;                       // Calculates Bill
    for (int i = 0; i < productIndex; i++)
    {
        bill += (productPrice[i] * items_bought_by_user[i]);
    }
    return bill;
}
// Tax calculator wrt bought items
int tax_calculator(int items_bought_by_user[], int productPrice[], int &productIndex,
int &taxRate)
{                                       // Calculates Tax
    int bill, tax;
    bill = bill_calculator(items_bought_by_user, productPrice, productIndex, taxRate);
    tax = (bill * taxRate) / 100;
    return tax;
}
// Pay bill function for user
void fn_paybill(int items_bought_by_user[], int total_bought_items[], int
productPrice[], int &productIndex, int &taxRate, int &totalRevenue)
{
    system("cls");                      // Pay bill Function
    printHeader();
    smallMenu("User Menu> Pay Bill");
    int bill, tax, totalBill;
    for (int i = 0; i < productIndex; i++)          // Puts all bought items by user in
Total Bougth items
```

```
    {
        total_bought_items[i] += items_bought_by_user[i];
    }
    bill = bill_calculator(items_bought_by_user, productPrice, productIndex, taxRate);
    if (bill == 0)
    {
        cout << endl;              // Check if bill is zero
        cout <<"\t\t\t\t\t\t" << "No bill to pay.. \n "<<"\t\t\t\t\t\t"<<"Not any item is
added to the cart..\n\n"<<"\t\t\t\t\t\t"<< "Please buy items to pay bill..\n";
        cout <<"\t\t\t\t\t\t"<< "Press any key to go back..\n";
        getch();
        return;
    }
    tax = tax_calculator(items_bought_by_user, productPrice, productIndex, taxRate);
    totalBill = bill + tax;
    cout <<"\t\t\t\t\t\t"<< "Your Bill is: " << totalBill << "\n";       // Displayes bill
    cout <<"\t\t\t\t\t\t"<< "Press any character to pay bill";
    getch();
    printHeader();
    smallMenu("User Menu> Pay Bill");
    cout <<"\t\t\t\t\t\t"<< "Your Bill is Payed..\n"<<"\t\t\t\t\t\t"<<" Thanks for
shoping\n";
    cout <<"\t\t\t\t\t\t"<< "Press any key to go back..";
    getch();
    for (int i = 0; i < productIndex; i++)            // Resets the cart after paying bill
    {
        items_bought_by_user[i] = 0;
    }
    totalRevenue += totalBill;             // Adds total bill to revenue
}

// Customer Support Function where user write review, complain, contact us and
check reviews
void customerSupport(string complains[], int &complains_index, string review[], int
ratings[], int &review_index)
{
    int options_length = 6;
```

```
    string customerSupport[options_length] = {"Report any complain", "Contact Us",
"Our Social Accounts", "Check Ratings and Reviews", "Review/Rate our Service",
"Go Back"};
    int current_opt = 0;
    char key;

    while (true)
    {
        printHeader();
        smallMenu("Customer Support");
        for(int i = 0; i < options_length; i++){
            if(i == current_opt){                  // Keys Color
                Color(06);
                cout <<"\t\t\t\t\t\t\t"<<"> "<<customerSupport[i] <<endl;
                Color(07);
            }
            else
            {
                cout <<"\t\t\t\t\t\t\t"<< customerSupport[i]<<endl;
            }
        }
    // Keys
        key = 0;
        while(!(key == 13 || key == 80 || key == 72)){
            key = _getch();
        }


        if (key == 80)
        {
            if (current_opt < options_length - 1)
            {
                current_opt++;
            }
        }
        else if (key == 72)
        {
            if (current_opt > 0)
```

```
            {
                current_opt--;
            }
        }
        else if (key == 13)
        {
            if (current_opt == 0)
            {
                write_complain(complains, complains_index);
            }
            else if (current_opt == 1)
            {
                contact_page();
            }
            else if (current_opt == 2)
            {
                Social_Accounts_page();
            }
            else if (current_opt == 3)
            {
                show_reviews(review, ratings, review_index);
            }
            else if (current_opt == 4)
            {
                review_rating(review, ratings, review_index);
            }
            else if (current_opt == 5)
            {
                break;          // Go back
            }
        }
    }
}
// Complain Function for user
void write_complain(string complains[], int &complains_index)
{
    printHeader();
```

```
    smallMenu("Write Complain");
    string complain_content;
    cout << endl;
    cin.clear();
    cin.sync();
    cout <<"\t\t\t\t\t\t"<< "Write your complain Below: \n\n";
    getline(cin, complain_content);

    complains[complains_index] = complain_content;            //  Adds complain to
database
    complains_index++;

    cout <<"\t\t\t\t\t\t"<< "Your complain has been sent to the administrator..\n";
    Sleep(1500);
}
// Show complains function for Manager , only manager can see complains
void show_complains(string complains[], int complains_index)
{
    printHeader();
    smallMenu("Show Complains");                          // Show Complains to only
manager
    cout << endl;
    cout <<"\t\t\t\t"<< setw(5) << left << "Sr" << setw(10) << right <<
"Complains\n";
    for (int i = 0; i < complains_index; i++)
    {
        cout <<"\t\t\t\t"<< " " << setw(5) << left << i + 1 << complains[i] << endl
<<endl;
    }
    cout << endl;
    cout << "\t\t\t\t\t" << "_____________________________________" <<endl;
    cout <<"\t\t\t\t\t\t"<< "..Press any key to go back..";
    getch();
}
// Contact us page, Cell Number and Email
void contact_page()
{
    printHeader();                          // A void Page for contact us
```

```
    smallMenu("Contact Us Page");
    cout << endl;
    cout <<"\t\t\t\t\t"<< "You can can contact us by using the following media..\n\n";
    cout <<"\t\t\t\t\t"<< " Email : generalStore@gmail.com\n"; // Readable Only
    cout <<"\t\t\t\t\t"<< " Phone : 056-876952220         \n\n";
    cout <<"\t\t\t\t\t"<< "Press any key to go back..\n";
    getch();
}
// A simple Social Accounts page
void Social_Accounts_page()
{
    printHeader();
    smallMenu("Social Accounts Page");
    cout << endl;
    cout <<"\t\t\t\t\t"<< "Follow our page On Social Media Platforms to stay tuned
with updates..\n\n";
    cout <<"\t\t\t\t\t"<< "Facebook : www.facebook.com/generalStore    \n"; //
Readable Only
    cout <<"\t\t\t\t\t"<< "Instagram : www.instagram.com/generalStore  \n";
    cout <<"\t\t\t\t\t"<< "Twitter : www.twitter.com/generalStore      \n\n";
    cout <<"\t\t\t\t\t"<< "Press any key to go back..\n";
    getch();
}
//  Both user and manager can use app settings function to change password and
theme
void app_settings(string &theme_color, string runing_username, string
passwords_data[], string user_Names_data[], int &userName_index, string
themes_data[], int &theme_index)
{
    int options_length = 3;
    string app_settings[options_length] = {"Change App Theme", "Change Password",
"Go back"};
    int current_opt = 0;
    char key;

    while (true)
    {
        printHeader();
```

```
smallMenu("App Settings");

for(int i = 0; i < 3; i++)
{                                          // Keys Color
   if(i == current_opt){
      Color(06);
      cout <<"\t\t\t\t\t\t\t"<<"> "<<app_settings[i] <<endl;
      Color(07);
   }
   else
   {
      cout <<"\t\t\t\t\t\t\t"<< app_settings[i]<<endl;
   }
}
// Keys
key = 0;
while(!(key == 13 || key == 80 || key == 72)){
   key = _getch();
}

if (key == 80)
{
   if (current_opt < options_length - 1)
   {
      current_opt++;
   }
}
else if (key == 72)
{
   if (current_opt > 0)
   {
      current_opt--;
   }
}

else if (key == 13)
{
```

```
            if (current_opt == 0)
            {
               changeSystemColor(theme_color, themes_data, userName_index,
theme_index);
            }
            else if (current_opt == 1)
            {
               change_Password(runing_username, passwords_data, user_Names_data,
userName_index);
            }
            else if (current_opt == 2)
            {
               break;      // Go back
            }
         }
      }
}
// Review funtion for user where he/she can rate us
void review_rating(string review[], int ratings[], int &review_index)
{                        //  User can write a review
   printHeader();
   smallMenu("Rate Us");
   string comment;
   int User_rating;
   cin.clear();
   cin.sync();
   cout << endl;
   cout <<"\t\t\t\t"<< "Enter your Review: ";
   getline(cin, comment);
   cout <<"\t\t\t\t"<< "Enter Rating out of 10: ";
   // Keys for rating
   string rating_count[11] = {"0","1","2","3","4","5","6","7","8","9","10"};
   char key;
   int current_opt = 0;
   while(true){
      printHeader();
      smallMenu("Rate Us");
      cout <<"\t\t\t\t\t\t"<< "Your Comments: " << comment << "\n";
```

```
        cout <<"\t\t\t\t\t\t"<< "Select Rating out of 10 " << endl <<endl;
        cout <<"\t\t\t\t\t\t"<< rating_count[current_opt] << endl;
    // Keys
        key = 0;
        while(!(key == 13 || key == 80 || key == 72)){
           key = _getch();
        }

        if (key == 80)
        {
           if (current_opt < 10)
           {
              current_opt++;
           }
        }
        else if (key == 72)
        {
           if (current_opt > 0)
           {
              current_opt--;
           }
        }
        else if(key == 13)  //Enter
        {
           User_rating = current_opt;          // Adds review to database
           review[review_index] = comment;
           ratings[review_index] = User_rating;
           review_index++;
           break;              //returns
        }
     }
     cout << endl;
     cout <<"\t\t\t\t"<< "Thank You ! \n"<<"\t\t\t\t"<< "Your Review has been
    Submitted to the Administrator...\n\n ";
     cout <<"\t\t\t\t\t\t"<< "Press any key to go back ";
     getch();
    }
    // Show Reviews function , both user and manger can use this function
```

```
void show_reviews(string review[], int ratings[], int review_index)
{
   printHeader();
   smallMenu("See Reviews/Ratings");
   cout << endl;
   cout <<"\t\t\t\t\t"<< setw(7) << left << "No." << setw(12) << left << "Rating/10"
<< setw(12) << left << "Review" << endl;

   for (int i = 0; i < review_index; i++)  // Displays Reviews
   {
      cout <<"\t\t\t\t\t"<< setw(7) << left << i + 1 << "  " << setw(12) << left <<
ratings[i] << setw(12) << left << review[i] << endl;
   }
   cout << endl <<endl;
   cout <<"\t\t\t\t\t\t"<< "_____\n";
   cout << endl;
   cout <<"\t\t\t\t\t\t"<< "Press any key to go back..\n";
   getch();
}
// User and manager both can change password from this menu
void change_Password(string runing_username, string passwords_data[], string
user_Names_data[], int &userName_Index)
{
   printHeader();                          // Change Password Function, Both can use
   smallMenu("Change Password");
   int index = userNameIndex(runing_username, user_Names_data,
userName_Index);       // Gets the index of runing username
   int choice;
   string old_password, newpassword;
   cout << endl;
   cin.clear();
   cin.sync();
   cout <<"\t\t\t\t\t"<< "Enter your Previous Password: ";     // Confirmation of old
pasword
   getline(cin,old_password);

   if (old_password == passwords_data[index])
   {
```

```
        cin.clear();
        cin.sync();
        cout <<"\t\t\t\t\t"<< "Enter New Password: ";
        getline(cin, newpassword);
        if(newpassword.length()>=6 && password_vald(newpassword)){      // Validates
New Password
            passwords_data[index] = newpassword;        // Changes Password
            cout << endl;
            cout <<"\t\t\t\t\t\t\t\t"<< "Password Changed Successfully!  \n";
            cout <<"\t\t\t\t\t\t\t\t"<< "Press any key to go back....";
            getch();
            return;
        }
        else
        {
            cout << "\n\n\t\t\t\t\t\t\t"<< "The new entered password does not match the
criteria..\n";
            cout << "\t\t\t\t\t\t"<< "Password should be atleast 6 characters long and
includes atleast one alphabet..\n\n";
            Sleep(3300);
            change_Password(runing_username, passwords_data, user_Names_data,
userName_Index);      // Try Again
        }
    }
    else
    {
        cout << "\t\t\t\t\t"<< "Password did'nt matched ... \n";

        string options[2] = {"Try Again", "Go Back"};
        char key;
        int current_opt = 0;
        while(true){
            printHeader();
            smallMenu("Change Password");
            cout <<"\t\t\t\t\t\t\t\t"<< "Password did'nt matched ... \n\n";
            for(int i = 0; i <2; i++){          //Keys color
                if(i == current_opt)
                {
```

```
                Color(06);
                cout <<"\t\t\t\t\t\t\t"<<"> "<<options[i] <<endl;
                Color(07);
            }
            else
            {
                cout <<"\t\t\t\t\t\t\t"<< options[i]<<endl;
            }
        }      // Keys
        key = 0;
        while(!(key == 13 || key == 80 || key == 72)){
            key = _getch();
        }
        if (key == 80)
        {
            if (current_opt < 10)
            {
                current_opt++;
            }
        }
        else if (key == 72)
        {
            if (current_opt > 0)
            {
                current_opt--;
            }
        }
        else if(key == 13)
        {
            if(current_opt == 0)
            {
                change_Password(runing_username, passwords_data, user_Names_data,
userName_Index);  //Try Again
            }
            else if(current_opt == 1)
            {
                return; // Go back
            }
```

```
            }

        }
    }
}
// Change app theme function
void changeSystemColor(string &theme_color, string themes_data[], int
&userName_index, int &theme_index)
{
    printHeader();                  // Function to change System Color
    smallMenu("Change App Theme");
    string colorCode;
    cout << endl;
    cout << "\t\t\t\t\t\t\t" << "0 = Black       8 = Gray" <<endl;
    cout << "\t\t\t\t\t\t\t" << "1 = Blue        9 = Light Blue" <<endl;
    cout << "\t\t\t\t\t\t\t" << "2 = Green       A = Light Green" <<endl;
    cout << "\t\t\t\t\t\t\t" << "3 = Aqua        B = Light Aqua" <<endl;              //
Displays Manual
    cout << "\t\t\t\t\t\t\t" << "4 = Red         C = Light Red" <<endl;
    cout << "\t\t\t\t\t\t\t" << "5 = Purple      D = Light Purple" <<endl;
    cout << "\t\t\t\t\t\t\t" << "6 = Yellow      E = Light Yellow" <<endl;
    cout << "\t\t\t\t\t\t\t" << "7 = White       F = Bright White" <<endl;
    cout << endl;
    cin.clear();
    cin.sync();
    cout << "\t\t\t\t\t\t"<< "Enter the color code (only Hexadecimal two digit value e.g
A9, E5): ";
    getline(cin, colorCode);

    if(is_valid_color_code(colorCode)){

        string full_command = "color " + colorCode;
        system(full_command.c_str());              // Sets the color to entered Color Code
        themes_data[theme_index] = full_command;
        theme_index++;
        cout << "\t\t\t\t\t"<< "App theme changed to " << colorCode << endl <<endl;
        cout << "\t\t\t\t\t"<< "Press any key to go back.....";
        getch();
```

```
    }
    else                // Validation
    {
        cout << "\n\t\t\t\t"<< "       >>>>>>>>>>>>>>>>>>  Invalid color code. Please
enter a valid hexadecimal color code.   <<<<<<<<<<<<<<<<<<    \n\n\n";
        Sleep(1500);
        changeSystemColor(theme_color, themes_data, userName_index, theme_index);
// Again Redirects
    }
}

// Validations
// UserName Validation
bool isValid_UserName(string name)
{

    int hasAlphabet = 0;
    int hasNoSpace = 0;
    for (int i = 0; name[i] != '\0'; i++)
    {
        if (!(('a' <= name[i] && name[i] <= 'z') ||
              ('A' <= name[i] && name[i] <= 'Z') ||
              ('0' <= name[i] && name[i] <= '9')))
        {
            return false;
        }

        if (('a' <= name[i] && name[i] <= 'z'))
        {
            hasAlphabet = 1;
        }
        if (name[i] != ' ')
        {
            hasNoSpace = 1;
        }
    }
    return hasAlphabet && hasNoSpace;
}
```

```
// Password Validation
bool password_vald(string password)
{
   if (password.empty())                // Password Validation
   {
      return false;
   }
   int hasAlphabet = 0;
   int hasNoSpace = 1;
   int hasNoComma = 1;
   for (int i = 0; password[i] != '\0'; i++)
   {

      if (('a' <= password[i] && password[i] <= 'z') || ('A' <= password[i] &&
password[i] <= 'Z'))
      {
         hasAlphabet = 1;
      }
      if (password[i] == ' ')
      {
         hasNoSpace = 0;
      }
      if (password[i] == ',' || password[i]=='^')
      {
         hasNoComma = 0;
      }
   }
   return hasAlphabet && hasNoSpace && hasNoComma;
}
// Color Code VALIDATION IN CHANGE THEME
bool is_valid_color_code(string color_code)
{

   if((color_code.length() == 2)&&(color_code[0] >= 'A' && color_code[0] <= 'F') ||
(color_code[0] >= '0'&&color_code[0] <= '9')&&
     (color_code[1] >= 'A' && color_code[1] <= 'F') || (color_code[1] >=
'0'&&color_code[1] <= '9'))
     {
```

```
            return true;
        }
    return false;
}
// functions for keys and colors
void Color(int color)
{
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
}
static void gotoxy(int x, int y)
{
    COORD coordinates;
    coordinates.X = x;
    coordinates.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
coordinates);
}
// Hide cursor Function
void hideAndVisibleCursor(bool isShow)
{
    HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO info;
    info.dwSize = 100;
    info.bVisible = isShow;
    SetConsoleCursorInfo(consoleHandle, &info);
}

// File Handling
// Load Users
void load_users_data(string user_Names_data[],string passwords_data[],string
roles_data[], int &userName_Index){
    fstream file;
    string line;
    userName_Index = 0;
    file.open("usersData.txt" , ios::in);
    while(!file.eof()){
        getline(file,line);
        user_Names_data[userName_Index] = parseData(line ,0);
```

```
        passwords_data[userName_Index] = parseData(line,1);
        roles_data[userName_Index] = parseData(line, 2);
        userName_Index++;
    }
    file.close();
}
// Save users
void save_users_data(string user_Names_data[],string passwords_data[],string
roles_data[], int &userName_Index){
    fstream file;
    file.open("usersData.txt", ios::out);
    for (int i =0; i<userName_Index; i++)
    {
        file << user_Names_data[i];
        file << "^";
        file << passwords_data[i];
        file << "^";
        file << roles_data[i];
        file << "^";
        if( i != userName_Index -1){
            file << endl;
        }
    }
    file.close();
}
// ParsData function
string parseData(string line,int index)
{
    int idx = 0;
    string word = "";
    int wordnum = 0;
    while(line[idx] != '\0'){
        if(line[idx] == '^'){
          if(wordnum == index){
            return word;
          }
          word = "";
          wordnum++;
```

```
        }else{
          word += line[idx];
        }
        idx++;
    }
}
// Save Items Function
void save_items(string products[], int productPrice[], int items_bought_by_user[], int
total_bought_items[], int productIndex) {
    fstream dataFile;
    dataFile.open("products.txt", ios::out);

    for (int i = 0; i < productIndex; i++) {
        // Check if the product name is not empty before saving
        if (!products[i].empty()) {
            dataFile << products[i];
            dataFile << "^";
            dataFile << productPrice[i];
            dataFile << "^";
            dataFile << items_bought_by_user[i];
            dataFile << "^";
            dataFile << total_bought_items[i];
            dataFile << "^";
            if(i != productIndex -1){
                dataFile << endl;
            }
        }
    }

    dataFile.close();
}
// Load items function
void load_items(string products[],int Productprices[],int items_bought_by_user[],int
total_bought_items[],int &productIndex){
    string data;
    productIndex = 0;
    fstream file;
    file.open("products.txt",ios::in);
```

```
    while(!(file.eof())){
        getline(file,data);
        products[productIndex] = parseData(data,0);
        Productprices[productIndex] = convertStoInt((parseData(data,1)));
        items_bought_by_user[productIndex] = convertStoInt((parseData(data,2)));
        total_bought_items[productIndex] = convertStoInt((parseData(data,3)));
        productIndex++;
    }
    file.close();
}
// Save reviews function
void save_reviews(string reviews[], int rating[],int review_index)
{
    fstream file;
    file.open("reviews.txt" , ios :: out);
    for(int i = 0; i< review_index; i++){
        file << reviews[i];
        file << "^";
        file << rating[i];
        file << "^";
        if(i != review_index-1){
            file << endl;
        }
    }
    file.close();
}
// Load reviews function
void load_reviews(string reviews[], int rating[],int &review_index)
{
    fstream file;
    string line = "";
    file.open("reviews.txt",ios::in);
    while(getline(file,line))
    {
        reviews[review_index] = parseData(line,0);
        rating[review_index] = convertStoInt((parseData(line,1)));
        review_index++;
```

```
    }
}
// Save Complains Function
void save_complains(string complains[], int complains_index )
{
    fstream file;
    string line;
    file.open("complains.txt",ios::out);
    for(int i =0; i <complains_index; i++){
        file << complains[i];
        file << "^";
        if(i != complains_index-1){
            file << endl;
        }
    }
}
// Load Function
void load_complains(string complains[], int &complains_index)
{
    fstream file;
    string line;
    file.open("complains.txt");
    while(!file.eof())
    {
        getline(file, line);
        complains[complains_index]= parseData(line, 0);
        complains_index++ ;
    }
}
// Load admin code
void load_admin_code(string admin_code)
{
    fstream file;
    string line;
    file.open("admin_code.txt", ios::in);
        admin_code = parseData(line , 0);
    file.close();
```

```
}
// Load ADMIN CODE
void save_admin_code(string admin_code)
{
   fstream file;
   file.open("admin_code.txt", ios::out);
   file << admin_code;
   file.close();
}
// Function that works like stoi...  TAkes string number returns int number
int convertStoInt(string num){
   int number = 0;
   int crNum;
   int difference = 1;
   for(int i=num.length()-1; i>=0;i--){
      crNum = num[i] - '0';
      number += (crNum * difference);
      difference *= 10;
   }
   return number;
}
// Headers

// Small Menu Navigator
void smallMenu(string submenu)
{
   cout <<
"_____
_____
_____\n\n";
   Color(06);
   cout << "                                        Main Menu> " + submenu +
"\n";
   Color(07);
   cout <<
"_____
_____
_____\n";
```

```
}
// Sign Up Header
void SignUp_Header()
{
   cout << R"(

##########################################################################
#######################################
            ##-------------------------------------------------------------------------
----------------##
            ##        _____    _____     _____      __     _     __   __   _____
##
            ##       / ____\  (_   _)   / ___ \    / \   / )    ) ) ( ( ( __ \
##
            ##      ( (___      ||    / /  \_)  / /\ \ / /    ( (    ) ) ) )_) )
##
            ##       \___ \     ||   ( (  ____   ) ) ) ) ) )     ) ) ( ( ( ___/
##
            ##           \ )    ||   ( ( (__  ) ( ( ( ( ( (     ( (   ) ) ) )                  ##
            ##        ___/ /   _| |__  \ \__/ /  / /  \ \/ /      ) \__/ (  ( (
##
            ##       /____/   /_____/   \____/  (_/   \__/       \______/ /__\
##

##_____
_____##

##########################################################################
#######################################
_____
_____
_____
----------------------------------------------------------------------------------------------
----------------------------------------------------------)";
}
 // Sign In Header
void SignIn_Header()
{
   cout << R"(
```

```
##########################################################################
#############################################################
            ##-------------------------------------------------------------------------------
--------------------------------##
            ##            _____    _____     _____       __       _      _____      __
_              ##
            ##           / ____\ (_   _)   / ___ \      / \      / )     (_   _)     / \     / )
##
            ##          ( (___     | |    / /   \_)    / /\ \     / /       | |      / /\ \   / /
##
            ##           \___ \    | |    | | ____    / /  \ \   / /        | |     / /  \ \ / /
##
            ##               ) )   | |    | |(__  )   / /    \ \ / /         | |    / /    \ V /
##
            ##           ___/ /   _| |__   \ \__/ /   / /      \ V /         | |__  / /      \ V
##
            ##          /____/    /_____(    \____/   (_/       \__/         /_____( (_/
\                ##

##_____
_____##

##########################################################################
#############################################################
_____
_____
_____
-------------------------------------------------------------------------------------------------
-----------------------------------------------------------)";
}
// Main Header
void printHeader()
{
   system("cls");
   cout << R"(

##########################################################################
```

```
##########################################################################
#####
       ##-------------------------------------------------------------------------------------
-------------------------------------------##
       ##                  _______                          __       ______   __
##
       ##                / _____/ ____   _____  ____   ______ _____ _  / /      / ___/  /
/_  _____   _____              ##
       ##               / / __   / _ \ / __ \/ _ \ / ___/ / __ `/ / /       \__ \  / __ \/
__ \  / __ \              ##
       ##              / /_/ /  /  __/ / / / //  __/ / /   / /_/ / / /       ___/ / / / / // /_/ / /
/_/ /              ##
       ##              \_____/  \___/ /_/ /_/ \___/ /_/    \__,_/ /_/       /____/ /_/
/_/ \____/  / .___/              ##
       ##                                                                    /_/
##
       ##-------------------------------------------------------------------------------------
-------------------------------------------##

##########################################################################
##########################################################################
#####

)";
}
```

Weakness in the Business Application:

- Anyone who knows admin code can sign up as a Manager.
- This application does not include stock system. User can buy any number of items from General Store.
- The change app theme feature do not properly due to color keys. When the theme is changed in function, apparently
- The application blinks whenever it changes menu or option.
- There are less username and password validations.

Future Directions:

- Make it more engaging in the next versions by adding some more functions and making it in more realistic way.
- Make the change in theme functional fully functional for each user so that when he/she signs in again, his/her selected theme is loaded with his ID.
- Enhance the user interface to make it more attractive and easy to use.
- Enable Notification system so that the users will be alert of updates.
- Make payment in more than one currency for users.
- Make it more secure application, so that it is prevented from leakage of user information.