**Toast the Host**

(Catering Service)



Session: 2022 – 2026

**Submitted by:**

Shahneela Iqbal 2022-CS-159

**Supervised by:**

Dr. Muhammad Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[1. Short Description of project 3](#_Toc128692767)

[2. Users of Application 3](#_Toc128692768)

[3. Functional Requirements 3](#_Toc128692769)

[4. Wireframes 7](#_Toc128692770)

[5. Data Structure 15](#_Toc128692771)

[6. Functions Prototypes 16](#_Toc128692772)

[7. Work Flow 18](#_Toc128692773)

[8. Code 19](#_Toc128692774)

[9. Weakness 52](#_Toc128692775)

[10. Future Direction 52](#_Toc128692776)

Toast the Host

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Short Description of project

Idea behind my application is to provide clients with fine catering service and quality food via online portal without the hassle of going out.

With vast use of internet and smartphones nowadays that has made our live increasingly dependent on devices, this user-friendly platform will enable the customer to easily go through the portal and place his order in a few basic steps just by sitting home.

# Users of Application

* **Customers:** Customers are able to view and order the services from the site by sitting at their home. Customer are also able to give reviews.
* **Service Provider:** Who will ensure that every order is complete and service which is provided to clients is at its best. Who will able to add the products and services, delete and change the prices of products and services.

# Functional Requirements

* As a **service provider** I want to check the orders given by customer so that I can provide the services to the customer.

|  |  |  |
| --- | --- | --- |
| Service Provider | Add the services / products. | Add the products and services to their list. |
| Service Provider | Delete the services / products. | Remove the services and products from site. |
| Service Provider | Update the services/ products. | Change the name and prices of products. |
| Service Provide | View the orders. | Check the orders. |
| Service Provider | Add the service provider | He can add another service provider. |
| Service Provider | Calculate balance | He can check and calculate the balance of the company. |
| Service Provider | Change password | He can change password of his account. |
| Service Provider | View reviews | Check the reviews about services and products. |
| Service Provider | View list of services | He can see all the services provided by the company. |
| Service Provider | View food menu | He can see the food menu. |

* As a **customer** I can give order and reviews on site. And I can also delete the products from my cart.

|  |  |  |
| --- | --- | --- |
| Customer | View and add products / services | Add products / services to the cart. |
| Customer | Orders | Place orders. |
| Customer | Reviews | Give reviews about services and products. |
| Customer | Log in | He can login. |
| Customer | Favorite | He can add items in favorite category. |
| Customer | View favorite | He can see favorite products. |

# Wireframes

****

*Figure 1: Main Screen*

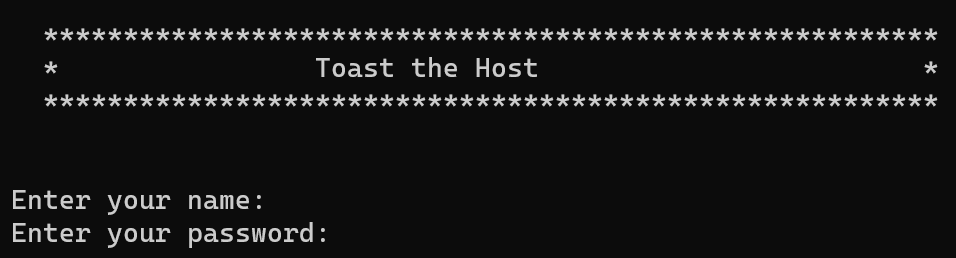


Figure 2: Admin Login Screen

Figure 3: Admin main menu

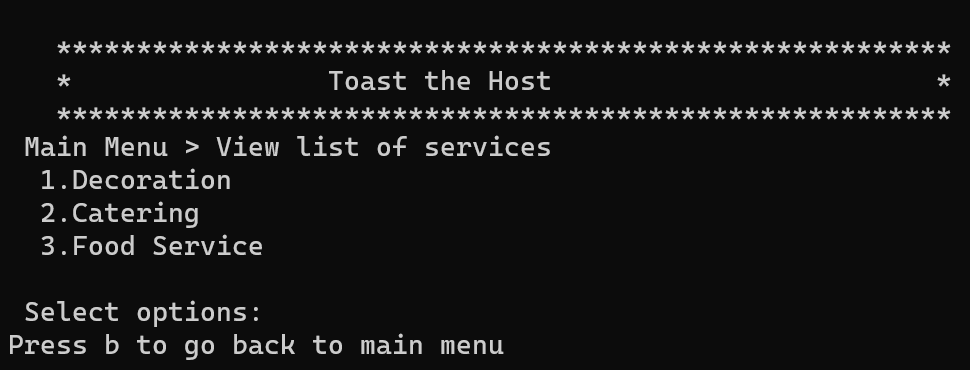


Figure 4: View list of services

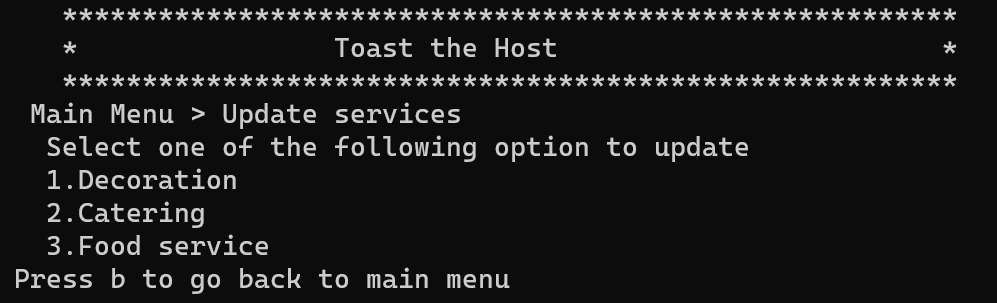


Figure 5a: Update Services

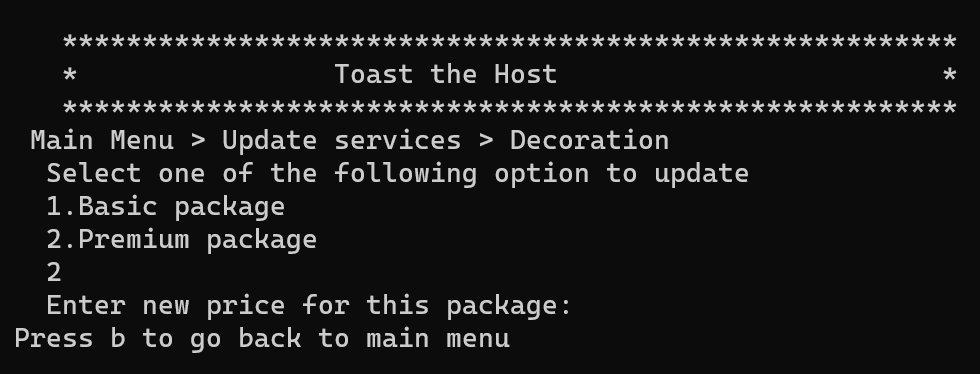


Figure 5b: Update Services

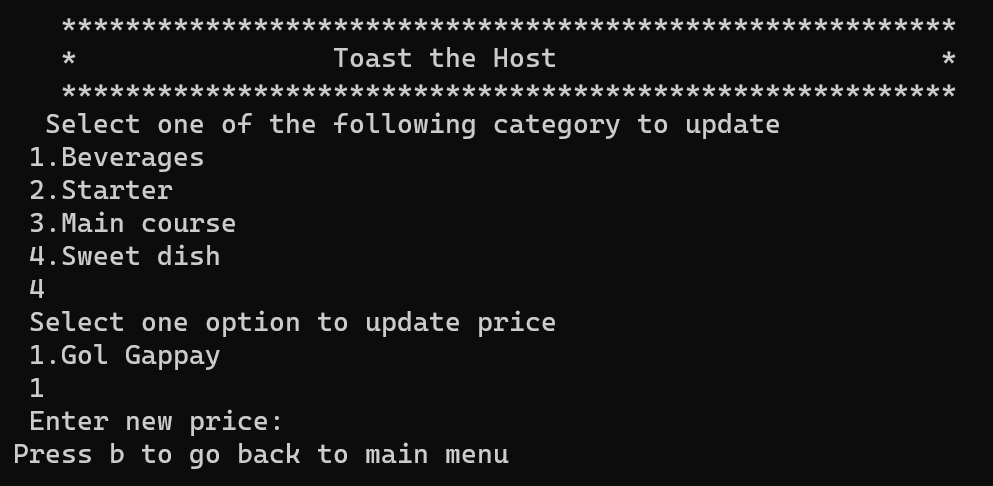


Figure 6: Update food menu

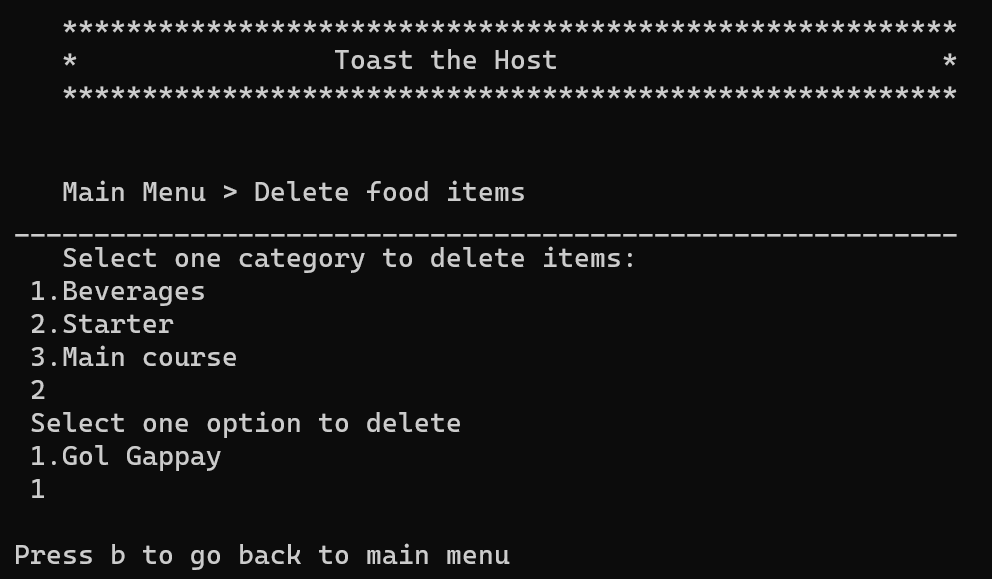


Figure 7: Delete food items and services

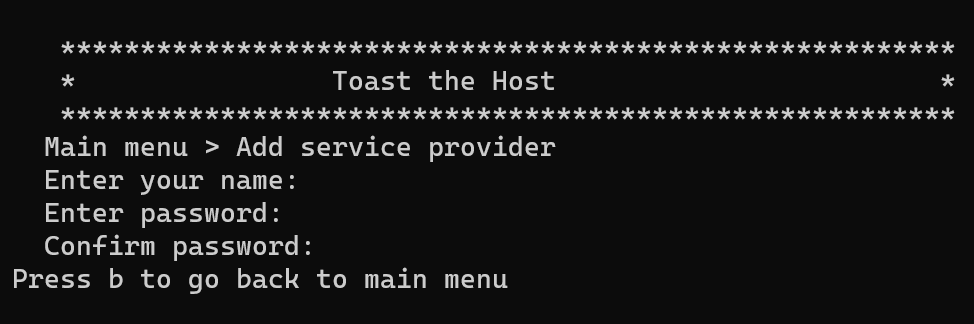


Figure 8: Add Service provider

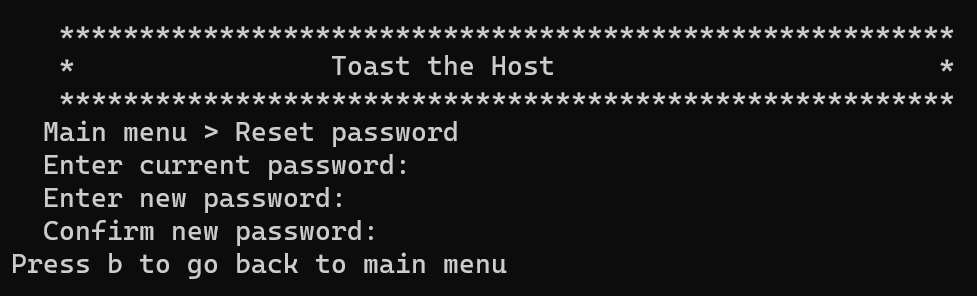


Figure 9: Reset Password

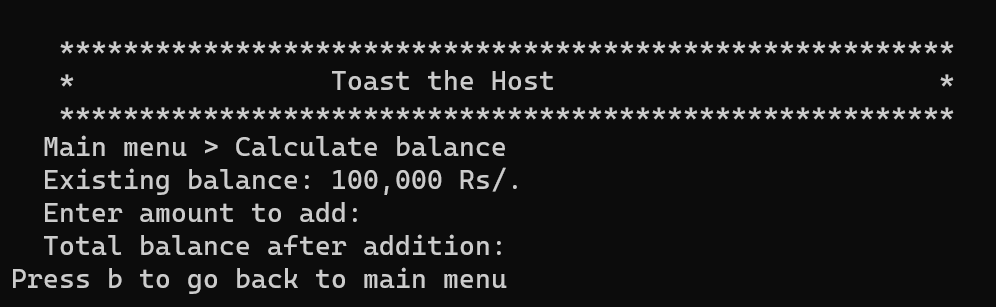


Figure 10: Calculate balance

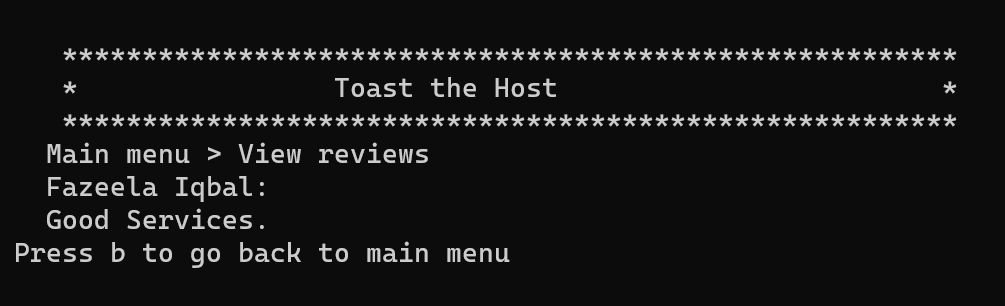


Figure 11: View reviews

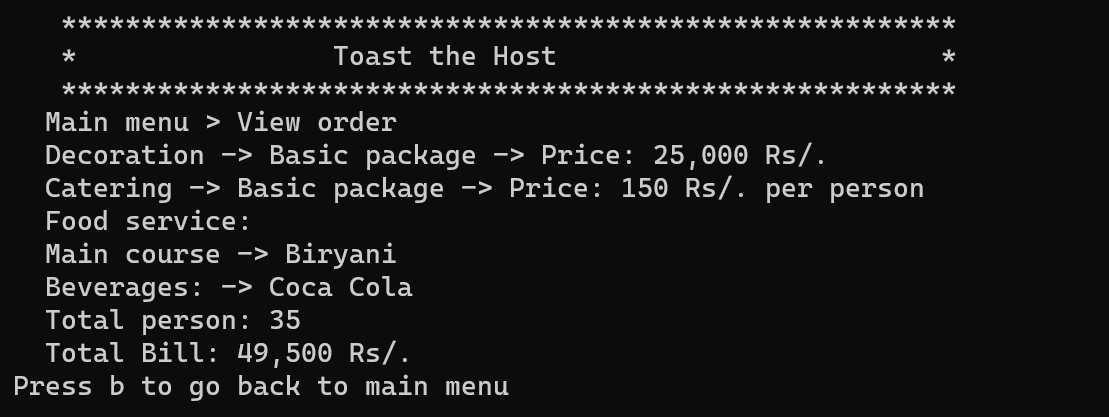


Figure 12: View order

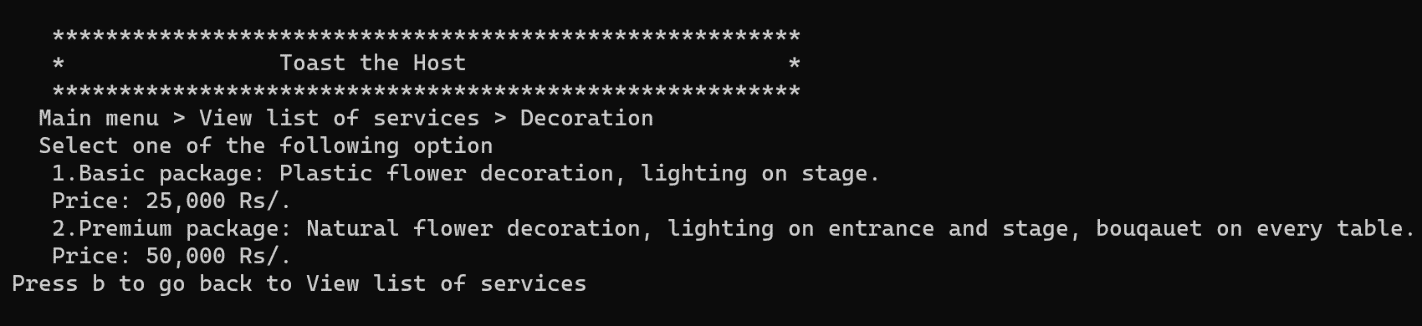


Figure 13: Decoration

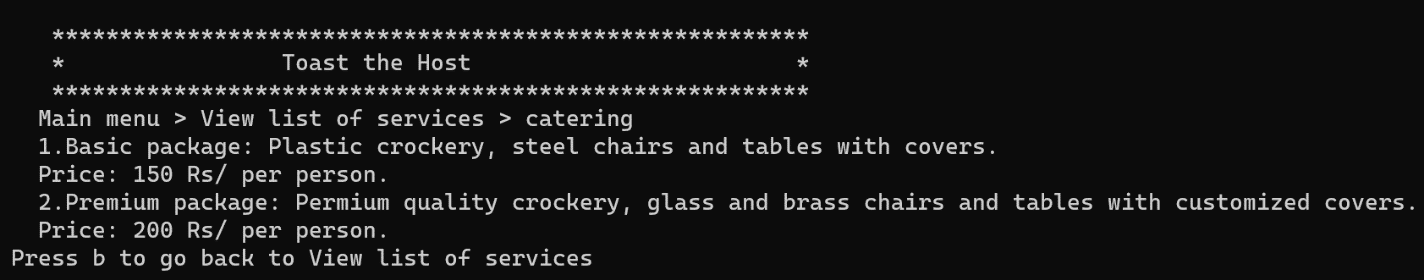


Figure 14: Catering

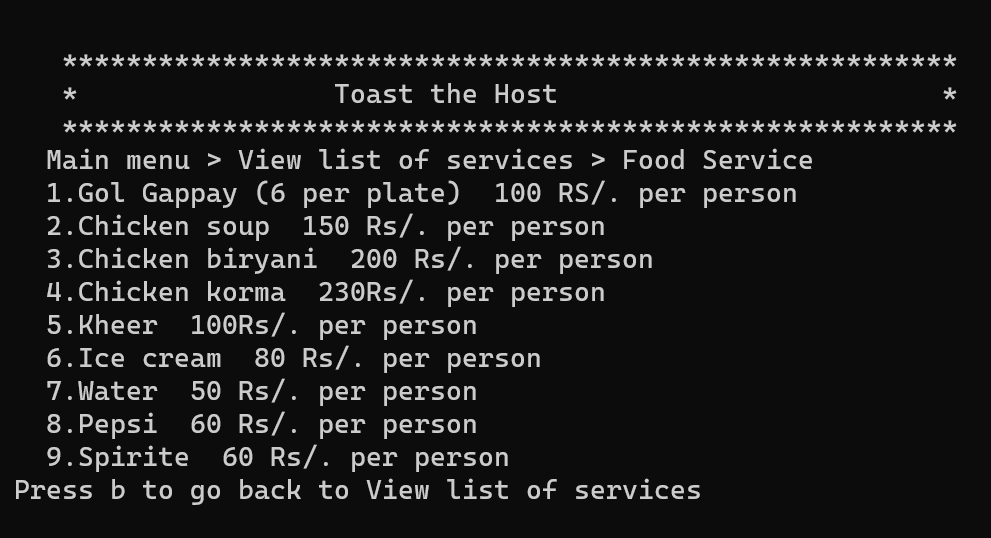


Figure 15: Food Service

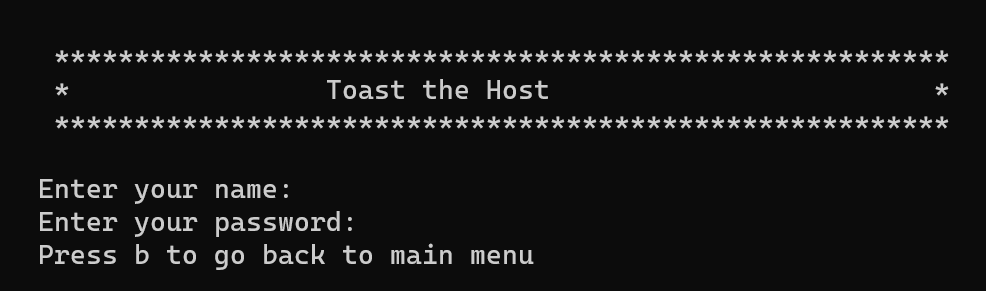


Figure 16: Customer login

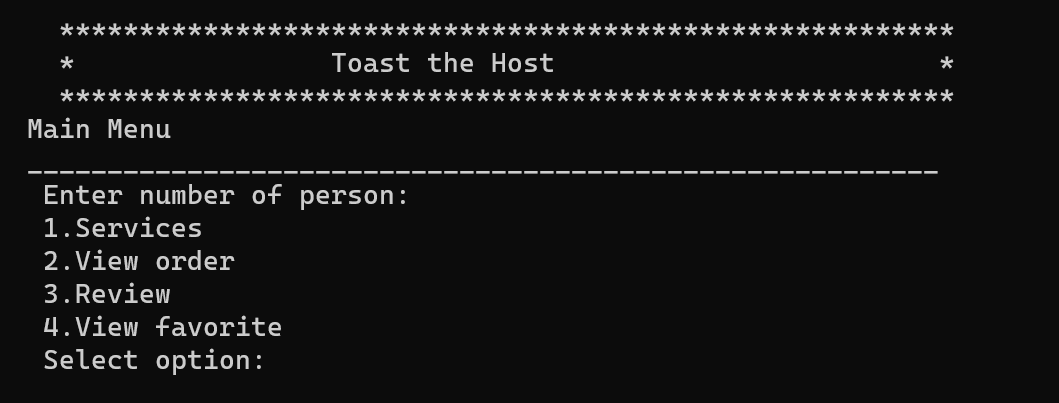


Figure 17: Customer main menu



Figure 18: Review

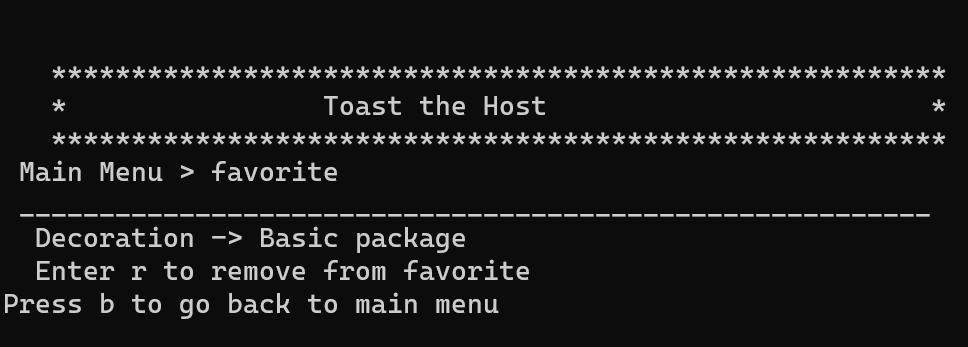


Figure 19: Favorite

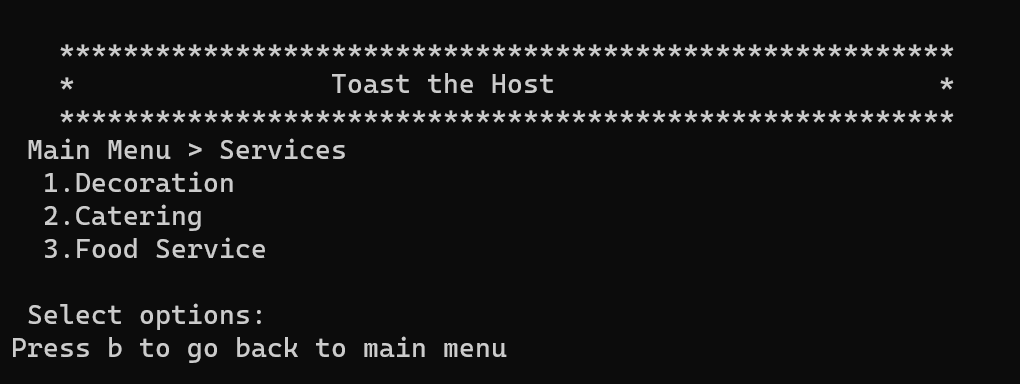


Figure 20: Services

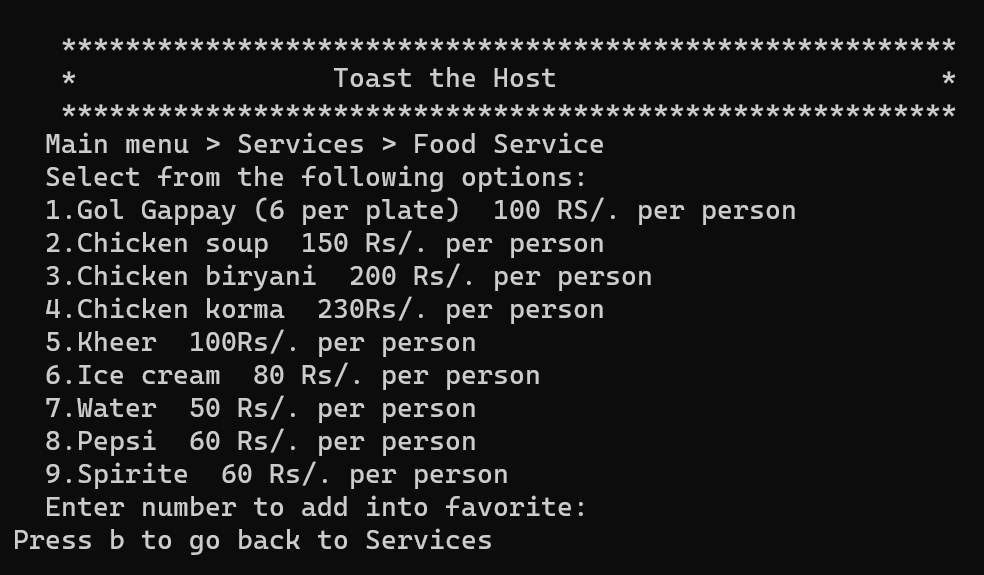


Figure 21: Food Service

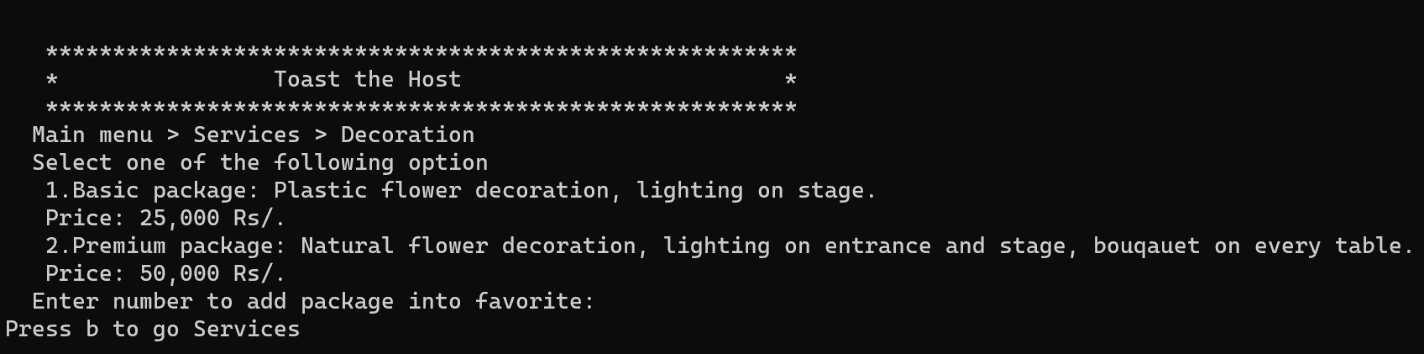


Figure 22: Decoration

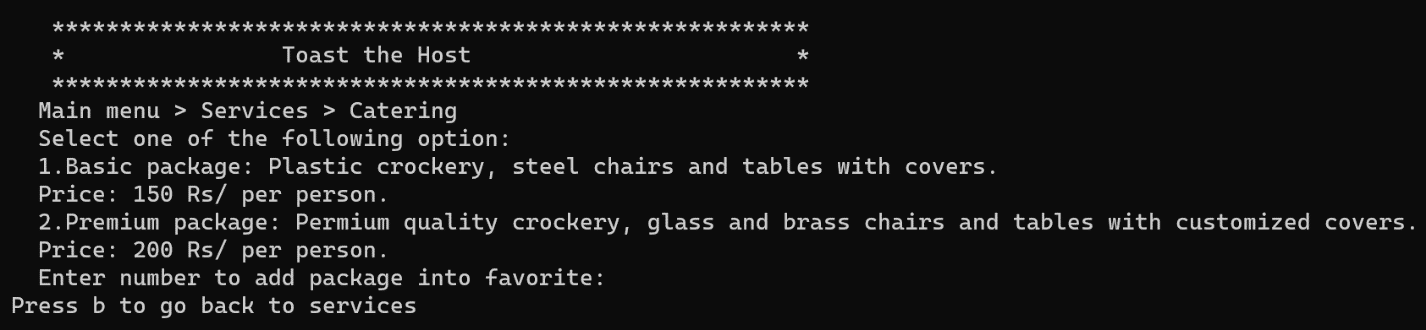


Figure 23: Catering

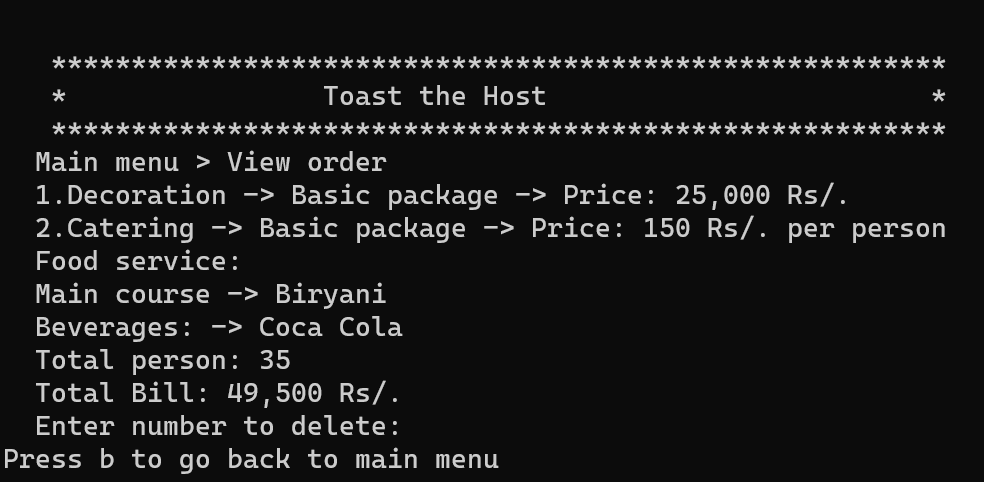


Figure 24: View order

# Data Structure

*// Admin credential Arrays*

**string** adminNameArray[5] = {"mano"};

**string** adminPassArray[5] = {"1234"};

*// Customer credentials Arrays*

**string** customerNameArray[10] = {"sheela"};

**string** customerPassArray[10] = {"1234"};

*// Package Arrays*

**string** food[15] = {"Gol Gappay", "Chicken Soup", "Chicken Biryani", "Chicken Korma", "Kheer", "Ice Cream", "Water", "Pepsi", "Sprite"};

int price[15] = {100, 150, 200, 230, 100, 80, 50, 60, 60};

**string** decor2[] = {"Basic package: Plastic flower decoration, lighting on stage", "Premium package: Permium quality crockery, glass and brass chairs and tables with customized covers."};

int priceDecor[] = {25000, 50000};

**string** catering2[2] = {"Basic package: Plastic crockery, steel chairs and tables with covers.", "Premium package: Permium quality crockery, glass and brass chairs and tables with customized covers."};

int cateringPrice[2] = {150, 200};

*// Order Arrays*

int decorOrderAr[20] = {};

int caterOrderAr[20] = {};

**string** foodOrderAr[20] = {};

int foodOrderPriceAr[20] = {};

int totalPersonsOrderAr[20] = {};

**string** foodNameCart[15] = {};

int foodPriceCart[15] = {};

*// Reviews Arrays*

**string** nameReviewsAr[35] = {};

**string** textReviewsAr[35] = {};

*// Favorite Array*

int decorFav[10] = {};

int cateringFav[10] = {};

int foodFav[10] = {};

# Functions Prototypes

*// Main header functions*

char **header**();

void **SeriveProviderPortal**();

void **CustomerPortal**();

*// Admin functions*

int **serviceProviderMenu**();

int **serviceMainMenu**();

int **serviceList**();

int **decor**();

int **catering**();

int **foodMenu**();

int **addFood**();

int **upService**();

int **upDateDecor**();

int **upDateCater**();

int **updateFoodItems**();

int **deleteFoodItems**();

int **ViewOrders**();

int **ViewReviews**();

int **AddServiceProvider**();

int **CalculateBalance**();

int **UpdatePassword**(**string** curPass, int idx);

*// Customer Functions*

int **customerServices**();

int **customerMainMenu**();

int **AddCustomer**();

int **customerMenu**();

int **customerDecor**();

int **customercatering**();

int **customerFoodMenu**();

int **review**();

int **viewFavorite**();

int **viewOrder**();

int **foodOrdering**(int price);

*// Store Data*

void **storeAdmin**(**string** adminName, **string** adminPassword);

void **storeCustomer**(**string** customerName, **string** customerPass);

void **loadCustomer**();

void **loadAdmin**();

# Work Flow

Main Menu

Customer

Service Provider

Services

Add food

View Order

Update Service

Write Review

Update food

View Favorite

Delete food

Log Out

View order

View Reviews

Add provider

Calculate balance

Update password

Log out

# Code

#include <iostream>

#include <fstream>

using namespace std;

// Admin credential Arrays

string adminNameArray[5] = {"mano"};

string adminPassArray[5] = {"1234"};

// Customer credentials Arrays

string customerNameArray[10] = {"sheela"};

string customerPassArray[10] = {"1234"};

// Package Arrays

string food[15] = {"Gol Gappay", "Chicken Soup", "Chicken Biryani", "Chicken Korma", "Kheer", "Ice Cream", "Water", "Pepsi", "Sprite"};

int price[15] = {100, 150, 200, 230, 100, 80, 50, 60, 60};

string decor2[] = {"Basic package: Plastic flower decoration, lighting on stage", "Premium package: Permium quality crockery, glass and brass chairs and tables with customized covers."};

int priceDecor[] = {25000, 50000};

string catering2[2] = {"Basic package: Plastic crockery, steel chairs and tables with covers.", "Premium package: Permium quality crockery, glass and brass chairs and tables with customized covers."};

int cateringPrice[2] = {150, 200};

// Order Arrays

int decorOrderAr[20] = {};

int caterOrderAr[20] = {};

string foodOrderAr[20] = {};

int foodOrderPriceAr[20] = {};

int totalPersonsOrderAr[20] = {};

// cart variables for customer

int decorCart = -1;

int caterCart = -1;

string foodNameCart[15] = {};

int foodPriceCart[15] = {};

int totalPersonsCart = 0;

int foodCount = 0;

int reviewOrder = 0;

int userCount = 0;

int customerCount = 0;

int reviewCount = 0;

// Reviews Arrays

string nameReviewsAr[35] = {};

string textReviewsAr[35] = {};

// Favorite Array

int decorFav[10] = {};

int cateringFav[10] = {};

int foodFav[10] = {};

int availableBalance = 0;

string adminName;

string adminPassword;

string customerName;

string customerPass;

string userNameReview;

string userReview;

// Main header functions

char header();

void SeriveProviderPortal();

void CustomerPortal();

// Admin functions

int serviceProviderMenu();

int serviceMainMenu();

int serviceList();

int decor();

int catering();

int foodMenu();

int addFood();

int upService();

int upDateDecor();

int upDateCater();

int updateFoodItems();

int deleteFoodItems();

int ViewOrders();

int ViewReviews();

int AddServiceProvider();

int CalculateBalance();

int UpdatePassword(string curPass, int idx);

// Customer Functions

int customerServices();

int customerMainMenu();

int AddCustomer();

int customerMenu();

int customerDecor();

int customercatering();

int customerFoodMenu();

int review();

int viewFavorite();

int viewOrder();

int foodOrdering(int price);

// Store Data

void storeAdmin(string adminName, string adminPassword);

void storeCustomer(string customerName, string customerPass);

void loadCustomer();

void loadAdmin();

void storeReview();

void loadReview();

main()

{

system("cls");

loadAdmin();

loadCustomer();

loadReview();

while (true)

{

char main = header();

if (main == 's')

{

SeriveProviderPortal();

}

else if (main == 'c')

{

CustomerPortal();

}

else

{

system("cls");

cout << "Please enter \"s\" to go to Admin portal or \"c\" to go to Customer portal. Thanks.\n";

}

}

}

char header()

{

char option;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* \*" << endl;

cout << "\* Toast the Host \*" << endl;

cout << "\* \*" << endl;

cout << "\* \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << "Enter s for Service Provider" << endl;

cout << "Enter c for Customer" << endl;

cout << endl;

cout << "Enter option: ";

cin >> option;

return option;

}

void SeriveProviderPortal()

{

int serviceMain;

int menu = serviceProviderMenu();

int adminIndex;

string loggedInPass;

if (menu == 1)

{

bool showOrnot = true;

while (showOrnot == true)

{

cout << "Enter your name: ";

cin >> adminName;

cout << "Enter your password: ";

cin >> adminPassword;

userCount++;

storeAdmin(adminName, adminPassword);

int count = 0;

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

{

if (adminName == adminNameArray[i] && adminPassword == adminPassArray[i])

{

loggedInPass = adminPassword;

adminIndex = i;

count++;

break;

}

}

if (count > 0)

{

serviceMain = serviceMainMenu();

showOrnot = false;

}

else

{

cout << "Wrong Credentials" << endl;

showOrnot = true;

}

}

while (true)

{

if (serviceMain == 1)

{

int listService = serviceList();

if (listService == 1)

{

int decoration = decor();

}

else if (listService == 2)

{

int cater = catering();

}

else if (listService == 3)

{

int food = foodMenu();

}

else if (listService == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 2)

{

int add = addFood();

if (add == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 3)

{

int upDateServices = upService();

if (upDateServices == 1)

{

int option;

int date = upDateDecor();

if (date == 1 || date == 2)

{

cout << "Enter new price for this package: ";

cin >> priceDecor[date - 1];

cout << "Price has been changed successfully for this service." << endl;

cout << endl;

cout << "Press 0 to Go back to main menu" << endl;

cin >> option;

if (option == 0)

{

{

serviceMain = serviceMainMenu();

}

}

}

}

else if (upDateServices == 2)

{

int option;

int dateCater = upDateCater();

if (dateCater == 1 || dateCater == 2)

{

cout << "Enter new price for this package: ";

cin >> priceDecor[dateCater - 1];

cout << "Price has been changed successfully for this service." << endl;

cout << endl;

cout << "Press 0 to Go back to main menu" << endl;

cin >> option;

if (option == 0)

{

{

serviceMain = serviceMainMenu();

}

}

}

}

}

else if (serviceMain == 4)

{

int backButton = updateFoodItems();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 5)

{

int backButton = deleteFoodItems();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 6)

{

int backButton = ViewOrders();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 7)

{

int backButton = ViewReviews();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 8)

{

int backButton = AddServiceProvider();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 9)

{

int backButton = CalculateBalance();

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 10)

{

int backButton = UpdatePassword(loggedInPass, adminIndex);

if (backButton == 0)

{

serviceMain = serviceMainMenu();

}

}

else if (serviceMain == 11)

{

system("cls");

return;

}

}

}

else

{

system("cls");

}

}

void CustomerPortal()

{

int serviceMain;

int menu = customerMenu();

string name;

string password;

if (menu == 1)

{

int backButton = AddCustomer();

if (backButton == 0)

{

CustomerPortal();

}

}

else if (menu == 2)

{

bool showOrnot = true;

while (showOrnot == true)

{

cout << "Enter your name: ";

cin >> name;

cout << "Enter your password: ";

cin >> password;

int count = 0;

for (int i = 0; i < 10; i++)

{

if (name == customerNameArray[i] && password == customerPassArray[i])

{

count++;

break;

}

}

if (count > 0)

{

serviceMain = customerMainMenu();

showOrnot = false;

}

else

{

cout << "Wrong Credentials" << endl;

showOrnot = true;

}

}

while (true)

{

if (serviceMain == 1)

{

int option = customerServices();

if (option == 0)

{

serviceMain = customerMainMenu();

}

else if (option == 1)

{

int ans = customerDecor();

if (ans == 1 || ans == 2)

{

int opt;

decorCart = ans;

cout << "Item added to cart successfuly!" << endl;

cout << "Press 0 to Go back to main menu: ";

cin >> opt;

}

else if (ans == 11 || ans == 22)

{

if (ans == 11)

{

ans = 1;

}

else if (ans == 22)

{

ans = 2;

}

int opt;

for (int idx = 0; idx <= 9; idx++)

{

if (decorFav[idx] == 0)

{

decorFav[idx] = ans;

cout << "Item added to favorite successfuly!" << endl;

cout << "Press 0 to Go back to main menu: ";

cin >> opt;

break;

}

}

}

}

else if (option == 2)

{

int ans = customercatering();

if (ans == 1 || ans == 2)

{

int opt;

caterCart = ans;

cout << "Item added to cart successfuly!\nPress 0 to Go back to main menu: ";

cin >> opt;

}

else if (ans == 11 || ans == 22)

{

int opt;

for (int idx = 0; idx <= 9; idx++)

{

if (ans == 11)

{

ans = 1;

}

else if (ans == 22)

{

ans = 2;

}

if (cateringFav[idx] == 0)

{

cateringFav[idx] = ans;

cout << "Item added to favorite successfuly!" << endl;

cout << "Press 0 to Go back to main menu: ";

cin >> opt;

break;

}

}

}

}

else if (option == 3)

{

int back = customerFoodMenu();

if (back == 0)

{

serviceMain = customerMainMenu();

}

}

}

else if (serviceMain == 2)

{

int orderFun = viewOrder();

if (orderFun == 0)

{

serviceMain = customerMainMenu();

}

else

{

totalPersonsCart = orderFun;

int placed = -1;

int foodPrice = 0;

int foodRaayPrice = 0;

int caterPrice = 0;

int decorPrice = 0;

if (decorCart != -1)

{

decorPrice = priceDecor[decorCart - 1];

}

for (int idx = 0; idx < 15; idx++)

{

foodRaayPrice = foodPriceCart[idx] + foodRaayPrice;

}

foodPrice = totalPersonsCart \* foodRaayPrice;

cout << "Total price for food: " << foodPrice;

if (caterCart != -1)

{

caterPrice = totalPersonsCart \* cateringPrice[caterCart - 1];

cout << "\nTotal price for catering: " << caterPrice;

}

cout << "\nGrand Total including food, decoration and catering: " << caterPrice + foodPrice + decorPrice;

int back = foodOrdering(foodRaayPrice);

if (back == 1)

{

cout << "placed";

}

serviceMain = customerMainMenu();

}

}

else if (serviceMain == 3)

{

int back = review();

if (back == 0)

{

serviceMain = customerMainMenu();

}

}

else if (serviceMain == 4)

{

int back = viewFavorite();

if (back == 0)

{

serviceMain = customerMainMenu();

}

}

else if (serviceMain == 5)

{

system("cls");

return;

}

}

}

else if (menu == 3)

{

system("cls");

}

}

int foodOrdering(int price)

{

int placed;

cout << "\nPress 1 to place order: ";

cin >> placed;

cout << "Order Placed successfully";

for (int idx = 0; idx < 20; idx++)

{

if (decorOrderAr[idx] == 0)

{

decorOrderAr[idx] = decorCart;

caterOrderAr[idx] = caterCart;

string food = "";

for (int j = 0; idx < 15; idx++)

{

if (foodNameCart[j] != "")

{

food = food + ", " + foodNameCart[j];

}

}

foodOrderAr[idx] = food;

foodOrderPriceAr[idx] = price;

totalPersonsOrderAr[idx] = totalPersonsCart;

break;

}

}

}

int serviceProviderMenu()

{

system("cls");

int option;

cout << endl;

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1. SignIn with your Credentials" << endl;

cout << "2. Go back to home page" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int customerMenu()

{

system("cls");

int option;

cout << endl;

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1. Create Account" << endl;

cout << "2. SignIn with your Credentials" << endl;

cout << "3. Go back to home page" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int serviceMainMenu()

{

system("cls");

int option;

cout << endl;

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " 1. View list of services" << endl;

cout << " 2. Add food to menu" << endl;

cout << " 3. Update services" << endl;

cout << " 4. Update food menu" << endl;

cout << " 5. Delete food items" << endl;

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

cout << " 7. View reviews" << endl;

cout << " 8. Add service provider" << endl;

cout << " 9. Calculate balance" << endl;

cout << " 10. Update password" << endl;

cout << " 11. Logout" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int customerMainMenu()

{

system("cls");

int option;

cout << endl;

cout << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu" << endl;

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << " 1. Services" << endl;

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

cout << " 3. Write Review" << endl;

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

cout << " 5. Logout" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int serviceList()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > View list of services" << endl;

cout << " 1. Decoration" << endl;

cout << " 2. Catering" << endl;

cout << " 3. Food Service" << endl;

cout << " Press 0 to Go back to main menu" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int decor()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Services > Decoration" << endl;

for (int idx = 0; idx < 2; idx++)

{

cout << decor2[idx] << " " << endl

<< priceDecor[idx] << " Rs/." << endl;

}

cout << endl;

cout << "Press 0 to go back to View list of services: ";

cin >> option;

return option;

}

int catering()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Services > Catering" << endl;

for (int idx = 0; idx < 2; idx++)

{

cout << catering2[idx] << " " << endl

<< cateringPrice[idx] << "Rs/. per person" << endl;

}

cout << endl;

cout << "Press 0 to go back to View list of services: ";

cin >> option;

return option;

}

int foodMenu()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > View list of services > Food Service\n"

<< endl;

int lengthOfFoodArray = sizeof(food) / sizeof(string);

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

{

if (food[idx] != "")

{

cout << idx + 1 << "." << food[idx] << " " << price[idx] << "Rs/person \n";

}

}

cout << "\n Press 0 to go back to View list of services: ";

cin >> option;

return option;

}

int addFood()

{

system("cls");

int option;

int option1;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Add food to menu" << endl;

int lengthOfFoodArray = sizeof(food) / sizeof(string);

if (food[lengthOfFoodArray - 1] != "")

{

cout << "Maximum limit reached to add items. Please delete items from food menu to add more\n";

}

else

{

cout << "How many things do you want to add? ";

cin >> option1;

for (int idx = 0; idx < option1; idx++)

{

int foodPrice;

string foodName;

cout << "Enter dish name: ";

cin >> foodName;

cout << "Enter price for dish: ";

cin >> foodPrice;

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

{

if (food[idx] == "")

{

food[idx] = foodName;

price[idx] = foodPrice;

break;

}

}

if (food[lengthOfFoodArray - 1] != "")

{

cout << "Maximum limit reached to add items. Please delete items from food menu to add more \n";

break;

}

}

}

cout << "Press 0 to go back to View list of services: ";

cin >> option;

return option;

}

int upService()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Update services\nSelect one of the following to update services." << endl;

cout << " 1. Decoration" << endl;

cout << " 2. Catering" << endl;

cout << " Press 0 to Go back to main menu" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int upDateDecor()

{

int option;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Update services > Decoration" << endl;

cout << " Select one of the following option to update" << endl;

cout << " 1.Basic package" << endl;

cout << " 2.Premium package\n";

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int upDateCater()

{

int option;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Update services > Catering" << endl;

cout << " Select one of the following option to update" << endl;

cout << " 1.Basic package" << endl;

cout << " 2.Premium package\n";

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int updateFoodItems()

{

int selectedIdx;

int input;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Update food menu" << endl;

cout << " Select one of the following items to update" << endl;

int lengthOfFoodArray = sizeof(food) / sizeof(string);

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

{

if (food[idx] != "")

{

cout << idx + 1 << ". " << food[idx] << " " << price[idx] << "Rs/person \n";

}

}

cout << endl;

cout << "Enter the Option Number: ";

cin >> selectedIdx;

cout << "Enter new price for this item: ";

cin >> price[selectedIdx - 1];

cout << "Price has been updated successfully for food item: " << food[selectedIdx - 1] << "\n";

cout << endl;

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int deleteFoodItems()

{

int selectedIdx;

int input;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Delete food items" << endl;

cout << " Select one of the following items to delete" << endl;

int lengthOfFoodArray = sizeof(food) / sizeof(string);

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

{

if (food[idx] != "")

{

cout << idx + 1 << ". " << food[idx] << " " << price[idx] << "Rs/person \n";

}

}

cout << endl;

cout << "Enter the Option Number: ";

cin >> selectedIdx;

string selectedItem = food[selectedIdx - 1];

for (int j = selectedIdx - 1; j <= (lengthOfFoodArray - 1); j++)

{

food[j] = food[j + 1];

price[j] = price[j + 1];

if (j == lengthOfFoodArray - 1)

{

food[j] = "";

}

}

cout << food[selectedIdx - 1] << " has been deleted successfully from food menu."

<< "\n";

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int ViewOrders()

{

int input;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > View Orders" << endl;

int lengthOfOrderArray = sizeof(decorOrderAr) / sizeof(int);

for (int j = 0; j < 20; j++)

{

int totalPrice = 0;

cout << " Order no." << j + 1 << ":\n";

cout << " \_\_\_\_\_\_\_\_\_\_\_\_" << endl;

if (decorOrderAr[j] != -1)

{

cout << " ->Decoration: " << decor2[decorOrderAr[j] - 1] << "\n Price: " << priceDecor[decorOrderAr[j] - 1] << "\n";

}

if (caterOrderAr[j] != -1)

{

cout << " ->Catering: " << catering2[caterOrderAr[j] - 1] << "\n Price: " << cateringPrice[caterOrderAr[j] - 1] << "\n";

}

if (foodOrderAr[j] != "")

{

cout << " ->Food Items: " << foodOrderAr[j] << "\n";

}

cout << " Total no of persons: " << totalPersonsOrderAr[j] << "\n\n";

cout << " Total payable: " << totalPrice << "\n"

<< endl;

}

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int ViewReviews()

{

int input;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > View Reviews" << endl;

int lengthOfOrderArray = sizeof(nameReviewsAr) / sizeof(string);

for (int j = 0; j < lengthOfOrderArray; j++)

{

if (nameReviewsAr[j] != "")

{

cout << " " << j + 1 << ". Name: " << nameReviewsAr[j] << "\n";

cout << " Feedback: " << textReviewsAr[j] << "\n\n";

}

}

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int AddServiceProvider()

{

int input;

string name;

string pass;

int count = 0;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Add service provider" << endl;

cout << "Enter service provider name: ";

cin >> name;

cout << "Enter password: ";

cin >> pass;

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

{

if (adminNameArray[i] == "")

{

adminNameArray[i] = name;

adminPassArray[i] = pass;

count++;

break;

}

}

if (count == 0)

{

cout << "You cannot add more service providers. Maximum limit reached.";

}

else

{

cout << "Service provider has been added successfully!\n";

}

cout << endl;

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int CalculateBalance()

{

int input;

int pass;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Calculate balance" << endl;

cout << "Available balance: " << availableBalance << " Rs.";

cout << endl;

cout << "Enter amount to add (in Rs): ";

cin >> pass;

availableBalance = availableBalance + pass;

cout << "Available balance: " << availableBalance << " Rs." << endl;

cout << endl;

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int UpdatePassword(string curPass, int idx)

{

int input;

string currentPass;

string newPass;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Update password" << endl;

bool boolean = false;

while (boolean == false)

{

cout << "Enter current password: ";

cin >> currentPass;

cout << "Enter new password: ";

cin >> newPass;

if (curPass == currentPass)

{

adminPassArray[idx] = newPass;

cout << "Password has been updated successfully!\n";

boolean = true;

}

else

{

cout << "Current password is not correct. Please try again.\n";

boolean = false;

}

}

cout << "Press 0 to go back to Main menu: ";

cin >> input;

return input;

}

int customerServices()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main Menu > Services" << endl;

cout << " 1. Decoration" << endl;

cout << " 2. Catering" << endl;

cout << " 3. Food Service" << endl;

cout << " Press 0 to Go back to main menu" << endl;

cout << endl;

cout << "Enter the Option Number: ";

cin >> option;

return option;

}

int customerDecor()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Services > Decoration" << endl;

for (int idx = 0; idx < 2; idx++)

{

cout << idx + 1 << "." << decor2[idx] << " " << endl

<< priceDecor[idx] << " Rs/." << endl;

}

cout << endl;

cout << "Enter 1 or 2 to enter item in cart\nEnter 11 to make number 1 favorite or 22 to make number 2 favorite\n";

cin >> option;

return option;

}

int AddCustomer()

{

int input;

int count = 0;

system("cls");

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Create Account" << endl;

cout << "Enter your name: ";

cin >> customerName;

cout << "Enter password: ";

cin >> customerPass;

customerCount++;

storeCustomer(customerName, customerPass);

for (int i = 0; i < 10; i++)

{

if (customerNameArray[i] == "")

{

customerNameArray[i] = customerName;

customerPassArray[i] = customerPass;

count++;

break;

}

}

if (count == 0)

{

cout << "Account cannot be created at this moment. Sorry for inconvenience.";

}

else

{

cout << "Account has been added successfully!\n";

}

cout << endl;

cout << "Press 0 to go back: ";

cin >> input;

return input;

}

int customercatering()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Services > Catering" << endl;

for (int idx = 0; idx < 2; idx++)

{

cout << catering2[idx] << " " << endl

<< cateringPrice[idx] << "Rs/. per person" << endl;

}

cout << endl;

cout << "Enter 1 or 2 to enter item in cart\nEnter 11 to make number 1 favorite or 22 to make number 2 favorite\n";

cin >> option;

return option;

}

int customerFoodMenu()

{

system("cls");

int option = -1;

int option1;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Services > Food Service\n"

<< endl;

int lengthOfFoodArray = sizeof(food) / sizeof(string);

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

{

if (food[idx] != "")

{

cout << idx + 1 << "." << food[idx] << " " << price[idx] << "Rs/person \n";

}

}

cout << endl;

cout << "Enter 0 to finish ordering\n";

while (option != 0)

{

cout << "Enter option number to add item in cart: ";

cin >> option;

foodPriceCart[foodCount] = price[option - 1];

foodNameCart[foodCount] = food[option - 1];

foodCount++;

}

return option;

}

int review()

{

system("cls");

int option;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \* Toast the Host \*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " Main menu > Write review\n";

cout << "Enter your name: ";

cin >> userNameReview;

cout << "Enter review: ";

cin >> userReview;

reviewCount++;

storeReview();

for (int idx = 0; idx <= 35; idx++)

{

if (nameReviewsAr[idx] == "")

{

nameReviewsAr[idx] = userNameReview;

textReviewsAr[idx] = userReview;

cout << "Review sent successfully!" << endl;

cout << "Press 0 to back to menu: ";

cin >> option;

break;

}

}

reviewOrder++;

return option;

}

int viewFavorite()

{

system("cls");

int option;

for (int idx = 0; idx < 10; idx++)

{

if (decorFav[idx] != 0)

{

cout << " ->Decoration: " << decor2[decorFav[idx] - 1] << endl;

}

if (cateringFav[idx] != 0)

{

cout << " ->Catering: " << catering2[cateringFav[idx] - 1] << endl;

}

}

cout << "Press 0 to go back to main menu: ";

cin >> option;

return option;

}

int viewOrder()

{

system("cls");

int option;

if (decorCart != -1)

{

cout << "Decoration: " << decor2[decorCart - 1] << endl;

}

if (caterCart != -1)

{

cout << "Catering: " << catering2[caterCart - 1] << endl;

}

cout << "Food Items:\n";

for (int idx = 0; idx < 15; idx++)

{

if (foodNameCart[idx] != "")

{

cout << foodNameCart[idx] << " Price: " << foodPriceCart[idx] << endl;

}

}

cout << "\nEnter no of persons to calculate total price or Press 0 to go back to main menu: ";

cin >> option;

return option;

}

void storeAdmin(string adminName, string adminPassword)

{

system("cls");

fstream file;

file.open("admin.txt", ios::app);

file << adminName << endl;

file << adminPassword << endl;

file.close();

}

void storeCustomer(string customerName, string customerPass)

{

system("cls");

fstream file;

file.open("customer.txt", ios::app);

file << customerName << endl;

file << customerPass << endl;

file.close();

}

void loadAdmin()

{

system("cls");

string adminName;

string adminPassword;

fstream file;

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

while(getline(file, adminName) && getline(file, adminPassword))

{

adminNameArray[userCount] = adminName;

adminPassArray[userCount] = adminPassword;

userCount++;

}

file.close();

}

void loadCustomer()

{

system("cls");

string customerName;

string customerPass;

fstream file;

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

while(getline(file, customerName) && getline(file, customerPass))

{

customerNameArray[customerCount] = customerName;

customerPassArray[customerCount] = customerPass;

userCount++;

}

file.close();

}

void storeReview()

{

system("cls");

fstream file;

file.open("review.txt", ios::app);

file << userNameReview << endl;

file << userReview << endl;

file.close();

}

void loadReview()

{

system("cls");

string userNameReview;

string userReview;

fstream file;

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

while(getline(file, userNameReview) && getline(file, userReview))

{

nameReviewsAr[reviewCount] = userNameReview;

textReviewsAr[reviewCount] = userReview;

userCount++;

}

file.close();

}

# Weakness

* It should be easier to understand.

# Future Direction

* User Friendly interface
* Better design

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** | |
| Documentation  Formatting **Grade:** | All the documentation meets all the criteria. | | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available | |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | | | |
| Documentation Contents **Grade:** | Documentation includes all of the criteria. | | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. | |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | | | |
| Project  Complexity **Grade:** | Project has at least 2 user’s types and each user  has at least 5 functionalities. | | Project complexity meet 80%  criteria given in extensive evidence | Project complexity meet 50%  criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence | |
| Code Style **Grade:** | All Code style criteria is followed | | All code style criteria followed but some  improvements required | lot of improvements required in coding style. | **Did not follow** code style, | |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added. | | | | | | |
| Code  Documentation Mapping **Grade:** | Code and documentation is synchronized. | | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. | |
| Data Structure  (Arrays) **Grade:** | Data structure is sufficient for the project requirements | | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. | |
| Modularity **Grade:** | Meet all Modularity criteria | | Meet all Modularity criteria  but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. | |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | | | |
| Validations **Grade:** | Validations on all number type inputs are applied | | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used | |
| File Handling **Grade:** | Separate files for separate data. Data in csv format | | File handing require some improvements | File handing require a lot of improvements | Not implemented | |
| Aesthetics of the  User Interface **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | | UI require some improvements | UI require a lot of improvements | Not implemented | |
| Presentation and  Demo  **Grade:** | Presentation and Demo was 100% working | | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working | |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | | Student has good understand but some place he does not  know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. | |
| **Checked by:** | |  | | | |
| **Comments:** | |  | | | |
|  | **A-Extensive Evidence** | | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** | |
| Documentation  Formatting **Grade:** | All the documentation meets all the criteria. | | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available | |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | | | |
| Documentation Contents **Grade:** | Documentation includes all of the criteria. | | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. | |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | | | |
| Project  Complexity **Grade:** | Project has at least 2 user’s types and each user  has at least 5 functionalities. | | Project complexity meet 80%  criteria given in extensive evidence | Project complexity meet 50%  criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence | |
| Code Style **Grade:** | All Code style criteria is followed | | All code style criteria followed but some  improvements required | lot of improvements required in coding style. | **Did not follow** code style, | |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added. | | | | | | |
| Code  Documentation Mapping **Grade:** | Code and documentation is synchronized. | | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. | |
| Data Structure  (Arrays) **Grade:** | Data structure is sufficient for the project requirements | | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. | |
| Modularity **Grade:** | Meet all Modularity criteria | | Meet all Modularity criteria  but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. | |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | | | |
| Validations **Grade:** | Validations on all number type inputs are applied | | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used | |
| File Handling **Grade:** | Separate files for separate data. Data in csv format | | File handing require some improvements | File handing require a lot of improvements | Not implemented | |
| Aesthetics of the  User Interface **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | | UI require some improvements | UI require a lot of improvements | Not implemented | |
| Presentation and  Demo  **Grade:** | Presentation and Demo was 100% working | | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working | |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | | Student has good understand but some place he does not  know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. | |
| **Checked by:** | |  | | | |
| **Comments:** | |  | | | |