



Transform Yourself

RESTAURANT BILL SYSTEM

Name : JANANI E

Roll No : 23ADR068

Date : 12/11/2024



Estd : 1984

INTRODUCTION

Purpose:

The purpose of the Restaurant Bill System is to automate and streamline the billing process in restaurants, ensuring faster and more accurate calculations while reducing human errors. It manages orders, discounts, and transaction records efficiently using a database.

Technology Used:-

- Java programming language
- JDBC (Java Database Connectivity) for interaction with the database
- SQL for data storage and retrieval

SYSTEM DESIGN AND FEATURE

Features:

- Real Time Bill Calculation
- Database Integration with MySQL
- Receipt Generation
- User-Friendly Interface
- Stores customer details such as name and date.
- Stores order details, including customer ID, item name, quantity, price, and total.

Database:

- MySQL (or any relational database)

CORE FUNCTIONALITIES

1. **Customer Management:**
Captures and stores customer details in the database.
2. **Order Management:**
Allows input and management of multiple order items.
3. **Bill Calculation:**
Calculates total bill with a 10% discount.
4. **Database Integration:**
Stores customer and order data persistently in MySQL.
5. **Receipt Printing:**
Prints a formatted, itemized receipt.
6. **Error Handling:**
Handles input validation and database connection errors.

KEY COMPONENTS OF THE CODE

1. **Format(String date, String name)**

Prints the restaurant name, current date, and customer name in a formatted bill header.

2. **printBill(String item, int qty, float price)**

Prints details of a single order item, including its name, quantity, and total price.

3. **CalculateBill(float[] prices, int[] quantities)**

Computes the total bill by multiplying quantities with prices, applies a 10% discount, and prints the final amount.

4. **InsertCustomer(String name)**

Adds the customer's name and current date into the database's

5. **insertOrder(int customerId, String item, int quantity, float price)**

Inserts an order's details (item name, quantity, price, total cost) into the order3 table, linked to a specific customer ID.

DATABASE DESIGN

1. TABLE NAME:

- customer1

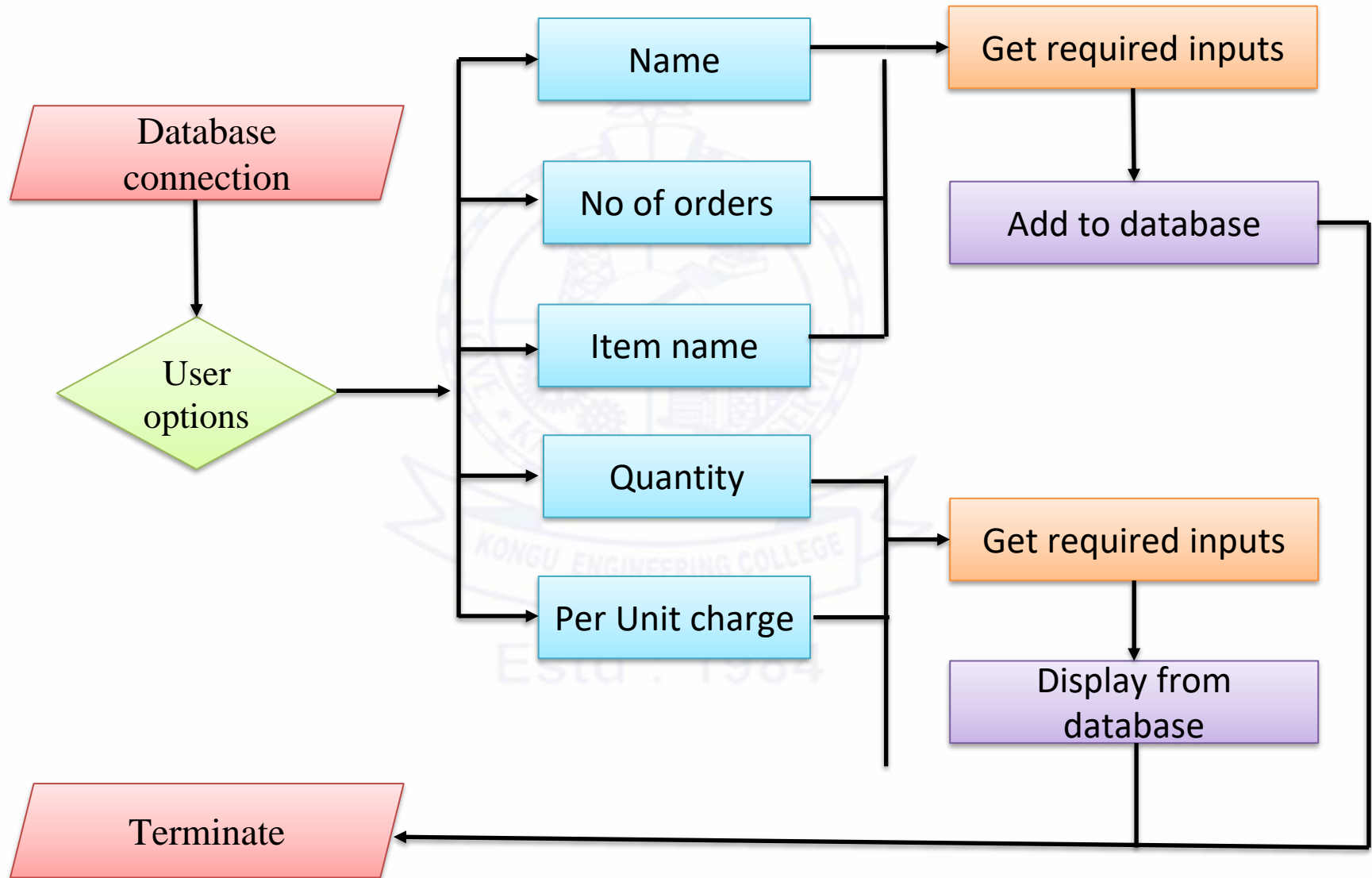
2. COLUMNS:

- id(INT,AUTO_INCREMENT,PRIMARY KEY)
- name(VARCHAR(100))
- date(DATE)

3. SQL OPERATIONS:

- Create Table
- Calculate the bill by obtaining the item,quantity and charge from the user.

Flow Chart



OUTPUT:

```
Command Prompt
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JANANI E>D:

D:\>cd 23adr068-java

D:\23adr068-java>set path="C:\Program Files\Java\jdk-22\bin"

D:\23adr068-java>javac Bill.java

D:\23adr068-java>java -cp ".;C:\Users\JANANI E\Desktop\java pro\mysql-connector-j-9.1.0\mysql-connector-j-9.1.0\mysql-connector-j-9.1.0.jar" Bill

Welcome to the restaurant billing code using Java

ENTER CUSTOMER NAME: jan
Enter the Number Of orders requested by the customer: 4

Enter Item 1:
Please Enter The Item Name: idli
Enter Quantity of Items: 3
Enter the Per Unit Charge of Item: 5

Enter Item 2:
Please Enter The Item Name: Dosa
Enter Quantity of Items: 4
Enter the Per Unit Charge of Item: 10

Enter Item 3:
Please Enter The Item Name: Vadai
Enter Quantity of Items: 5
Enter the Per Unit Charge of Item: 6

Enter Item 4:
Please Enter The Item Name: poori
Enter Quantity of Items: 5
Enter the Per Unit Charge of Item: 12
```


BILL RECEIPT:

```
Command Prompt
Enter the Per Unit Charge of Item: 6

Enter Item 4:
Please Enter The Item Name: poori
Enter Quantity of Items: 5
Enter the Per Unit Charge of Item: 12

=====
GREAT OBSERVER RESTAURANT CLUB
=====
Date :1/1/2024
Invoice To :jan
=====
Items                QTY                Total
-----
idli                  3                  15.0
Dosa                  4                  40.0
Vadai                 5                  30.0
poori                 5                  60.0
=====

Discount:              10%              14.5
=====

Total Amount :              130.5
=====

Thank you and visit again!!!

D:\23adr068-java>
```

DATABASE CODE:

The screenshot displays the MySQL Workbench interface. The main editor window contains the following SQL code:

```
-- Create the customer1 table
CREATE TABLE customer1 (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  date DATE
);

-- Create the order3 table with a foreign key referencing customer1
CREATE TABLE order3 (
  id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT,
  item_name VARCHAR(100),
  quantity INT,
  price FLOAT,
  total FLOAT,
```

The Output window at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	09:57:31	DROP DATABASE IF EXISTS restaurants	2 row(s) affected	0.047 sec
2	09:57:31	CREATE DATABASE restaurants	1 row(s) affected	0.015 sec
3	09:57:31	USE restaurants	0 row(s) affected	0.000 sec
4	09:57:31	CREATE TABLE customer1 (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), date D...	0 row(s) affected	0.032 sec
5	09:57:31	CREATE TABLE order3 (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, item_name VAR...	0 row(s) affected	0.031 sec

CUSTOMER TABLE:

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- javademo
- javademo
- restaurant
- sys

order x

```
1 use restaurants;
2 select * From customer1;
3
```

Limit to 2000 rows

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

id	name	date
NULL	NULL	NULL

customer1 x

Apply Revert

Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	09:57:31	DROP DATABASE IF EXISTS restaurants	2 row(s) affected	0.047 sec
2	09:57:31	CREATE DATABASE restaurants	1 row(s) affected	0.015 sec
3	09:57:31	USE restaurants	0 row(s) affected	0.000 sec
4	09:57:31	CREATE TABLE customer1 (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), date ...	0 row(s) affected	0.032 sec
5	09:57:31	CREATE TABLE order3 (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, item_name VA...	0 row(s) affected	0.031 sec
6	09:58:30	use restaurants	0 row(s) affected	0.000 sec
7	09:58:30	select * From customer1 LIMIT 0, 2000	0 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

No object selected

09:58 24-11-2024

ORDER TABLE:

The screenshot displays the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'Schemas' section with a list of databases: javademo, javademo, restaurant, and sys. The 'Query Editor' pane in the center shows the following SQL queries:

```
1 • use restaurants;
2 • select * From order3;
```

The 'Result Grid' pane below the query editor shows the following columns: id, customer_id, item_name, quantity, price, total. The 'Output' pane at the bottom shows the 'Action Output' for the execution of the queries:

#	Time	Action	Message	Duration / Fetch
1	09:57:31	DROP DATABASE IF EXISTS restaurants	2 row(s) affected	0.047 sec
2	09:57:31	CREATE DATABASE restaurants	1 row(s) affected	0.015 sec
3	09:57:31	USE restaurants	0 row(s) affected	0.000 sec
4	09:57:31	CREATE TABLE customer1 (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), date ...	0 row(s) affected	0.032 sec
5	09:57:31	CREATE TABLE order3 (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, item_name VA...	0 row(s) affected	0.031 sec
6	09:58:30	use restaurants	0 row(s) affected	0.000 sec
7	09:58:30	select * From customer1 LIMIT 0, 2000	0 row(s) returned	0.000 sec / 0.000 sec
8	09:59:21	use restaurants	0 row(s) affected	0.000 sec
9	09:59:21	select * From order3 LIMIT 0, 2000	0 row(s) returned	0.000 sec / 0.000 sec

The 'Object Info' pane on the left shows 'No object selected'. The 'SQLAdditions' pane on the right displays a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

OUTPUT:

ORDER DB:

MySQL Workbench screenshot showing the ORDER DB schema and execution log. The schema includes tables: orders, restaurants, items, and users. The execution log shows the following actions:

Time	Action	Message	Duration / Feat
09:57:01	DROP DATABASE IF EXISTS restaurants	2 rows affected	0.047 sec
09:57:01	CREATE DATABASE restaurants	1 row(s) affected	0.015 sec
09:57:01	USE restaurants	0 rows(s) affected	0.000 sec
09:57:01	CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), date	0 rows(s) affected	0.032 sec
09:57:01	CREATE TABLE orders (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, item_name VA	0 rows(s) affected	0.031 sec
09:58:30	USE restaurants	0 rows(s) affected	0.000 sec
09:58:30	select * from customers LIMIT 0, 2000	0 rows(s) returned	0.000 sec / 0.000 sec
09:58:30	USE restaurants	0 rows(s) affected	0.000 sec
09:58:30	select * from orders LIMIT 0, 2000	0 rows(s) returned	0.000 sec / 0.000 sec
10:02:25	USE restaurants	0 rows(s) affected	0.000 sec
10:02:25	select * from orders LIMIT 0, 2000	4 rows(s) returned	0.000 sec / 0.000 sec

CUSTOMER DB:

MySQL Workbench screenshot showing the CUSTOMER DB schema and execution log. The schema includes tables: customers, orders, restaurants, items, and users. The execution log shows the following actions:

Time	Action	Message	Duration / Feat
09:57:01	USE restaurants	0 rows(s) affected	0.000 sec
09:57:01	CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), da	0 rows(s) affected	0.032 sec
09:57:01	CREATE TABLE orders (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT, item_name V	0 rows(s) affected	0.031 sec
09:58:30	USE restaurants	0 rows(s) affected	0.000 sec
09:58:30	select * from customers LIMIT 0, 2000	0 rows(s) returned	0.000 sec / 0.000 sec
09:58:30	USE restaurants	0 rows(s) affected	0.000 sec
09:58:30	select * from orders LIMIT 0, 2000	0 rows(s) returned	0.000 sec / 0.000 sec
10:02:25	USE restaurants	0 rows(s) affected	0.000 sec
10:02:25	select * from orders LIMIT 0, 2000	4 rows(s) returned	0.000 sec / 0.000 sec
10:02:25	USE restaurants	0 rows(s) affected	0.000 sec
10:02:25	select * from customers LIMIT 0, 2000	1 rows(s) returned	0.000 sec / 0.000 sec

OUTPUT IN DATABASE:

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'javademo', 'jvademio', 'restaurant', and 'sys'. The main workspace is titled 'order' and contains a SQL query:

```
1 • use restaurants;
2 • select * From customer1;
3 • select * From order3;
```

Below the query editor, the 'Result Grid' shows the output of the query. It has columns: id, customer_id, item_name, quantity, price, total. The data rows are:

id	customer_id	item_name	quantity	price	total
1	1	idli	3	5	15
2	1	Dosa	4	10	40
3	1	Vadai	5	6	30
4	1	poori	5	12	60

The bottom panel shows the 'Output' tab with 'Action Output' selected. It displays a log of database actions and their results:

#	Time	Action	Message	Duration / Fetch
9	09:59:21	select * From order3 LIMIT 0, 2000	0 row(s) returned	0.000 sec / 0.000 sec
10	10:02:28	use restaurants	0 row(s) affected	0.000 sec
11	10:02:28	select * From order3 LIMIT 0, 2000	4 row(s) returned	0.000 sec / 0.000 sec
12	10:02:55	use restaurants	0 row(s) affected	0.000 sec
13	10:02:55	select * From customer1 LIMIT 0, 2000	1 row(s) returned	0.000 sec / 0.000 sec
14	10:03:14	use restaurants	0 row(s) affected	0.000 sec
15	10:03:14	select * From customer1 LIMIT 0, 2000	1 row(s) returned	0.000 sec / 0.000 sec
16	10:03:14	select * From order3 LIMIT 0, 2000	4 row(s) returned	0.000 sec / 0.000 sec
17	10:03:21	use restaurants	0 row(s) affected	0.000 sec
18	10:03:21	select * From customer1 LIMIT 0, 2000	1 row(s) returned	0.000 sec / 0.000 sec
19	10:03:21	select * From order3 LIMIT 0, 2000	4 row(s) returned	0.000 sec / 0.000 sec

The right sidebar shows the 'SQLAdditions' panel with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

Conclusion

The Restaurant Bill System offers an efficient solution for managing customer details and order transactions using a database-driven approach. By automating the process of adding, updating, removing, and displaying customer information and orders, the system helps streamline the billing process. Integrating with MySQL ensures data integrity and scalability. The user-friendly flowchart and system design make it easier to handle multiple operations while ensuring accurate calculations and record management. This system ultimately contributes to faster service, improved customer experience, and reliable data storage for restaurant operations.

THANK YOU



Estd : 1984