peter :

bcm-05-0106/2022.

SRS DOCUMENT, FLOWCHART, DFDS, UML DIAGRAM AND SOURCE CODE.

Software requirements document for a restaurant point of sale system.

INTRODUCTION.

<-> A restaurant point of sale system is a software created to make the running of the restaurant operations easier. It is used for;

* Order management.
* Menu management.
* Billing and payment.
* Inventory management.
* Reporting.

1. FUNCTIONAL REQUIREMENTS.

a) User management.

=> The software system should accommodate a variety of user roles such as manager, chef and cashier.

=> The user should also be able to access the system according to duties allocated to them.

b) Menu management.

=> The system should enable menu item creation and management by the restaurant employees.

=> The menu should include items such as name price description and categorize items such as appetizers, main courses and dessert.

c) Order management.

=> The system should enable staff to take customer orders and add them to the system.

=> Orders should include details like table number, items ordered, quantity, special meals among others.

d) Billing and payment.

=> The system should calculate and give the bill of ordered items.

=> It should also support various payment methodS example cash, mobile payment and credit cards then after it should give receipts for customers.

e) Inventory management.

=> The system should;

- provide a lot of any item running low.

- follow inventory levels of menu items.

f) Reporting.

=> The system should generate reports on various aspects example sales, menu and others.

=> The report should be available also in various formats such as pdf, word document.

2. NON-FUNCTIONAL REQUIREMENTS.

a) Usability.

=> The system should have an intuitive and user-friendly interface and also provide customers with feedback and clear directions.

b) Performance.

=> The system should respond to user inputs in a timely manner.

=> It should manage a high amount of concurrent users and orders efficiently.

c) Security.

=> The system should ensure the confidentiality and integrity of user data and also provide access control according to role of the user to restrict unauthorized access to sensitive functionality.

d) Reliability.

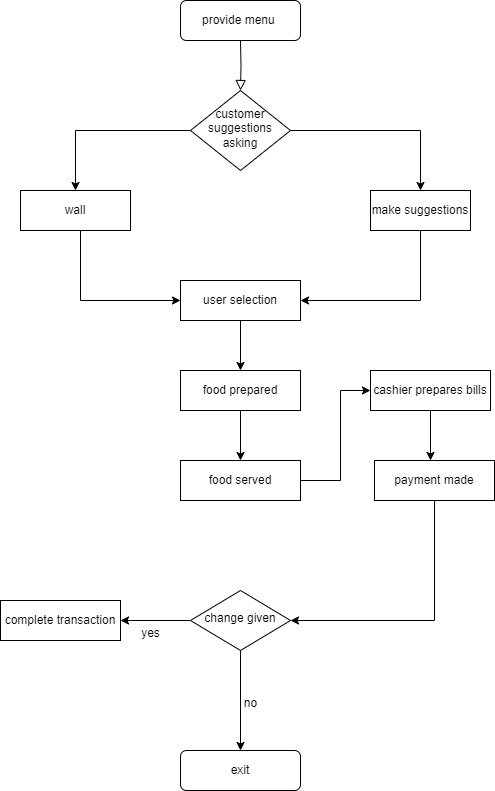
=> The system needs to be accessible and running when the restaurant is open.

e) Scalability

=> The system needs to be able to manage the restaurant's expanding demands, such as an increase in menu items and order volume.

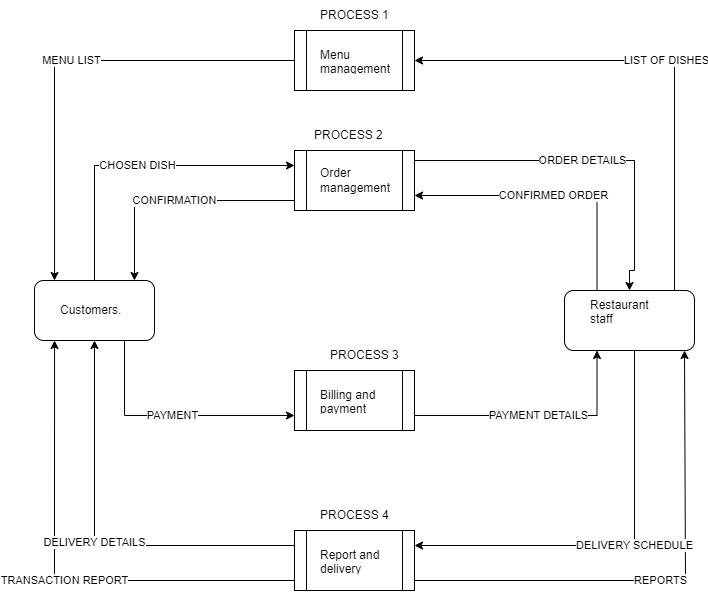
=> It aims to permit expanding the software infrastructure or adding more hardware resources.

FLOWCHART

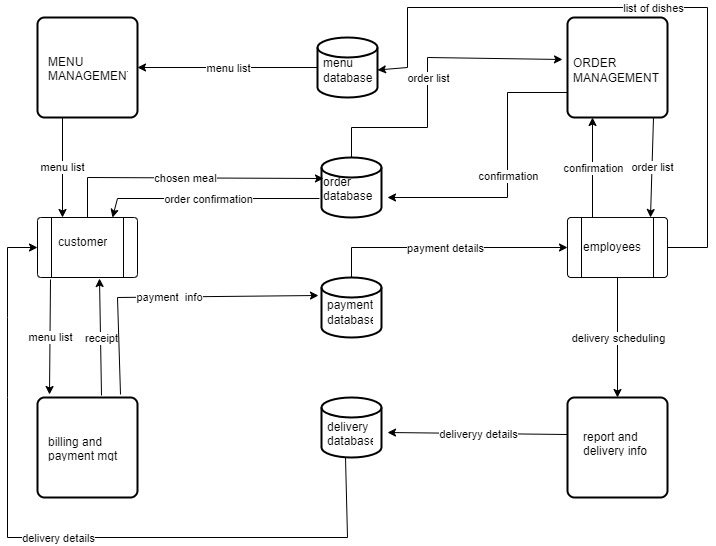


THE DATA FLOW DIAGRAM OF THE RESTAURANT.

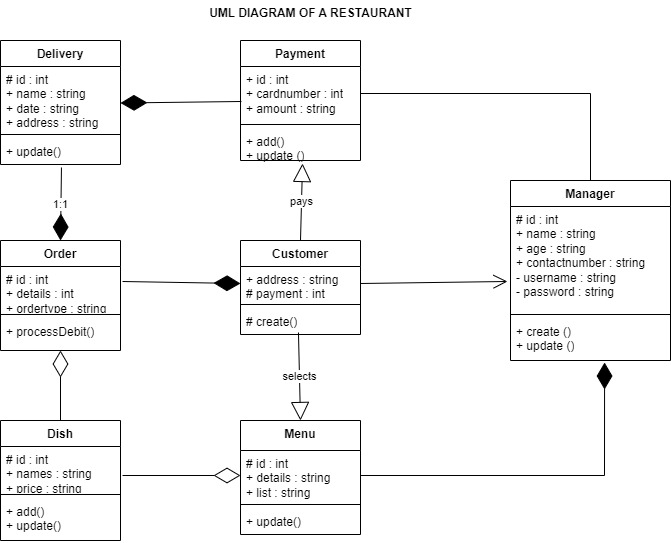
A.



B.



UML DIAGRAMS



THE PROGRAM CODE.

#include <iostream>

#include <iomanip>

#include <vector>

#include <string>

using namespace std;

struct MenuItem {

string name;

double price;

string currency; // New member to store currency

};

struct Customer {

string name;

double totalSpent;

};

class Restaurant {

private:

vector<MenuItem> menu;

vector<Customer> customers; // Vector to hold customer information

double totalSales;

public:

Restaurant() {

totalSales = 0.0;

}

void addMenuItem(const string& name, double price, const string& currency) {

MenuItem item;

item.name = name;

item.price = price;

item.currency = currency;

menu.push\_back(item);

}

void displayMenu() {

cout << "Menu:" << endl;

for (size\_t i = 0; i < menu.size(); i++) {

const MenuItem& item = menu[i];

cout << item.name << " - " << item.currency << fixed << setprecision(2) << item.price << endl;

}

}

bool isMenuItemAvailable(const string& itemName) {

for (size\_t i = 0; i < menu.size(); i++) {

if (menu[i].name == itemName) {

return true;

}

}

return false;

}

double getItemPrice(const string& itemName) {

for (size\_t i = 0; i < menu.size(); i++) {

if (menu[i].name == itemName) {

return menu[i].price;

}

}

return 0.0; // Return 0.0 if item not found

}

void processOrder() {

string customerName;

cout << "Enter your name: ";

getline(cin, customerName);

Customer customer;

customer.name = customerName;

customer.totalSpent = 0.0;

vector<string> orderedItems;

bool orderAgain = true;

while (orderAgain) {

string itemName;

cout << "Enter the food item you want to order: ";

getline(cin, itemName);

if (!isMenuItemAvailable(itemName)) {

cout << "Item not found in the menu." << endl;

} else {

double itemPrice = getItemPrice(itemName);

orderedItems.push\_back(itemName);

cout << "Item " << itemName << " is available. Price: " << itemPrice << " Ksh" << endl;

double payment;

cout << "Enter the payment amount: ";

cin >> payment;

cin.ignore(); // Ignore the newline character left in the input buffer

if (payment < itemPrice) {

cout << "Insufficient payment. Order canceled." << endl;

} else {

double change = payment - itemPrice;

totalSales += itemPrice;

customer.totalSpent += itemPrice;

cout << "Order placed successfully. Change: " << change << " Ksh" << endl;

generateReceipt(customerName, itemName, itemPrice, payment, change);

cout << "Do you want to order more? (1: Yes / 0: No): ";

int choice;

cin >> choice;

cin.ignore(); // Ignore the newline character left in the input buffer

orderAgain = (choice == 1);

}

}

}

// List the food items ordered

for (int i = 0; i < orderedItems.size(); i++) {

cout << orderedItems[i] << endl;

}

customers.push\_back(customer); // Store customer information after they are done ordering

}

void generateReceipt(const string& customerName, const string& itemName, double itemPrice, double payment, double change) {

cout << endl;

cout << "Receipt:" << endl;

cout << "---------------------------------" << endl;

cout << "Customer: " << customerName << endl;

cout << "Item: " << itemName << endl;

cout << "Price: " << itemPrice << " Ksh" << endl;

cout << "Payment: " << payment << " Ksh" << endl;

cout << "Change: " << change << " Ksh" << endl;

cout << "---------------------------------" << endl;

cout << endl;

}

void displayTotalSales() {

cout << "Total Sales: " << totalSales << " Ksh" << endl;

}

void displayCustomerInfo() {

cout << "Customer Information:" << endl;

for (size\_t i = 0; i < customers.size(); i++) {

cout << "Name: " << customers[i].name << ", Total Spent: " << customers[i].totalSpent << " Ksh" << endl;

}

}

};

int main() {

Restaurant restaurant;

// Adding menu items

restaurant.addMenuItem("mutura", 20, "Ksh");

restaurant.addMenuItem("nyama choma", 220, "Ksh");

restaurant.addMenuItem("mukimo", 60, "Ksh");

restaurant.addMenuItem("fried fish", 150, "Ksh");

restaurant.addMenuItem("sukuma wiki", 40, "Ksh");

restaurant.addMenuItem("tea", 20, "Ksh");

restaurant.addMenuItem("samosa", 20, "Ksh");

restaurant.addMenuItem("fries", 100, "Ksh");

restaurant.addMenuItem("chapati", 20, "Ksh");

restaurant.addMenuItem("omena", 100, "Ksh");

restaurant.addMenuItem("beef stew", 150, "Ksh");

restaurant.addMenuItem("mahamri", 10, "Ksh");

restaurant.addMenuItem("uji power", 60, "Ksh");

restaurant.addMenuItem("ugali", 30, "Ksh");

restaurant.addMenuItem("rice ndengu", 60, "Ksh");

restaurant.addMenuItem("rice potato", 70, "Ksh");

restaurant.addMenuItem("soda", 50, "Ksh");

restaurant.addMenuItem("smoothie", 50, "Ksh");

restaurant.addMenuItem("pilau", 150, "Ksh");

// Add other menu items here...

// Displaying the menu

restaurant.displayMenu();

// Processing orders

restaurant.processOrder();

// Displaying the total sales

restaurant.displayTotalSales();

// Displaying customer information

restaurant.displayCustomerInfo();

return 0;

}