```
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <ctime>
#include <cstdlib>
#include <unordered_map> // For storing credentials
using namespace std;

class FoodItem {
public:
    std::string name;
    double price;
    double discount; // Adding discount field

    FoodItem(const std::string& itemName, double itemPrice, double itemDiscount = 0.0)
        : name(itemName), price(itemPrice), discount(itemDiscount) {}

    double discountedPrice() const {
        return price * (1.0 - discount);
    }

    void applyDiscount(double newDiscount) {
        discount = newDiscount;
    }
};

class Restaurant {
public:
    std::string name;
    std::vector<FoodItem> menu;
    int rating;  // Adding rating as an integer

    Restaurant(const std::string& restaurantName, int restaurantRating)
        : name(restaurantName), rating(restaurantRating) {}

    void addToMenu(const FoodItem& item) {
        menu.push_back(item);
    }

    void updateDiscount(const std::string& itemName, double newDiscount) {
        for (auto& item : menu) {
            if (item.name == itemName) {
                item.applyDiscount(newDiscount);
                break;
            }
        }
    }
```

```
};

class Customer {
public:
    std::string name;
    std::string address;  // Adding customer address for delivery
    std::vector<FoodItem> cart;

    Customer(const std::string& customerName, const std::string& customerAddress)
        : name(customerName), address(customerAddress) {}

    void addToCart(const FoodItem& item) {
        cart.push_back(item);
    }

    double calculateTotal() const {
        double total = 0.0;
        for (const auto& item : cart) {
            total += item.discountedPrice(); // Calculate total with discounted price
        }
        return total;
    }
};

// Function to generate a random captcha
std::string generateCaptcha() {
    const std::string charSet =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    const int length = 6;
    std::string captcha;
    srand(time(0));
    for (int i = 0; i < length; ++i) {
        captcha += charSet[rand() % charSet.length()];
    }
    return captcha;
}

// Function to simulate user database with credentials
std::unordered_map<std::string, std::string> userDatabase = {
    {"user1", "password1"},
    {"user2", "password2"},
    // Add more users if needed
};

bool authenticateUser(const std::string& username, const std::string& password) {
    auto it = userDatabase.find(username);
    if (it != userDatabase.end() && it->second == password) {
        return true;
    }
    return false;
}

int main() {
    std::string username;
    std::string password;

    std::cout << "          🙏🙏Welcome to Restaurant Ordering System🙏🙏\n";
    std::cout << "              -----------------------------------\n";
    std::cout << "\nPlease sign in to continue:\n";

    int attempts = 0;
    do {
        if (attempts <3) {

            std::cout << "Username: ";
            std::cin >> username;

            std::cout << "Password: ";
            std::cin >> password;

            if(username=="abc" && password=="123456")
            {
                std::cout << "\nAuthentication successful. Welcome, " << username << "!\n";
                break;
            }
            else{
                std::cout << "Incorrect username or password. Please try again.\n";
            }

            attempts++;
        }
        else{
             std::cerr << "Too many failed attempts. Exiting.\n";
            break;
        }
    } while (!authenticateUser(username, password) && attempts < 3);


    // Create multiple restaurants
    std::vector<Restaurant> restaurants;

    restaurants.emplace_back("Adones Fast Food", 4);
    restaurants.back().addToMenu(FoodItem("Pizza🍕🍕🍕", 240.0, 0.0)); // Initial discount is
0%
    restaurants.back().addToMenu(FoodItem("Burger🍔🍔🍔", 160.0, 0.0)); // Initial discount
is 0%
```

```
    restaurants.back().addToMenu(FoodItem("Sandwich🥪🥪🥪", 100.0, 0.0)); // Initial
discount is 0%

    restaurants.emplace_back("Gourmet Delights", 5);
    restaurants.back().addToMenu(FoodItem("Steak🥩🥩🥩 ", 350.0, 0.0)); // Initial discount
is 0%
    restaurants.back().addToMenu(FoodItem("Salad🥗🥗🥗", 180.0, 0.0)); // Initial discount is
0%
    restaurants.back().addToMenu(FoodItem("Pasta🥧🥧🥧", 200.0, 0.0)); // Initial discount is
0%

    // Get customer details including address
    std::string customerName;
    std::string customerAddress;
    std::cout << "Enter customer name: ";
    std::cin.ignore(); // Ignore newline character left in buffer
    std::getline(std::cin, customerName);

    std::cout << "Enter customer address: ";
    std::getline(std::cin, customerAddress);

    Customer customer(customerName, customerAddress);

    // Display available restaurants and their ratings
    std::cout << "\nAvailable restaurants:\n";
    for (size_t i = 0; i < restaurants.size(); ++i) {
        std::cout << i + 1 << ". " << restaurants[i].name << " (Rating: " << restaurants[i].rating
<< ")\n";
    }

    // Select a restaurant
    int restaurantChoice;
    std::cout << "Enter restaurant number (1-" << restaurants.size() << "): ";
    std::cin >> restaurantChoice;
    if (restaurantChoice < 1 || restaurantChoice > restaurants.size()) {
        std::cerr << "Invalid choice. Exiting.\n";
        return 1;
    }

    // Display available food items at selected restaurant
    std::cout << "\nAvailable food items at " << restaurants[restaurantChoice - 1].name <<
":\n";
    for (size_t i = 0; i < restaurants[restaurantChoice - 1].menu.size(); ++i) {
        const FoodItem& item = restaurants[restaurantChoice - 1].menu[i];
        std::cout << i + 1 << ". " << item.name << " - $" << item.price;
        if (item.discount > 0.0) {
            std::cout << " (Discounted Price: $" << item.discountedPrice() << ")";
        }
        std::cout << "\n";
    }

    // Take input for food items to be ordered
    int choice;
    do {
        std::cout << "Enter food item number (1-" << restaurants[restaurantChoice -
1].menu.size() << ", 0 to finish): ";
        std::cin >> choice;
        if (choice >= 1 && choice <= restaurants[restaurantChoice - 1].menu.size()) {
            customer.addToCart(restaurants[restaurantChoice - 1].menu[choice - 1]);
        }
    } while (choice != 0);

    // Show current amount and estimated delivery time
    double totalAmount = customer.calculateTotal();
    std::cout << "\nTotal amount: $" << totalAmount << "\n";
    std::cout << "Estimated delivery time: 40 minutes\n";

    // Generate a captcha for confirmation
    std::string generatedCaptcha = generateCaptcha();
    std::string enteredCaptcha;
    std::cout << "\nTo confirm your order, please enter the following captcha: " <<
generatedCaptcha << "\n";
    std::cout << "Enter captcha (case-sensitive): ";
    std::cin >> enteredCaptcha;

    // Validate captcha
    if (enteredCaptcha != generatedCaptcha) {
        std::cerr << "Captcha verification failed. Order canceled.\n";
        return 1;
    }

    // Payment options
    std::cout << "\nChoose payment method:\n";
    std::cout << "1. UPI\n";
    std::cout << "2. Debit Card\n";
    std::cout << "3. Credit Card\n";
    int paymentChoice;
    std::cout << "Enter your choice: ";
    std::cin >> paymentChoice;

    // Simulate payment processing
    bool paymentSuccessful = false;
    switch (paymentChoice) {
        case 1:
            std::cout << "Processing UPI payment...⏳⏳\n";
            // Simulate processing time
```

```
      paymentSuccessful = true;
      cout<<"\nPayment is successfully done✅✅\n";
      break;
    case 2:
      std::cout << "Processing Debit Card payment...\n";
      // Simulate processing time
      paymentSuccessful = true;
      cout<<"\nPayment is successfully done✅✅\n";
      break;
    case 3:
      std::cout << "Processing Credit Card payment...\n";
      // Simulate processing time
      paymentSuccessful = true;
      cout<<"\nPayment is successfully done✅✅\n";
      break;
    default:
      std::cout << "Invalid choice. Payment failed.\n";
      break;
  }

  // Print receipt
  std::cout << "\n---------------------- Receipt ----------------------\n";
  std::cout << std::left << std::setw(25) << "Customer Name: " << customer.name << "\n";
  std::cout << std::left << std::setw(25) << "Customer Address: " << customer.address <<
"\n";
  std::cout << std::left << std::setw(25) << "Restaurant Name: " <<
restaurants[restaurantChoice - 1].name << "\n";
  std::cout << "-----------------------------------------------------\n";
  std::cout << std::left << std::setw(25) << "Ordered Items:\n";
  for (const auto& item : customer.cart) {
    std::cout << std::left << std::setw(25) << item.name << " $" << item.discountedPrice()
<< "\n";
  }
  std::cout << "-----------------------------------------------------\n";
  std::cout << std::left << std::setw(25) << "Total Amount: " << "$" << totalAmount << "\n";
  std::cout << std::left << std::setw(25) << "Estimated Delivery: " << "40 minutes\n";
  std::cout << "-----------------------------------------------------\n";
  std::cout << std::left << std::setw(25) << "Payment Status: ";
  if (paymentSuccessful) {
    std::cout << "Successful✔✔\n";
  } else {
    std::cout << "Failed\n";
  }
  std::cout << "-----------------------------------------------------\n";
  cout<<"\nHelpdesk-\nBe free to call on any query regarding order details 9352339642";
  cout<<"        \nTHANK YOU FOR VISTING";
```

```
    return 0;
}
```