

**Birla Institute of Technology & Science, Pilani**

Dubai

United Arab Emirates

**A Report**

On

**Restaurant Management**

Prepared for

**Ms. Sapna Sadhwani**

By

**Syeda Suha Amber - 2018A7PS0057U**

**Aafia Iqbal - 2018A7PS0061U**

**Mardiyah Khadijah - 2018A7PS0257U**

11 December 2019

## **ABSTRACT**

This report documents the process of designing, developing and testing a software system to be used in a restaurant; usually given the name restaurant management system. The restaurant management system is there to help communication between all teams within a restaurant by minimising the probability of human errors.

## **ACKNOWLEDGEMENTS**

We express our sincere and heartfelt gratitude to Prof. R. N. Saha, Director, BITS Pilani, Dubai Campus for giving us an opportunity to apply and understand our engineering concepts in a practical atmosphere.

We would also like to thank Ms. Sapna Sadhwani, our instructor in-charge of Object Oriented Programming, for providing us with all the knowledge and assistance required for successful completion of this report and particularly for giving us the motivation at every step.

Also, our sincere gratitude to Dr. Sujala D.Shetty, for guiding us and giving us her valuable time, without which this report could not have been possible.

We are indebted to Mr. Mishra and Mr. Kashyap for their invaluable help with developing our project.

## **TABLE OF CONTENTS**

Abstract

Acknowledgement

Table of contents

Chapter 1 : Introduction

1.1. The Problem

1.2. Objectives -

1.3. Scope

1.4. Importance

1.5. Report Preview

1.6. Flow Diagram

Chapter 2: Code

2.1. Main Menu

2.2. Salad

2.3. Pasta

2.4. Order

2.5. Bill

2.6. Feedback

2.7. Database

Chapter 3: Conclusion

References

## **CHAPTER 1: INTRODUCTION**

### **1.1. The Problem**

According to a research article written by Horizons, in 2006 within the UK there was just over 26,000 restaurants with 734 million meals served that year. As this restaurant sector was worth £7.61 billion, any restaurant generating a good business reputation could lead to the making of a very successful and profitable business. The problem for many businesses is to ensure that they not only attract new customers but to ensure they maintain their existing clientele. It has been argued many times that an existing customer is worth more to a business than a new customer as the cost to attract a new customer can be up to five times the cost to retain an old customer. An online article by Paul Lemberg[9], discusses the pros and cons of this argument.

Within the restaurant sector, a customer is likely to return to the restaurant in the future if they received an excellent customer service as well as appetising food. However, if they had to wait for an unreasonable amount of time or there was a mistake in the order, it's very unlikely the customer would return.

Therefore a solution to this problem would be to minimise mistakes within the order and bill, and help eradicate delays as well as encouraging teamwork and communication within the team.

We have used Java class 'awt' to design the graphic user interface for the project, to make it more user-friendly. We have also connected our program to MySQL, which stores information regarding the orders and customer feedback for later use.

### **1.2. Objective**

The objective of this project is to build an electronic restaurant management system using all of the skills and techniques taught to us in our object oriented programming course, using Java.

### **1.3. Scope**

This project will focus on the dine-in/take-away food ordering system, feedback portal, and the creation of a menu where individuals can search for dishes and order what they are interested in. It also has a rating where customers are encouraged to post their opinions of the restaurant so that the restaurant can improve its performance.

#### **1.4. Importance**

One of the main objectives of any business is to maximize profit by increasing efficiency without compromising customer satisfaction. Currently, many restaurants use paper-based system to communicate between the restaurant and customers which can be shown to be one of the least efficient approaches. Even though this approach is implemented in successful profitable restaurants, there are several problems which could be seen as reducing the restaurant's efficiency:

- Miscommunication caused by handwriting.
- Unmanageable order logging.
- Inefficient restaurant-customers communication.
- Difficult order tracking and time management.
- Limited statistical output.

By introducing an electronic restaurant management system these problems can be avoided or improved leading to an increase in profits.

#### **1.5. Report Preview**

Besides Introduction, the report contains three chapters. Chapter 2 gives an analysis of the code which further contains seven units. Chapter 3 sums up the program and shows the results through screenshots of the program's output.

## **CHAPTER 2: CODE**

### **2.1. Class Main Menu:**

This class loads the main function when the program is run. An important feature of this class is the “order” button that loads the menu page for salads.

```
order.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        new SaladPage();  
        welcome.dispose();  
    }  
});  
  
public static void main(String[] args) {  
    new MainMenu();  
}
```



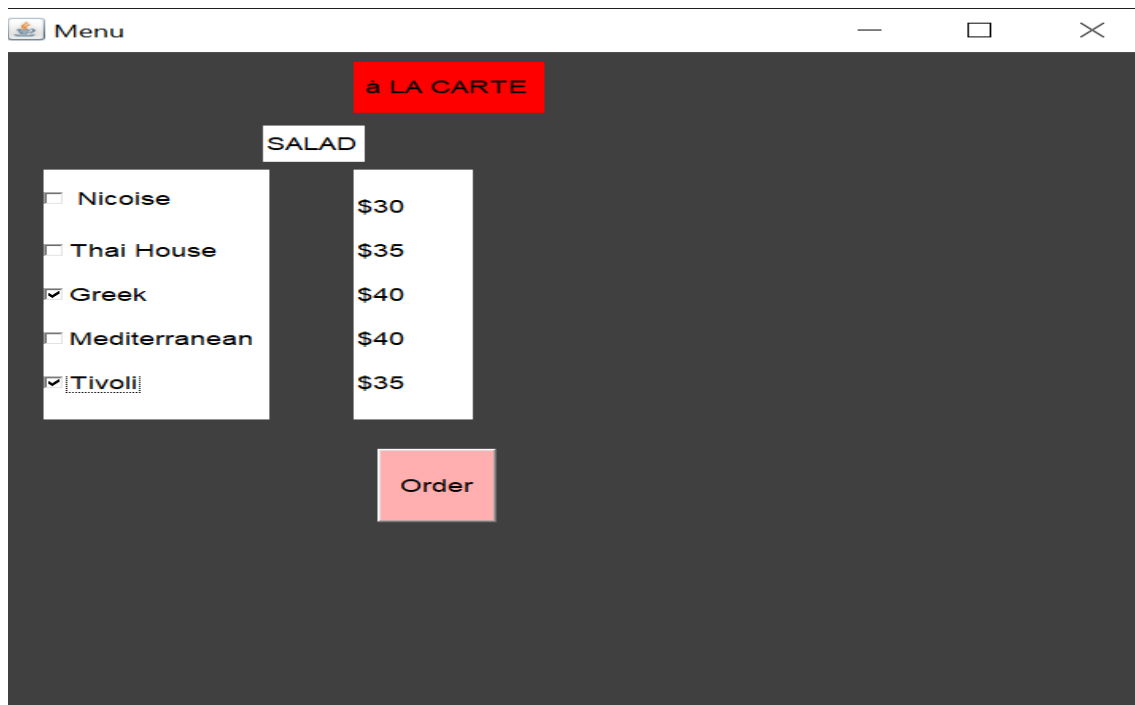
## 2.2. Menu

### 2.2.1 Class SaladPage

This class is responsible for creating the frame of the salad menu, when the constructor of the class is called. When the submit button is pressed, the selected menu items are added into an object of the class “*Order*”. This object is later passed as a parameter to the constructor of the “*PastaPage*” class, responsible for creating the pasta menu.

```
public void actionPerformed(ActionEvent e)
{
    if(c1.getState()==true)
        (order).addSalad(s1,p1);
    if(c2.getState()==true)
        order.addSalad(s2,p2);
    if(c3.getState()==true)
        order.addSalad(s3,p3);
    if(c4.getState()==true)
        order.addSalad(s4,p4);
    if(c5.getState()==true)
        order.addSalad(s5,p5);

    saladpage.dispose();
    new PastaPage(order);
}
```



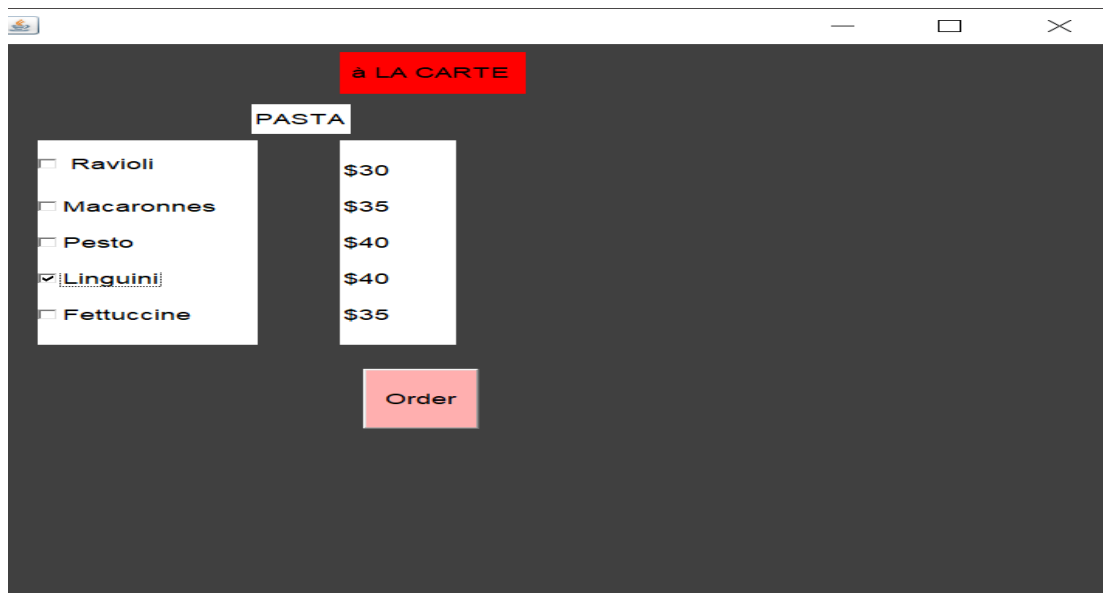


### 2.2.2 Class PastaPage

This class is responsible for creating the frame of the pasta menu, when the constructor of the class is called. It receives an object of class “*Order*” that already stores the previously selected salad items. When the submit button is pressed, the selected pasta items are added into an object of the class “*Order*”. This object is later passed as a parameter to the constructor of the “*Billing*” class, responsible for creating the receipt for the order.

```
public void actionPerformed(ActionEvent e)
{
    if(c1.getState()==true)
        (order).addPasta(s1,p1);
    if(c2.getState()==true)
        order.addPasta(s2,p2);
    if(c3.getState()==true)
        order.addPasta(s3,p3);
    if(c4.getState()==true)
        order.addPasta(s4,p4);
    if(c5.getState()==true)
        order.addPasta(s5,p5);

    pastapage.dispose();
    new Billing(order);
}
```



### **2.3. Class Order**

The purpose of this class is to store and pass on details of ordered menu items to parameterized constructors of the next menu page. It has four array lists that store names of dishes and prices, along with getter and setter functions for the size and content of the class.

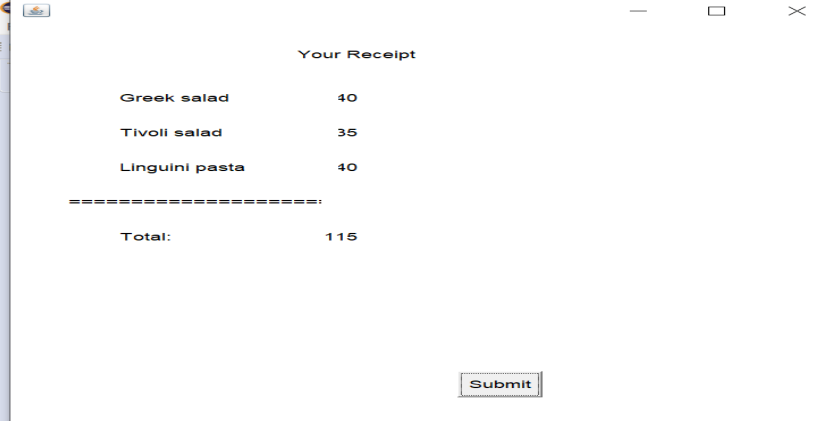
```
public class Order {

private ArrayList <String> salads = new ArrayList<String>();
private ArrayList <Integer> saladsPrice = new
ArrayList<Integer>();
private ArrayList <String> pastas = new ArrayList<String>();
private ArrayList <Integer> pastasPrice = new
ArrayList<Integer>();

public void addPasta(String pasta, int pastaPrice);
public void addSalad(String s1, int p1) ;
public String getSalad(int i);
public Integer getSaladPrice(int i);
public String getPasta(int i);
public Integer getPastaPrice(int i);
public int getSaladSize();
public int getPastaSize();
}
```

## **2.4. Class Billing**

The purpose of this class is to produce a receipt of the menu items ordered. When the *submit* button is pressed, this order is copied into the MySQL database linked with the program.



The screenshot shows a Java Swing window with a title bar containing standard OS controls (minimize, maximize, close). The window's content area displays a receipt titled "Your Receipt". The receipt lists three items: "Greek salad" for 40, "Tivoli salad" for 35, and "Linguini pasta" for 40. A horizontal line of equals signs separates the items from the total. The total is listed as "Total: 115". At the bottom right of the window is a button labeled "Submit".

Your Receipt	
Greek salad	40
Tivoli salad	35
Linguini pasta	40
=====	
Total:	115

Submit

## **2.5. Class Feedback**

This class does the job of displaying and collecting feedback of the restaurant from the customer. Once the submit button is pressed, the information is copied into the database and a new menu page is loaded for the next customer.

```
import java.sql.Connection ;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.*;
import java.awt.*;
import java.awt.event.*;

public class Feedback implements WindowListener
{

    public Feedback()
    {
        initComponents();
    }

    private CheckboxGroup A;
        private Checkbox A1;
        private Checkbox A2;
        private Checkbox A3;

        private CheckboxGroup F;
        private Checkbox F1;
        private Checkbox F2;
        private Checkbox F3;

        private CheckboxGroup S;
        private Checkbox S1;
        private Checkbox S2;
        private Checkbox S3;

        private CheckboxGroup O;
        private Checkbox O1;
        private Checkbox O2;
        private Checkbox O3;
```

```

private CheckboxGroup Q;
private Checkbox Q1;
private Checkbox Q2;
private Checkbox Q3;

private CheckboxGroup V;
private Checkbox V1;
private Checkbox V2;
private Checkbox V3;

private TextField NameT;
private Button jButton1;
private Frame menu;
private Label head, f, a, s, o, q, v;

private void initComponents()
{
    menu = new Frame("Feedback");
        menu.addWindowListener(this);

        head = new Label("Name: ");
        head.setBounds(50, 30, 70, 30);
        f = new Label("Food");

        f.setBounds(40, 220, 60, 30);
        a = new Label("Ambience");
        a.setBounds(40, 60, 60, 30);
        s = new Label("Service");
        s.setBounds(40, 380, 60, 30);

        o = new Label("Money");
        o.setBounds(160, 60, 60, 30);
        q = new Label("Quantity");
        q.setBounds(160, 220, 60, 30);

        v = new Label("Overall");
        v.setBounds(160, 380, 60, 30);
        menu.add(f);
        menu.add(a);
        menu.add(s);
        menu.add(o);
        menu.add(q);
        menu.add(v);
        menu.add(head);

```

```

menu.setSize(260, 600);
menu.setLayout(null);
menu.setVisible(true);

jButton1 = new Button("Submit");
jButton1.setBounds(100, 560, 60, 30);
menu.add(jButton1);

NameT = new TextField();
NameT.setBounds(130, 30, 60, 30);
menu.add(NameT);
A = new CheckboxGroup();
A1 = new Checkbox("1", A, false);
A2 = new Checkbox("2", A, false);
A3 = new Checkbox("3", A, true);
menu.add(A1); menu.add(A2); menu.add(A3);
A1.setBounds(40, 100, 40, 30);
A2.setBounds(40, 140, 40, 30);
A3.setBounds(40, 180, 40, 30);

F = new CheckboxGroup();
F1 = new Checkbox("1", F, false);
F2 = new Checkbox("2", F, false);
F3 = new Checkbox("3", F, true);
menu.add(F1); menu.add(F2); menu.add(F3);
F1.setBounds(40, 260, 40, 30);
F2.setBounds(40, 300, 40, 30);
F3.setBounds(40, 340, 40, 30);

S = new CheckboxGroup();
S1 = new Checkbox("1", S, false);
S2 = new Checkbox("2", S, false);
S3 = new Checkbox("3", S, true);
menu.add(S1); menu.add(S2); menu.add(S3);
S1.setBounds(40, 420, 40, 30);
S2.setBounds(40, 460, 40, 30);
S3.setBounds(40, 500, 40, 30);

O = new CheckboxGroup();
O1 = new Checkbox("1", O, false);
O2 = new Checkbox("2", O, false);
O3 = new Checkbox("3", O, true);
menu.add(O1); menu.add(O2); menu.add(O3);
O1.setBounds(160, 100, 40, 30);

```

```

O2.setBounds(160,140,40,30);
O3.setBounds(160,180,40,30);

Q = new CheckboxGroup();
Q1 = new Checkbox("1",Q,false);
Q2 = new Checkbox("2",Q,false);
Q3 = new Checkbox("3",Q,true);
menu.add(Q1);menu.add(Q2);menu.add(Q3);
Q1.setBounds(160,260,40,30);
Q2.setBounds(160,300,40,30);
Q3.setBounds(160,340,40,30);

V = new CheckboxGroup();
V1 = new Checkbox("1",V,false);
V2 = new Checkbox("2",V,false);
V3 = new Checkbox("3",V,true);
menu.add(V1);menu.add(V2);menu.add(V3);
V1.setBounds(160,420,40,30);
V2.setBounds(160,460,40,30);
V3.setBounds(160,500,40,30);

jButton1.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent
evt) {
        jButton1ActionPerformed(evt);
    }
});

}

    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }

@Override
public void windowOpened(WindowEvent e) {}
@Override
public void windowClosed(WindowEvent e) {}
@Override
public void windowIconified(WindowEvent e) {}
@Override

```

```

public void windowDeiconified(WindowEvent e) {}
@Override
public void windowActivated(WindowEvent e) {}
@Override
public void windowDeactivated(WindowEvent e) {}

String
foodrate, ambrate, qntyrate, servicerate, moneyrate, overallrate;

private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt)
{

    if (F1.getState() == true)
        foodrate = "1";
    else if (F2.getState() == true)
        foodrate = "2";
    else
        foodrate = "3";

    if (A1.getState() == true)
        ambrate = "1";
    else if (A2.getState() == true)
        ambrate = "2";
    else
        ambrate = "3";

    if (Q1.getState() == true)
        qntyrate = "1";
    else if (Q2.getState() == true)
        qntyrate = "2";
    else
        qntyrate = "3";

    if (S1.getState() == true)
        servicerate = "1";
    else if (S2.getState() == true)
        servicerate = "2";
    else
        servicerate = "3";

    if (O1.getState() == true)
        moneyrate = "1";
    else if (O2.getState() == true)
        moneyrate = "2";

```



```

else
moneyrate="3";

if (V1.getState()==true)
overallrate="1";
else if (V2.getState()==true)
overallrate="2";
else
overallrate="3";

try
{
    Class.forName("com.mysql.cj.jdbc.Driver");
    String query = "Insert into Feedback
(Name,foodrating,ambienccrating,quantityrating,
servicerating,moneyrating,overallrating) values(?,?,?,?,?,?,?);";
    Connection con
=DriverManager.getConnection("jdbc:mysql://localhost:3306/oops","r
oot","123456789");
    Statement stmt = con.createStatement();
    PreparedStatement ps = con.prepareStatement(query);
    ps.setString(1,NameT.getText());
    ps.setString(2,foodrate);
    ps.setString(3,ambrate);
    ps.setString(4,qntyrate);
    ps.setString(5,servicerate);
    ps.setString(6,moneyrate);
    ps.setString(7,overallrate);
    int i=ps.executeUpdate();
//    stmt.executeUpdate(query);

    new MainMenu();

}


catch (Exception e)
{
    System.out.print(e);
}

menu.dispose();
new MainMenu();

```

}

}

 **Feedback**

Name:

Ambience

1

2

3

Food

1

2

3

Service

1

2

3

Money

1

2

3

Quantity

1

2

3

Overall

1

2

3

Submit

## 2.7. Database

The database comprises of two tables, *kitchen* and *feedback*. The *kitchen* table stores the orders (in terms of dishes and price), whereas the *feedback* table stores the feedback customer wise.

```
MySQL 8.0 Command Line Client
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use oops
Database changed
mysql> show tables;
+-----+
| Tables_in_oops |
+-----+
| feedback       |
| k1             |
| kitchen        |
+-----+
3 rows in set (0.00 sec)

mysql> select * from feedback;
+-----+-----+-----+-----+-----+-----+-----+
| Name | foodrating | ambiencerating | quantityrating | servicerating | moneyrating | overallrating |
+-----+-----+-----+-----+-----+-----+-----+
| Aafia | 3          | 3              | 3              | 3              | 3          | 3              |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from kitchen ;
+-----+-----+
| OrderName | price |
+-----+-----+
| Greek     | 40    |
| Tivoli    | 35    |
| Linguini  | 40    |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

### **CHAPTER 3: CONCLUSION**

This project has helped us to attain new skills as well as develop existing skills. The skills attained have been both technical and individual with the main individual skill being project management which required good time keeping and management of the workload. Some technical skills that have been developed include:

- Using multiple classes and organizing a large program into smaller modules using classes.
- Using constructors of classes to pass objects of other user-defined classes
- Initializing and defining arrays of class objects
- Using array lists and adding, modifying and accessing their content
- Using GUI components using java class awt
- Using multiple awt components in frames and applying action listeners on them
- Using ActionListener and WindowListener interfaces
- Using anonymous classes to
- Connecting MySQL databases to Java IDE
- Establishing a connection with MySQL within the Java program
- Sending queries to enter user's data into database
- Dealing with try and catch blocks to catch exceptions
- Dealing with compilation and runtime errors and learning how to resolve them