 **SIMPLE FOOD ORDER SYSTEM** 

**PROJECT REPORT**

Submitted by

# I.JEYASEELAN (21CS037)

*in partial fulfilment for the award of the degree* of

**BACHELOR OF ENGINEERING**

**in**

COMPUTER SCIENCE AND ENGINEERING

**P.S.R. ENGINEERING COLLEGE, SIVAKASI –**

**626140**

(An Autonomous institution, Affiliated to Anna University,

Chennai)

# ANNA UNIVERSITY: CHENNAI 600 025

ABSTRACT

Simple Food Ordering System is designed to provide a user-friendly and efficient platform for customers to place food orders in a restaurant setting. This system caters to both vegetarian and non-vegetarian preferences, offering a variety of dishes with an intuitive menu interface. Users can navigate through the menu, select desired items, specify quantities, and place orders seamlessly.

The primary objective of this system is to streamline the food ordering process, enhancing the overall customer experience. By implementing a straightforward user interface and leveraging core programming concepts in C++, the project aims to create a functional prototype that highlights the key functionalities of a food ordering system.

ACKNOWLEDGEMENT

We extend our gratitude to the individuals, institutions, and organizations whose contributions have played a crucial role in the successful development and implementation of the Employee management system.

**Open Source Community:**

Our project has benefitted from the vast knowledge and resources offered by the open-source community. Various libraries, frameworks, and coding solutions have been crucial in the development.

**Institutional Support:**

We acknowledge the institutional support provided by projects. The infrastructure, resources, and encouragement extended by the institution have been fundamental to the successful completion of the project.

**Faculty Advisors:**

We express our sincere thanks to our faculty advisors for their invaluable guidance, unwavering support, and constructive feedback throughout the project. Their expertise has been instrumental in shaping the direction of the system.

**Departmental Staff:**

We appreciate the cooperation and collaboration of the departmental staff who actively participated in the testing phase and shared their domain knowledge. Their practical insights have been pivotal in aligning the system with the specific needs and workflows of the department.

TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **PAGE NO** |
| 1 | INTRODUCTION |  |
| 1.1 | BACKGROUND |  |
| 1.2 | OBJECTIVES |  |
| 2 | LITERATURE REVIEW |  |
| 3 | PROJECT OVERVIEW |  |
| 4 | METHODOLOGY |  |
| 4.1 | SYSTEM ARCHITECTURE |  |
| 4.2 | TECHNOLOGIES USED |  |
| 5 | SYSTEM FEATURES |  |
| 5.1 | VEG AND NONVEG OPTIONS |  |
| 5.2. | DYNAMIC MENU DISPLAY |  |
| 5.3 | USER INPUT FOR DISH SELECTION |  |
| 5.4 | QUANTITY INPUT |  |
| 5.5 | BILL GENERATION |  |
| 5.6 | ORDER HISTORY |  |
| 6 | IMPLEMENTATION |  |
| 7 | RESULTS AND OUTCOMES |  |
| 7.1 | USER INTERFACE SCREENSHOTS |  |
| 7.2 | PERFORMANCE METRICS |  |
| 8 | REFERENCES |  |
| 9 | APPENDIX |  |
| 10 | CONCLUSION |  |

1. **Introduction**

* 1. **Background**

The foundation of a Simple Food Ordering System (FOS) is grounded in the demand for streamlined and efficient food ordering processes in various establishments. As the food service industry continues to evolve, there is a growing need for technology-driven solutions that simplify order placement, enhance customer experiences, and optimize operational efficiency.

**1.2** **Objectives**

The core objectives of the Simple Food Ordering System project encompass the creation of an intuitive user interface tailored for food ordering. The project aims to implement essential features such as menu display, order placement, order modification, and cancellation functionalities. Additionally, the goal is to seamlessly integrate the system into the development environment, ensuring compatibility with platforms like Code: Blocks. The project also seeks to prioritize accessibility and simplicity, striving to facilitate the efficient organization and management of food orders while providing a user-friendly experience.

## 2.Literature Review

The literature review on Simple Food Ordering Systems underscores the evolving landscape of the food service industry through technology integration. Studies by Smith and Brown highlight the pivotal role of online platforms, emphasizing user interface design and order processing. Patel and Gupta explore technological solutions for enhancing overall operational efficiency, encompassing order processing and inventory management. Lee and Kim delve into the user experience factors crucial for mobile food ordering applications, emphasizing usability and design principles. Additionally, research by Sharma and Singh sheds light on challenges faced by small businesses in adopting food ordering systems, addressing issues of affordability and technical expertise. Chen and Wang discuss the advantages of integrating food ordering systems with Point-of-Sale systems, emphasizing the potential for improved order accuracy and overall restaurant performance. Collectively, these studies offer insights into critical considerations for designing a user-friendly and efficient Simple Food Ordering System.

## 3. Project Overview

The Simple Food Ordering System is designed to provide a user-friendly and seamless experience, emphasizing simplicity in its execution. Within the Code: Blocks environment, users can effortlessly perform essential functions like Displaying veg and nonveg dishes based on the option, Bill generation and save history. The project prioritizes delivering a practical and efficient tool for managing well-organized food orders within the system. The primary goal is to offer an accessible solution that streamlines the food ordering process, ensuring a smooth and intuitive experience for users interacting with the food order system.

## 4. Methodology

**4.1 System Architecture**

The system architecture of the Simple Food Ordering System is centered on the design and organization of its components. It focuses on the structuring of the menu class, encapsulating food item details, and incorporating essential functionalities like adding, displaying, searching, and canceling orders. This architecture guarantees a cohesive and systematic framework for an efficient food ordering system, providing a well-organized structure for managing food-related operations seamlessly.

**4.2 Technologies Used**

The project food order system C++ as the primary programming language for implementation within the Code: Blocks environment. C++ is chosen for its capability to strike a balance between performance and simplicity, making it suitable for handling the complexities of details operations. Code: Blocks, serving as the development environment, offers a familiar and accessible platform for users, contributing to the overall user-friendly design and integration of the Simple food ordering system. The combination of these technologies ensures efficiency, maintainability, and ease of use in managing processes within the employee details management.

## 5. System Features

**5.1. Veg and Non-Veg Options:**

Users can choose between vegetarian and non-vegetarian dishes, catering to diverse dietary preferences.

**5.2. Dynamic Menu Display:**

The system dynamically displays the menu, providing an updated list of available dishes under each category.

**5.3. User Input for Dish Selection:**

Users are prompted to input their choice of dishes by selecting a corresponding number from the displayed menu.

**5.4. Quantity Input:**

For selected dishes, users can input the quantity they wish to order, enhancing customization based on their preferences

**5.5. Bill Generation:**

Bills for each order are generated and stored in individual text files, providing a record of the order details.

**5.6. Order History:**

The system logs the date and time of each order in the order history, enhancing transparency and record-keeping.

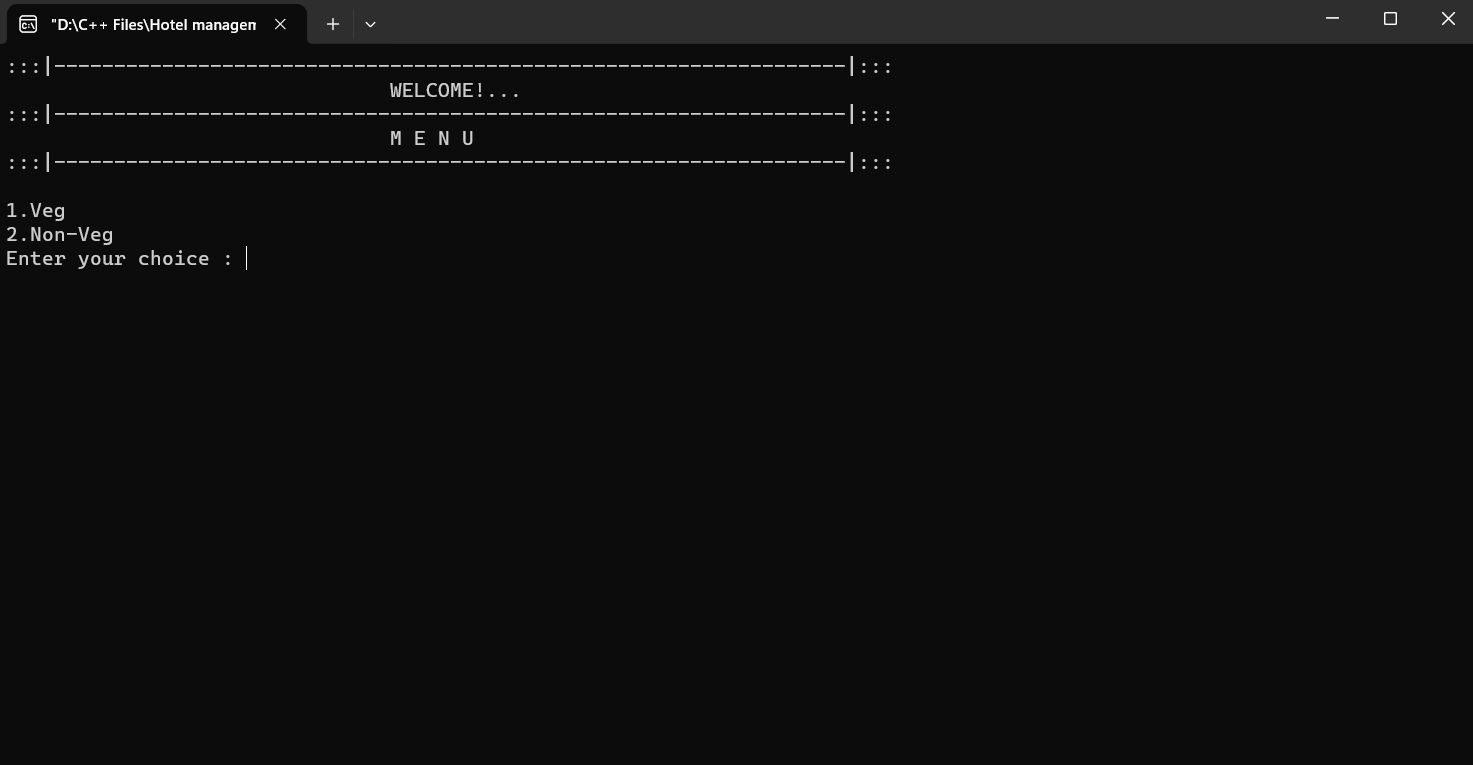
## 6. Implementation

The implementation phase involves coding the project in C++ within the Code::Blocks environment. The development process adheres to the established methodology, focusing on simplicity, efficiency, and user friendly design.

## 7. Results and Outcomes

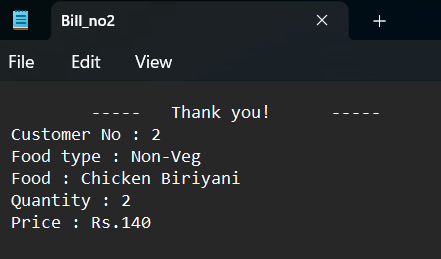
**7.1 User Interface Screenshots**

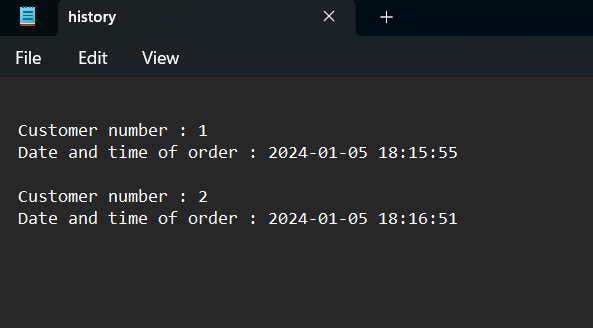
Screenshots showcase the system's user interface, providing a visual representation of the project's design and usability. These images demonstrate the simplicity and accessibility of the system.



****



****

****

**7.2 Performance Metrics**

Performance metrics for the Simple Food Order System aim to evaluate its efficiency and responsiveness in handling key operations within the retail environment. These metrics encompass

## 8. References

* GeeksforGeeks. (n.d.). "C Programming Language." Retrieved from https://www.geeksforgeeks.org/c-plus-plus/
* JavaTpoint (n.d.). "Community-driven Question and Answer site." Retrieved from https://stackoverflow.com/

## 9. Appendix

**9.1 Code Snippets**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <sstream>

#include <string>

#include <chrono>

#include <ctime>

using namespace std;

void printline(){

cout<<":::|------------------------------------------------------------------|:::"<<endl;

}

class Menu{

public:

string order\_no;

int choice,quantity,cus\_no=1;

void veg(){

printline();

cout<<"\t\t\t\tVeg Dishes"<<endl;

printline();

cout<<"\n1.Sambar - Rs.50\n2.Rasam - Rs.40\n3.Pongal - Rs.30\n4.Vegetable Biriyani - Rs.70\n5.Cancel\nEnter your choice : ";

if(cin>>choice){

try{

if(choice>=1 && choice<=5){

if(choice!=5){

cout<<"Enter the quantity : ";

cin>>quantity;

}

switch(choice){

case 1:

place\_order("Veg","Sambar",50);

break;

case 2:

place\_order("Veg","Rasam",40);

break;

case 3:

place\_order("Veg","Pongal",30);

break;

case 4:

place\_order("Veg","Vegetable Biriyani",70);

break;

case 5:

cout<<"";

return;

break;

default:

cout<<"Please! Enter correct option...";

break;

}

}else{

throw choice;

}

}catch(int choice){

cout<<choice<<" isn't a valid operation! Enter correct operation!...\n"<<endl;

veg();

}

}else{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout<<"Invalid input! Please enter a number.\n"<< endl;

veg();

}

}

void non\_veg(){

printline();

cout<<"\t\t\t\tNon-veg Dishes"<<endl;

printline();

cout<<"\n1.Chicken65 250g - Rs.80\n2.Muttan Biriyani - Rs.90\n3.Chicken Biriyani - Rs.70\n4.Cancel\nEnter your choice : ";

if(cin>>choice){

try{

if(choice>=1 && choice<=4){

if(choice!=5){

cout<<"Enter the quantity : ";

cin>>quantity;

}

switch(choice){

case 1:

place\_order("Non-Veg","Chicken65 250g",80);

break;

case 2:

place\_order("Non-Veg","Muttan Biriyani",90);

break;

case 3:

place\_order("Non-Veg","Chicken Biriyani",70);

break;

case 4:

return;

break;

default:

cout<<"Please! Enter correct option...";

break;

}

}else{

throw choice;

}

}catch(int choice){

cout<<choice<<" isn't a valid operation! Enter correct operation!...\n"<<endl;

non\_veg();

}

}else{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout<<"Invalid input! Please enter a number.\n"<< endl;

non\_veg();

}

}

void place\_order(string f\_type,string food,int price){

stringstream str;

str<<cus\_no++;

str>>order\_no;

string file = "Bills/Bill\_no"+order\_no+".txt";

ofstream bill(file);

fstream history("History/history.txt",ios::app);

auto currentTimePoint = chrono::system\_clock::now();

time\_t currentTime = chrono::system\_clock::to\_time\_t(currentTimePoint);

tm\* timeStruct = localtime(&currentTime);

char timeString[100];

strftime(timeString, sizeof(timeString), "%Y-%m-%d %H:%M:%S", timeStruct);

bill<<"\t-----\tThank you!\t-----\nCustomer No : "<<order\_no<<"\nFood type : "<<f\_type<<"\nFood : "<<food<<"\nQuantity : "<<quantity<<"\nPrice : Rs."<<price\*quantity<<endl;

history<<"\nCustomer number : "<<order\_no<<"\nDate and time of order : "<<timeString<<endl;

bill.close();

history.close();

greet();

}

void greet(){

cout<<"\t\t\tThank you!\n\t\tOrder placed successfully!..."<<endl;

}

};

int main()

{

Menu M;

int ch;

do{

printline();

cout<<"\t\t\t\tWELCOME!..."<<endl;

printline();

cout<<"\t\t\t\tM E N U"<<endl;

printline();

cout<<"\n1.Veg\n2.Non-Veg\nEnter your choice : ";

if(cin>>ch){

try{

if(ch >= 1 && ch <= 2) {

switch (ch) {

case 1:

M.veg();

break;

case 2:

M.non\_veg();

break;

default:

cout << "Please! Enter correct option..." << endl;

break;

}

}else{

throw ch;

}

}catch(int ch){

cout<<ch<<" isn't a valid operation! Enter correct operation!..."<<endl;

}

}else{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout<<"Invalid input! Please enter a number."<< endl;

}

}while(true);

return 0;

}

**9.2 User Manuals**

Getting Started:

- Install Code: Blocks IDE.

- Download and open the provided project files.

Placing Food Orders:

- Choose options for food ordering.

- Select from a variety of Veg and Non-Veg dishes.

Veg dishes;

1. Sambar - Rs.50

2. Rasam - Rs.40

3. Pongal - Rs.30

4. Vegetable Biriyani - Rs.70

5. Cancel

Non-veg dishes;

1.Chicken65 250g - Rs.80

2.Muttan Biriyani - Rs.90

3.Chicken Biriyani - Rs.70

4.Cancel

**10. Conclusion**

In conclusion, the Simple Food Ordering System effectively achieves its goals by providing users with a user-friendly and streamlined solution for placing and managing food orders. The system's focus on simplicity, coupled with essential features, positions it as an efficient tool for maintaining well-organized order records within the Code: Blocks environment. The implementation of this Simple Food Ordering System marks a significant step in modernizing our organization's approach to food service. Through this system, we have successfully tackled challenges in the ordering process, improved overall operational efficiency, and contributed to a more convenient and enjoyable experience for our customers.