# Restaurant Management System

# 1. Introduction

This project is a console-based **Restaurant Management System** developed in C++. The main purpose of this system is to help restaurant owners and staff manage their day-to-day activities efficiently. It replaces manual paperwork with a digital system.

The project uses **File Handling (CSV files)** to permanently save data like Menu items, Users, Orders, and Bills. This ensures that data is not lost even if the program is closed.

## 2. Project Folder Structure

The project is organized into separate folders to keep the code clean and manageable:

Plaintext

```
Project-Name/
│
├── README.md              <-- Project Documentation
├── .gitignore             <-- Files to ignore (like .exe)
├── src/
│   ├── main.cpp           <-- Entry point of the program
│   ├── login.cpp          <-- Handles Login & Signup logic
│   ├── menu.cpp           <-- Manages Food Menu & Stock
│   ├── tables.cpp         <-- Manages Table Status (Free/Busy)
│   ├── dine_in_order.cpp  <-- Handles Dine-in orders
│   ├── online_order.cpp   <-- Handles Online/Phone orders
│   ├── billing.cpp        <-- Generates Bills & Payments
│   └── utils.cpp          <-- Helper functions (Logs)
│
└── docs/
    └── project_description.pdf
```

### 3. How the Project Works (Step-by-Step Logic)

The system works in a continuous loop until the user decides to exit. Here is the detailed working logic of the modules:

#### *A. Login System (login.cpp)*

- When the program starts, it asks the user to **Login** or **Register**.
- **Signup:** The user enters a username and password and selects a role (Admin or Worker). This data is saved in users.csv.

- **Login:** The program reads users.csv. It matches the username and password.
    - If the Role is **Admin**, the full dashboard opens.
    - If the Role is **Worker**, a restricted dashboard opens (Worker cannot add food items).

### *B. Menu Management (menu.cpp)*

- **View Menu:** Reads menu.csv and displays Item Name, Price, and Available Stock.
- **Add Item (Admin Only):** Admin can add new food items. The program appends this data to the CSV file.
- **Stock Logic:** Whenever an order is placed, the system checks if enough stock is available. If yes, it automatically subtracts the quantity from the file.

### *C. Table Management (tables.cpp)*

- The restaurant has numbered tables. Each table has a status: **"Free"** or **"Busy"**.
- When a customer sits for Dine-In, the waiter marks the table as "Busy".
- When the bill is paid, the system automatically resets the table status to "Free".

### *D. Taking Orders*

The system handles two types of orders:

1. **Dine-In Order (dine_in_order.cpp):**
    - Requires a Table Number.
    - Checks if the table is valid.
    - User enters Order ID and Food Item name.
    - **Validation:** System checks getItemPrice(). If the item exists and stock is available, the order is saved to dine_in_order.csv.
2. **Online Order (online_order.cpp):**
    - Does not require a table.
    - Takes Customer Name and Phone Number instead.
    - Updates stock and saves data to online_order.csv.

### *E. Billing & Checkout (billing.cpp)*

- The user enters the **Order ID**.
- The system searches for that ID in the order files.
- It calculates the **Grand Total** (Price × Quantity).
- **Payment:** If the user confirms payment:
    - The Bill Status changes to "Paid" in bills.csv.
    - If it was a Dine-In order, the **Table is automatically released (set to Free)**.

## *5. How to Run the Project*

To compile and run this project, you need a C++ compiler (like G++ or Dev-C++).

**Steps:**

1. Open the terminal/command prompt in the project folder.

## Features Summary

- **Security:** Role-based login (Admin/Worker).
- **Data Saving:** All records (Menu, Users, Orders) are saved in CSV files.
- **Stock Control:** Auto-deduction of stock when items are sold.
- **Table Tracking:** Live status of restaurant tables.
- **Logs:** Keeps a text file record of who logged in and when.

## Flow Chart