

IV Semester B.Tech. (CCE)
ICT 2266: DATABASE SYSTEMS LAB
MINI-PROJECT IMPLEMENTATION DOCUMENT

MIT Cafeteria Management System

A PROJECT REPORT

submitted by

Aditya Chandra

Reg no1: 200953032

Akshay Kalidatta

Reg no2: 200953106

Vinayak Mehrotra

Reg no3: 200953054

Siddhant Gupta

Reg no4: 200953092



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

ABSTRACT

This Project will provide an interface for the cafeteria manager to access a database that allows them to view all customer, orders, employees, menu items and branches by storing all these details.

The aim is to produce a database that is capable of being instrumental in using a real portal. All of the background functionality is hidden and the user is given a sample UI to insert, update or delete all of the data in the database.

INTRODUCTION

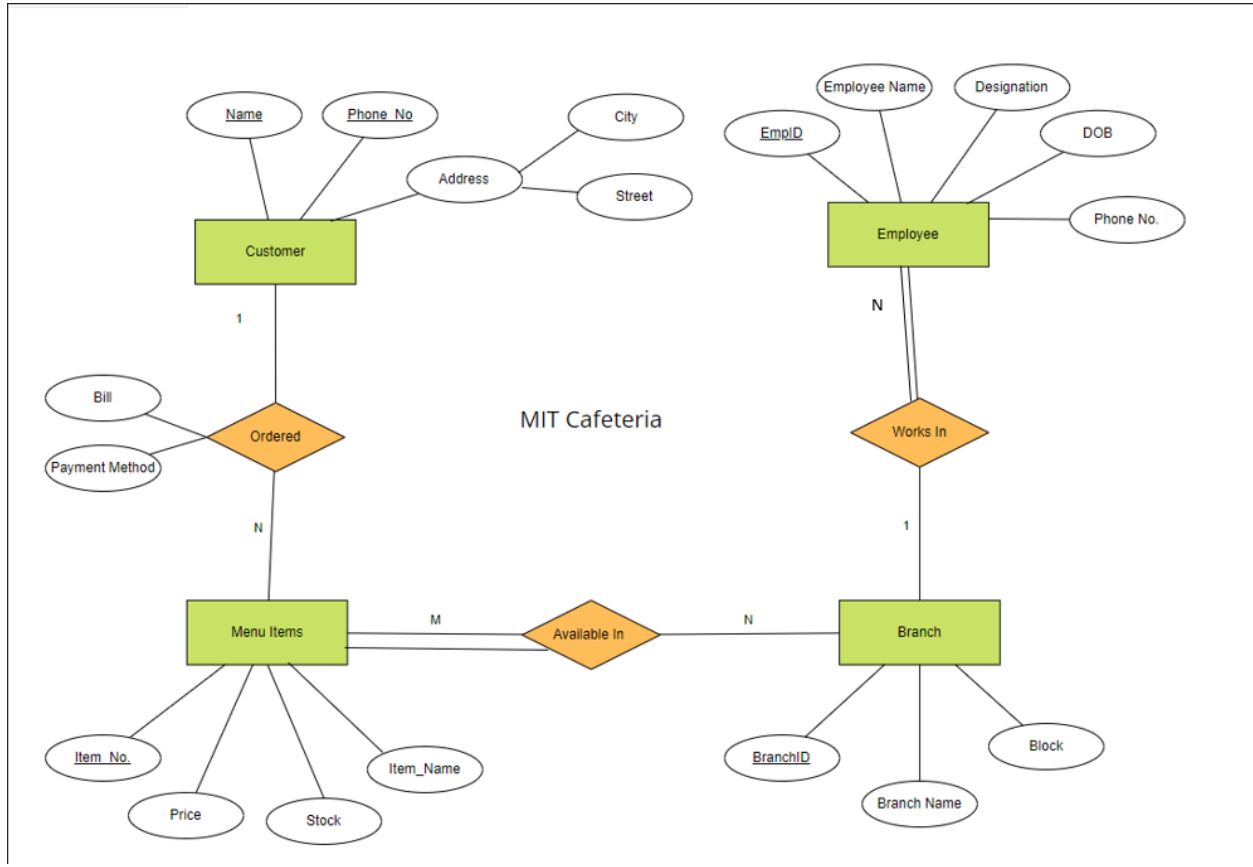
The front end is built on Visual C# using Windows Forms with .NET Framework.

The frontend is connected to an SQL database which is the backend to store, maintain and update database. The database is hosted locally, Data Adapter is used along with a database connection string.

MIT Cafeteria Management is developed to manage all hustle and bustle in the cafeteria by making each and every process efficient and user friendly for the administration by helping the person sitting on the cashier desk. Our product is comprehensive software solution that simplifies day to day operations and increases performance efficiency. It delivers a fully integrated approach to cafeteria management with all its business operations built in a single and comprehensive solution. This solution is not particularly built for a specific kind of cafeteria and it is developed to be general.

METHODOLOGY

Entity-Relationship Diagram



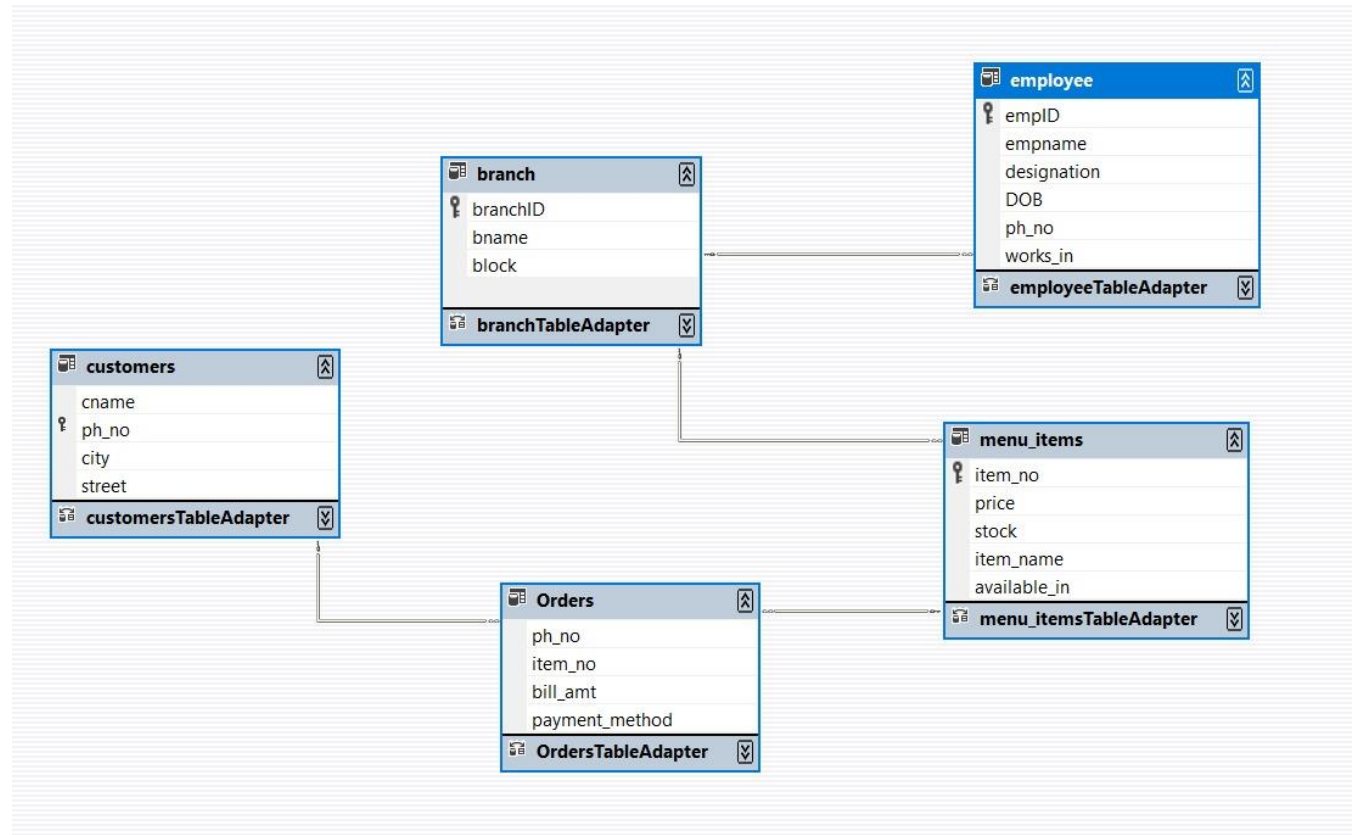
Normalization:

We had initially considered all the information required for the project in one table. This would create a lot of anomalies (insert, update and delete) and would lead to a lot of data redundancy. Hence, we split/normalize the tables to remove these issues.

All the final tables follow the Boyce-Codd Normal Form (BCNF) which ensures it automatically lies in 1NF, 2NF and 3NF. The tables have no partial dependency, transitive dependency which ensures that they are in 2NF, 3NF. All the tuples are such that all the attributes are dependent on just the primary key of their

corresponding tables. This ensure they are in BCNF as well. When a table is in BCNF , there is 0% data redundancy and all the anomalies are removed.

Schema Diagram




IMPLEMENTATION

The project has the following components:

- Login Page for allowing only authorized employees to Login to the application.
- A Dashboard for displaying important information regarding customers, employees, orders and branches, all of which are fetched from the database.
- Employee Forms and Order Forms to create orders.
- And, a Stock Maintenance Dashboard

Features and Components:

1. Login Page (landing page)



A screenshot of a web browser window titled "Login". The page has a yellow background with a wood-grain texture. The title "Employee Login" is displayed in a large, bold, black serif font. Below the title, there are two input fields. The first is labeled "Username" in a black serif font, and the second is labeled "Password" in a black serif font. The "Username" field contains the text "Aditya". The "Password" field contains ten asterisks "*****". Below the input fields is a blue rectangular button with the word "Login" in white serif font. The browser window has a dark gray title bar with standard minimize, maximize, and close buttons.

Employee Login

Username

Password

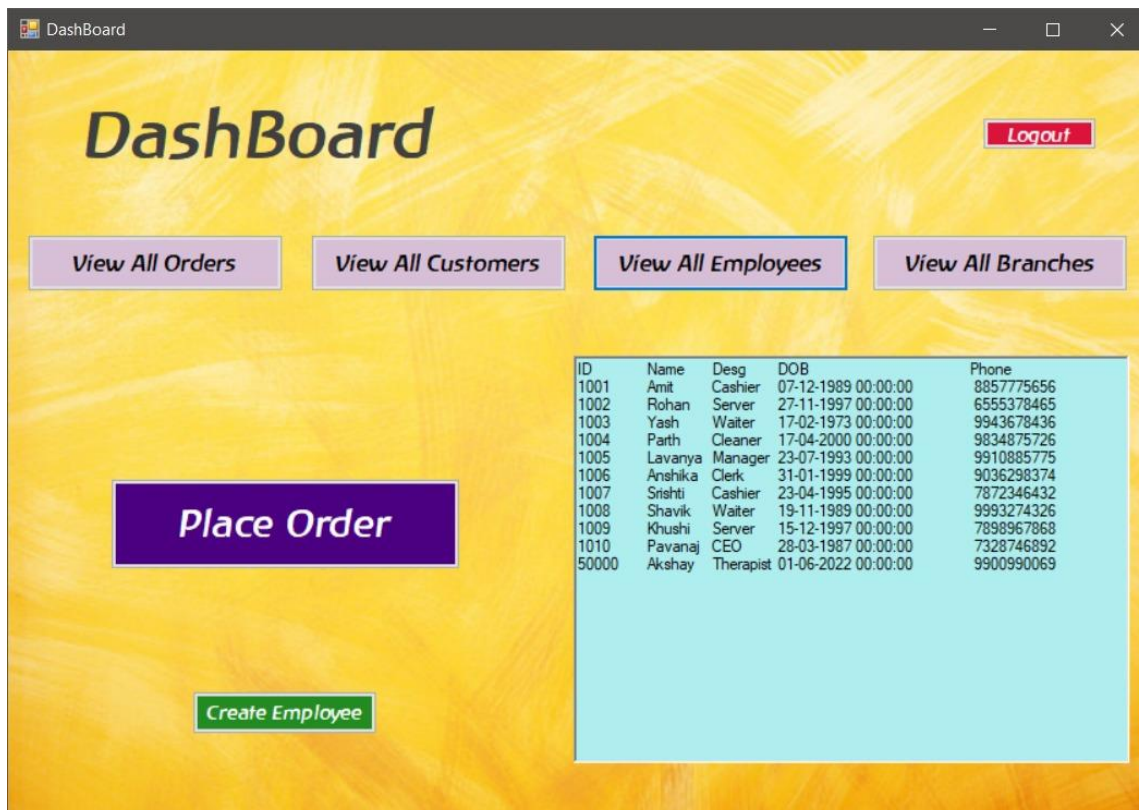
Login

Login page which can be used only by the admin/manager.

2. Incorrect Password or username opens a new window .



3. After successful login you will be directed to the DashBoard where you can view all the orders, customers, employees and branches, the details will be shown in the blue field on the right side. Manager can finally place the order as requested by the customer or can create new employee profile.



Queries Used;

```
select * from customers;
```

```
select * from branch;
```

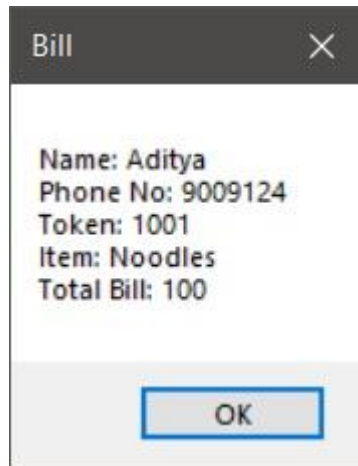
```
select * from employee;
```

```
select * from Orders;
```

4. When the employee clicks on “Place order”, there will be an option to enter customer’s details along with the selection of item they need, with the feature of knowing the availability of the item and finally the mode of payment.

Item No.	Name	Stock
1	Idli	10
2	Dosa	3
3	Vada	7
4	Pizza	2
5	Burger	4
6	Sandwich	6
7	Noodles	2
8	Tea	20
9	Coffee	16
10	Ice Lime	15

After clicking Check Out, Bill will be generated with a unique token number and order details.



PL/SQL Procedures and Functions used:

1)

```
CREATE PROCEDURE order_in
```

```
    @phno varchar(10),
```

```
    @itemNo int,
```

```
    @bill_amt int,
```

```
    @payment varchar(10)
```

```
AS
```

```
    INSERT INTO Orders VALUES(@phno,@itemNo,@bill_amt,@payment);
```

2)

```
CREATE PROCEDURE customer_in
```

```
    @cname varchar(10),
```

```
    @phno varchar(10),
```

```
    @city varchar(20),
```

```
    @street varchar(20)
```

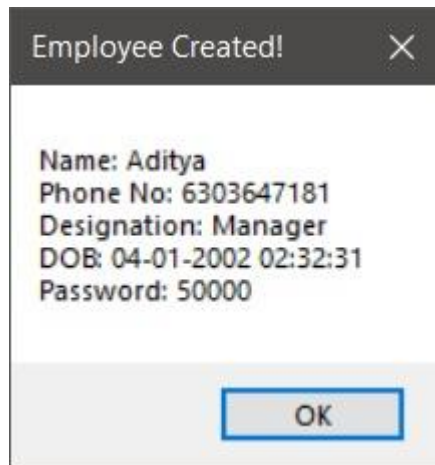
```
AS
```

```
    INSERT INTO customers VALUES(@cname,@phno,@city,@street);
```

5. When Create Employee on Dashboard is clicked we are directed to a new window, asking for the name and various details of the employee.

The screenshot shows a web application window titled "addEmployee". The window has a yellow background with a wood-grain texture. The main heading is "Add Employee". Below the heading, there are four input fields: "Name", "Designation", "Phone number", and "Date of Birth". The "Date of Birth" field is pre-filled with "31 May 2022". At the bottom of the window is a blue button labeled "Create Employee".

On clicking “Create Employee” in the window shown a pop-up with appropriate message is displayed.



PL/SQL Procedures used:

1.)

```
CREATE PROCEDURE employee_in
```

```
    @eid int,
```

```
    @name varchar(50),
```

```
    @designation varchar(15),
```

```
    @DOB date,
```

```
    @phno varchar(10),
```

```
    @worksat varchar(25)
```

```
AS
```

```
    INSERT INTO          employee
```

```
VALUES(@eid,@name,@designation,@DOB,@phno,@worksat);
```

2.)

CREATE OR REPLACE TRIGGER

AFTER INSERT OR UPDATE ON EMPLOYEE

FOR EACH ROW

DECLARE

name varchar;

BEGIN

SELECT ename

INTO name

FROM employee

WHERE vname = :NEW.vname

GROUP BY vname;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.put_line('WARNING.');

END IF;

END;

Implementation Snippets

This screenshot shows the Visual Studio IDE with the `MIT_CafeteriaOrder.cs` file open. The `button1_Click` method is implemented, which connects to a SQL database, executes a query to retrieve menu items, and displays the results in a `richTextBox1`. The `richTextBox1_TextChanged` method is also shown, which is currently empty. The `button2_Click` method is partially visible at the bottom.

```
private void button1_Click(object sender, EventArgs e)
{
    string connection = "Data Source = LAPTOP-HLSQHKF9; " +
        "Initial Catalog = master; " +
        "Integrated Security=true";
    SqlConnection con = new SqlConnection(connection);
    con.Open();
    string query = "select * from menu_items";
    SqlCommand cmd = new SqlCommand(query, con);
    SqlDataReader reader = cmd.ExecuteReader();
    string final = "Item No." + "\t" + "Name" + "\t\t" + "Stock" + "\t" + "\n";
    while (reader.Read())
    {
        string output = reader.GetValue(0) + "\t" + reader.GetValue(3) + "\t\t" + reader.GetValue(2) + "\n";
        final += output;
    }
    richTextBox1.Text = final;
}

private void richTextBox1_TextChanged(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{
    string name = textBox1.Text;
    string phone = textBox2.Text;
    string selected_Item = comboBox1.SelectedItem.ToString();
    if (radioButton1.Checked == true)
    {
        payment = "UPI";
    }
    else if (radioButton2.Checked == true)
    {
    }
}
```

This screenshot shows the Visual Studio IDE with the `MIT_CafeteriaForm1.cs` file open. The `button1_Click` method is implemented, which connects to a SQL database, executes a query to retrieve employee information, and displays the results in a `richTextBox1`. The `textBox1_TextChanged` method is also shown, which is currently empty.

```
private void button1_Click(object sender, EventArgs e)
{
    string username = textBox1.Text;
    string password = textBox2.Text;
    string connection = "Data Source = LAPTOP-HLSQHKF9; " +
        "Initial Catalog = master; " +
        "Integrated Security=true";
    SqlConnection con = new SqlConnection(connection);
    con.Open();
    string query = "select * from employee where empname=" + username + " ";
    SqlCommand cmd = new SqlCommand(query, con);
    SqlDataReader reader = cmd.ExecuteReader();
    int pmd = 0;

    while (reader.Read())
    {
        pmd = Convert.ToInt32(reader.GetValue(0));
    }

    if (pmd == Convert.ToInt32(password))
    {
        this.Hide();
        Dashboard frm = new Dashboard();
        frm.Show();
    }
    else
    {
        MessageBox.Show("Wrong Password", "Wrong Password", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
}
```

CONCLUSION

We created a database that the cafeteria can use for keeping track of orders, customers, employees, and branches. Multiple employee profiles can be created.

The above MIT Cafeteria Management System allows the administration to use an easy-to-use user interface and hereby increasing the efficiency of placing the orders. The time can be saved in a great way if there is unavailability of the item by telling the customer beforehand with the help of database.

All the procedures, functions and triggers allow a hassle-free user interface, enhancing employee's overall experience.

During our database management project, we have learned about the basics of database design. This project gave us the opportunity to try our new skills in practice. While doing this project we also gained deeper understanding on database design and how it can be implemented in real life situations.