Bistro Bee

Project submitted to the
SRM University – AP, Andhra Pradesh

Submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

School of Engineering and sciences

Submitted By

Siddi Sathvik Kuruba (AP23110010516)

Ponnuru Venkata Naga Sai Yaswanth (AP23110010528)

Rajesh ponnada(AP23110010519)

Ratan Raj (AP23110010515)

Under the guidance of

Mrs.Karnena Kavitha Rani

Department of Computer Science and Engineering

SRM University-AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
[November,2024]

Department of Computer Science and Engineering

SRM University, AP

CERTIFICATE

This is to certify that the Project report entitled **“Bristro bee”** is being submitted by **Group-,** a student of Department of Computer Science and Engineering, SRM University, AP, in partial fulfillment of the requirement for the degree of **“B.Tech (CSE)”** carried out by her/his during the academic year 2024-2025.

Signature of the Supervisor

Signature of Head of the Dept.

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. I am extremely grateful and express my profound gratitude and indebtedness to my project guide, **Mrs. Karnena Kavitha Rani,** Department of Computer Science & Engineering, SRM University, Andhra Pradesh, for her kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

Table of Contents

1.INTRODUCTION

1.1. Background

In today's busy world, convenience and speed are key for food delivery app like bristro is the ideal way to satisfy your hunger effortlessly. Whether you're craving a light snack, a fine dining experience, or a late-night treat, our app delivers your favourite meals right to your doorstep. Featuring a user-friendly design, a wide selection of restaurants, our platform transforms the way you enjoy food. Created to save time, simplify the process our app makes delicious meals just a few clicks away, bringing more flavour and convenience to your life.

1.2. Project Significance

1.2.1 Technological Relevance

This project demonstrates how C++ and the concept of Object-Oriented Programming (OOP) can model a complex food delivery system. By using OOP, we create a modular and main table codebase to implement features like wide range of selection of restaurants, variety of foods, and tastes of various restaurants in a single order.

1.2.2. Real-World Application

The food delivery C++ project addresses the growing need for efficient and reliable online food ordering systems. Utilizing C++ for its high performance, the system is built to handle large volumes of orders and coordinate deliveries effectively. It integrates with restaurant databases, offering customers real-time updates and a wide selection of meals. The project also ensures safe delivery, making it an essential tool for food delivery services seeking to improve both user experience and operational efficiency.

1.2.3. Educational Value

This project deepens understating of C++ and OOPS principles such as classes, objects, Encapsulation and polymorphism by applying them to real world entities like restaurants, delivery persons, scalable code design, etc.,.

1.3. Scope

This project focuses on implementing core functionalities without backend elements such as real time location tracking and payment processing.

Key features include:

1.3.1 Ordering from different restaurants: A mechanism which allows user to order wide range of items from different restaurants in a single order

1.3.2 Pricing Model: A module that calculates the cost of all the items ordered and give the final cost.

1.4 Purpose

The objective of this project is to create a fundamental prototype for a food delivery system, named *Bistro bee*, using C++ and Object-Oriented Programming (OOP) techniques. The program is designed to replicate key aspects of a real-world food delivery service, including placing orders across multiple restaurants, managing different menu items, and assigning a delivery person to fulfil each order. By simulating these core features, this project demonstrates the essential functionalities of a food delivery platform in a simplified manner.

2.Literature Review

In developing a bristro program that demonstrates object-oriented programming (OOP) principles, understanding the technological landscape and identifying gaps is crucial. This literature review covers related technologies, existing gaps, and how the proposed app addresses specific knowledge gaps.

2.1 Related Technologies

A food delivery program typically integrates several modern technologies and concepts, including:

- **Geolocation Services:** Technologies like GPS and mapping APIs (Google Maps, Open Street Map) are essential for delivery address, route optimization and real time location tracking.
- **Backend and Database systems:** Platforms such as Node.js, Django, or Flask are commonly used for the backend, while databases like MySQL, Postgre SQL, etc., are used to save the sign up details of the user and also to save the history of a user.
- **Object-Oriented Programming (OOP):** This programming paradigm is pivotal in building modular and scalable applications. OOP concepts like classes, objects, inheritance, polymorphism, and encapsulation provide a framework for creating reusable and maintainable code.
- **Micro services Architecture:** Many modern ride-sharing apps use micro services to separate functionalities such as user management, ride-matching, and payment processing, which enhances scalability and reliability.
- **Machine Learning for Pricing:** Algorithms for surge pricing, demand prediction, and user matching utilize machine learning techniques for better accuracy and optimization.

2.2 Existing Gaps

While bristro bee have become highly sophisticated, several challenges remain, especially from an educational perspective and in terms of implementing simpler models:

- **Implementation Complexity:** Most modern food delivery platforms are built with intricate, highly efficient codebases that integrate advanced technologies such as real-time tracking, distributed systems, and AI-driven recommendations. For those new to programming, this complexity can make understanding the system difficult and intimidating.
- **Lack of Emphasis on OOP Fundamentals:** Many educational examples focus on functional aspects but often overlook key Object-Oriented Programming (OOP) principles like encapsulation, polymorphism,etc.,. By using a food delivery app as a practical example, these OOP concepts can be clearly demonstrated in a relatable context.
- **Focus on Scalability over Simplicity:** While established food delivery apps emphasize scalability and system optimization, these features may not be essential for small-scale or beginner-level projects. Simplified versions of food delivery apps can provide a clearer focus on core functionalities, helping learners build a solid understanding without the overwhelming complexity of large-scale optimizations.

2.3 Knowledge Gaps Addressed

The proposed project aims to bridge several gaps by focusing on simplified, educational implementations of key food delivery features using OOP principles.Some of them are as follows:

Classes: Representing restaurents.
Ex:- class Restaurant {
public:
    string name;
    vector<MenuItem> menu;
};

Objects: It is an instance of a class.
Ex:- Restaurant pizzaPalace("Pizza Palace", ...);

Encapsulation: The process of hiding implementation details and showing only necessary details.
Ex:-

```
class Order {

private:

    double totalCost;

public:

    void addItem(...);

};
```

Polymorphism:

Polymorphism allows the same method to behave differently based on the object type, typically via method overriding.
Example:

```
virtual void showDetails() const = 0;

void showDetails() const override;
```

2.3 Balancing Scalability and Simplicity in Educational Projects

One of the key challenges in teaching software development using food delivery apps is finding the right balance between **scalability** and **simplicity**. Real-world applications are designed to handle large-scale operations, including **multi-server architectures**, **load balancing**, and

fault tolerance .However these elements are often unnecessary for educational purposes and can make projects overly complex.

To help students focus on core programming concepts, **simplified models** of food delivery systems can exclude complex features like advanced database management or traffic handling. This approach allows learners to grasp the basics of application functionality without being distracted by scalability concerns. By limiting the scope of the system to manageable components, students can focus on understanding essential concepts like OOP and basic system design.

3.Problem Statement

Problem statement for food delivery (Bistro bee) code functionality

The rising demand for delivery services and the need for convenience have our app essential in today's world. However, managing various order requests, handling different user needs, and ensuring smooth operations can be complex. This complexity arises due to the diverse functionalities such as ordering food from different restaurants, bill calculation, assigning delivery person, and payment handling.
In this context, a well-structured software solution is needed that uses object-oriented programming (OOP'S) principles, such as classes, objects, encapsulation, polymorphism. These principles help in organizing the system into manageable components, ensuring code reusability, flexibility, and secure programming.

Classes and objects: The program can be structured into different classes like restaurant, user, order, etc., where each class represents a specific entity, and objects represent instances of these entities.

Encapsulation: The process of hiding implementation details and showing only necessary details like order details,etc.,.

Polymorphism: The process handles orders, billing, allocation of a delivery person, etc.,. This problem requires attention because it directly impacts the user experience and operational efficiency. A secure solution built on these OOP concepts will improve the scalability, maintainability, and overall functionality of the food delivery program.

4.Objectives

Objectives are broadly divided into two

- Primary Objectives
- Secondary objectives

4.1. Primary Objectives:

4.1.1. User Account Management: To allow users to log in using already saved information.

4.1.2. Total bill calcution : To calculate the total bill of the order of the user.

4.1.3. Driver Matching Simulation: To simulate the assignment of drivers to pick up the orders from the restaurants user has ordered and delivery it to the user.

4.2. Secondary Objectives:

4.2.1. Error Handling and Validation

To check whether the user logged in with the details present in the data base and if the username and password does not match with any user name and password in the data base then it will show invalid username or password.

4.2.2. User Experience Simulation

Provide a smooth interaction, allowing users to select the food from various restaurants.

5. Methodology

The development of the ride-sharing app involves several core components, each utilizing specific programming techniques and data structures. This section describes the key methodologies used in the project.

5.1 Account Management:

For handling user data (e.g., driver and rider profiles), file I/O in C++ is implemented using the stream library.
This allows the app to:

- **Create, read, and write files:** User information (username, password) is stored in text files, with basic read functions to manage account data.
- **Simplified Security Measures:** Passwords are stored in plain text for simplicity, but in a real-world scenario, encryption (e.g., hashing) would be required to protect user credentials.
- **Efficient Data Access:** Using file I/O enables persistent storage, allowing user data to be retrieved and updated across multiple sessions.

5.3 Total cost Calculation:

The program uses a static total cost model to determine the total cost of all items ordered by the user from all restaurants:

Item Prices.at(item.first) * item. second

Potential Extensions: Future versions of the program could implement dynamic pricing based on factors like demand and distance between the user and restaurant.

6. Implementation:

1. Class Design and Functional Roles:

- **ServiceEntity**: Serves as an abstract base class with a virtual function for displaying details.
- **MenuItem:** Represents individual menu items, including their names and prices.
- **Restaurant:** Represents a restaurant, containing its name and menu details.
- **DeliveryPerson:** Represents a delivery agent identified by their name.
- **Order:** Handles the items in an order, calculates the total cost, and assigns a delivery person.
- **Bistrobee:** Acts as the main interface, managing restaurants, orders, and delivery processes.
- **Login System:** Handles user authentication using a file-based approach to store user credentials.

2. **Required Libraries:**

- **Core logic:** iostream, vector, string, map are the header files that are used to basic logic.
- **File handling:** fstream for managing persistent user data.
- **Randomization:** ctime and cstdlib for randomly assigning delivery personnel.

7. Implementation Plan

1. Class Creation:

- Design the **ServiceEntity** class as a foundation to ensure consistent functionality across derived classes.
- Implement **MenuItem**, **Restaurant**, and **DeliveryPerson** classes to manage menu and delivery systems.
- Develop the **Order** class to process customer orders, calculate costs, and manage delivery assignments.
- Build the **Bistrobee** class to handle the overall app workflow, such as displaying restaurants and placing orders.
- Create the **LoginSystem** class to securely manage user authentication with a file-based backend.

2. Data Setup:

- Initialize the **Bistrobee** with sample restaurants, menus, and delivery agents.
- Populate the **LoginSystem** with sample user data for login functionality.

3. Authentication Process:

- Present a login interface where users can enter credentials.
- Verify user details against data stored in a **users.txt** file.
- If authentication succeeds, greet the user by their username.

4. **Order Workflow:**

- Present a list of restaurants to the user.
- Allow selection of menu items along with quantities from the chosen restaurant.
- Dynamically calculate the total cost of selected items and manage the order.

5. **Delivery Handling:**

- Assign a delivery person randomly to each order.

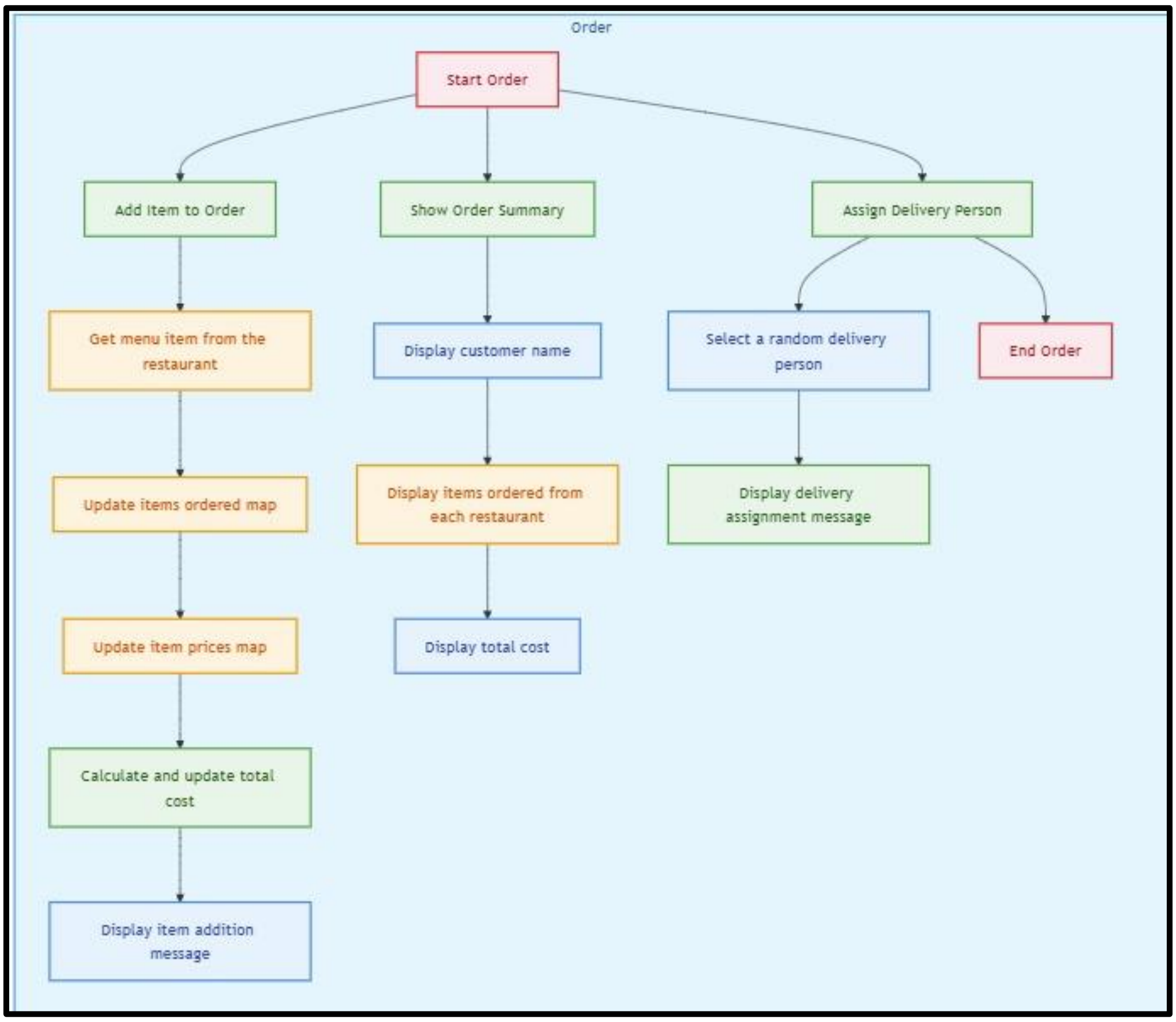
6. **Order Summary:**

- Display a detailed summary of the order, including the selected restaurant, items, quantities, and total cost.

7. **File Operations:**

- Use file I/O operations for persistent storage of user credentials and login details.

8. **Flow chart:**


Order
Start Order
Add Item to Order
Show Order Summary
Assign Delivery Person
Get menu item from the restaurant
Display customer name
Select a random delivery person
End Order
Update items ordered map
Display items ordered from each restaurant
Display delivery assignment message
Update item prices map
Display total cost
Calculate and update total cost
Display item addition message

8. Discussion

The project aims to create basic food delivery program using Object oriented programming(OOP) concepts in C++,focusing on implementing features like total bill calculation, user ordering food, assigning delivery person to an order. The primary objectives were to demonstrate the practical application of OOP principles and to provide a simple yet functional example of a food delivery system.

1.Comparison of initial objectives

The initial objectives of project included:

1.Implementing a functional food delivery program:

The project successfully delivered a working prototype of a food ordering functionality that allows users to book food from multiple restaurants in a single order, match with delivery persons, and calculate fares based on food ordered and there rates.

The use of OOP principles (e.g., classes, encapsulation, polymorphism) was evident in the modular structure of the code, making it easier to maintain and extend.

2. Demonstrating order-delivery person Matching Logic:

The matching logic was implemented using a vector-based approach, where the program iterates through a list of available delivery persons to find a suitable match. This basic method was effective for small datasets and met the objective of showcasing a simple, beginner-friendly algorithm.

3. Providing a Clear Example of OOP Concepts:

The project effectively used OOP features such as file I/O (for user account management), and encapsulation (for data handling in classes). The modular design made the code clear and easy to understand, aligning with the educational objective of the project.

Overall, the project met its objectives by providing a simple yet practical application of OOP concepts in a real-world context.

8.2 Limitations

Despite the successful implementation and promising results, there are several limitations to the project:

1. Basic delivery person Matching Logic:

The vector-based matching approach is efficient for small datasets but may become inefficient as the number of users and delivery persons increases. In real-world applications, more advanced algorithms (e.g., spatial indexing or machine learning-based matching) are typically used to handle larger datasets and optimize driver-rider pairing.

2. Limited Security Measures:

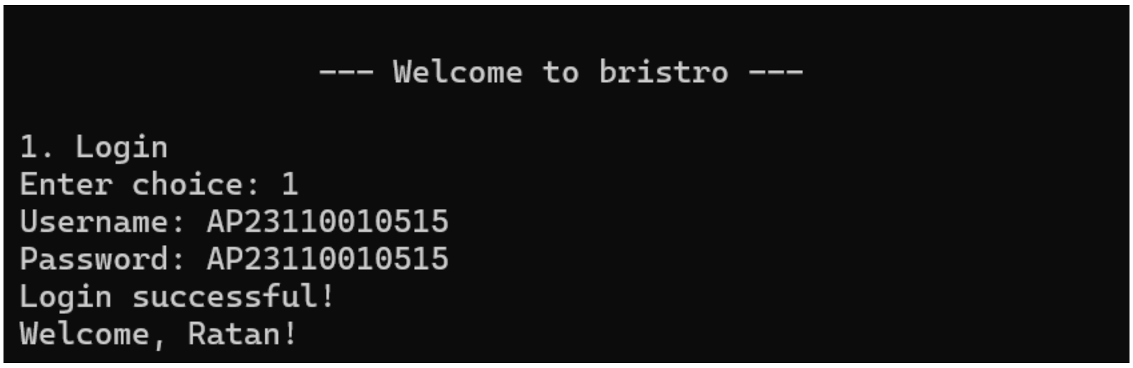
The current implementation uses only entered usernames and passwords, which poses a security risk. In real-world scenarios, user data should be protected using encryption techniques (e.g., hashing passwords) to prevent unauthorized access.

3. No Real-Time Data:

The program does not show the delivery time i.e., in how much time the order will reach the user by the delivery persons.

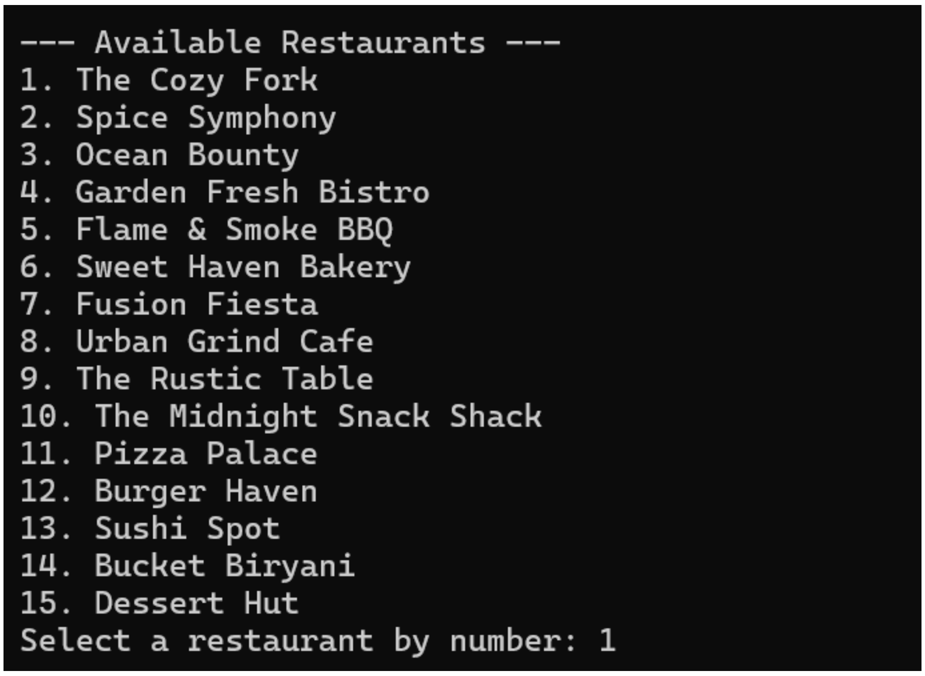
9.User experience Analysis:

9.1 Login:



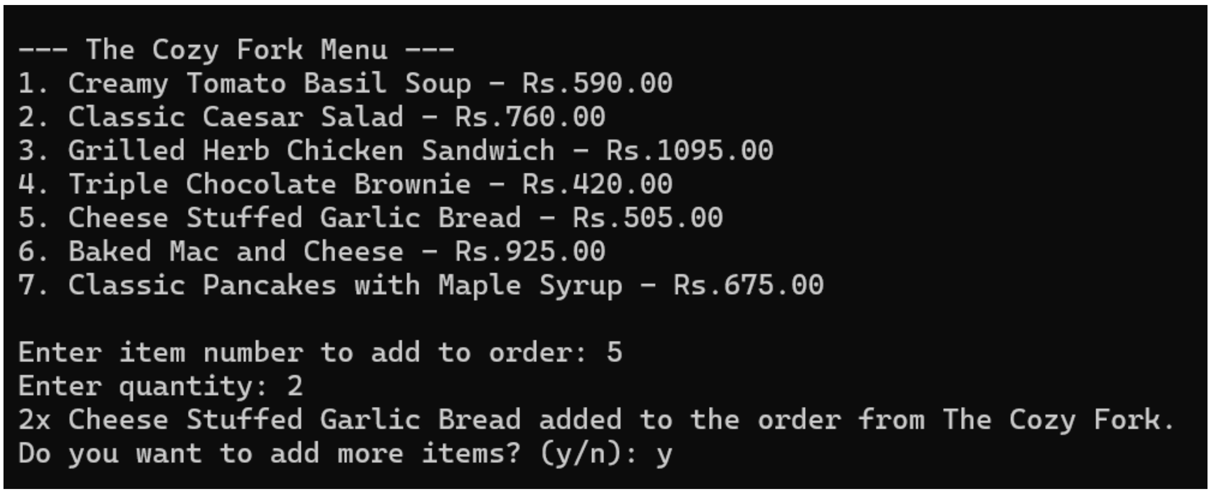
```
                 --- Welcome to bristro ---

1. Login
Enter choice: 1
Username: AP23110010515
Password: AP23110010515
Login successful!
Welcome, Ratan!
```

This part of output asks the user to enter the username and password after successful login it navigates to available restaurents page.



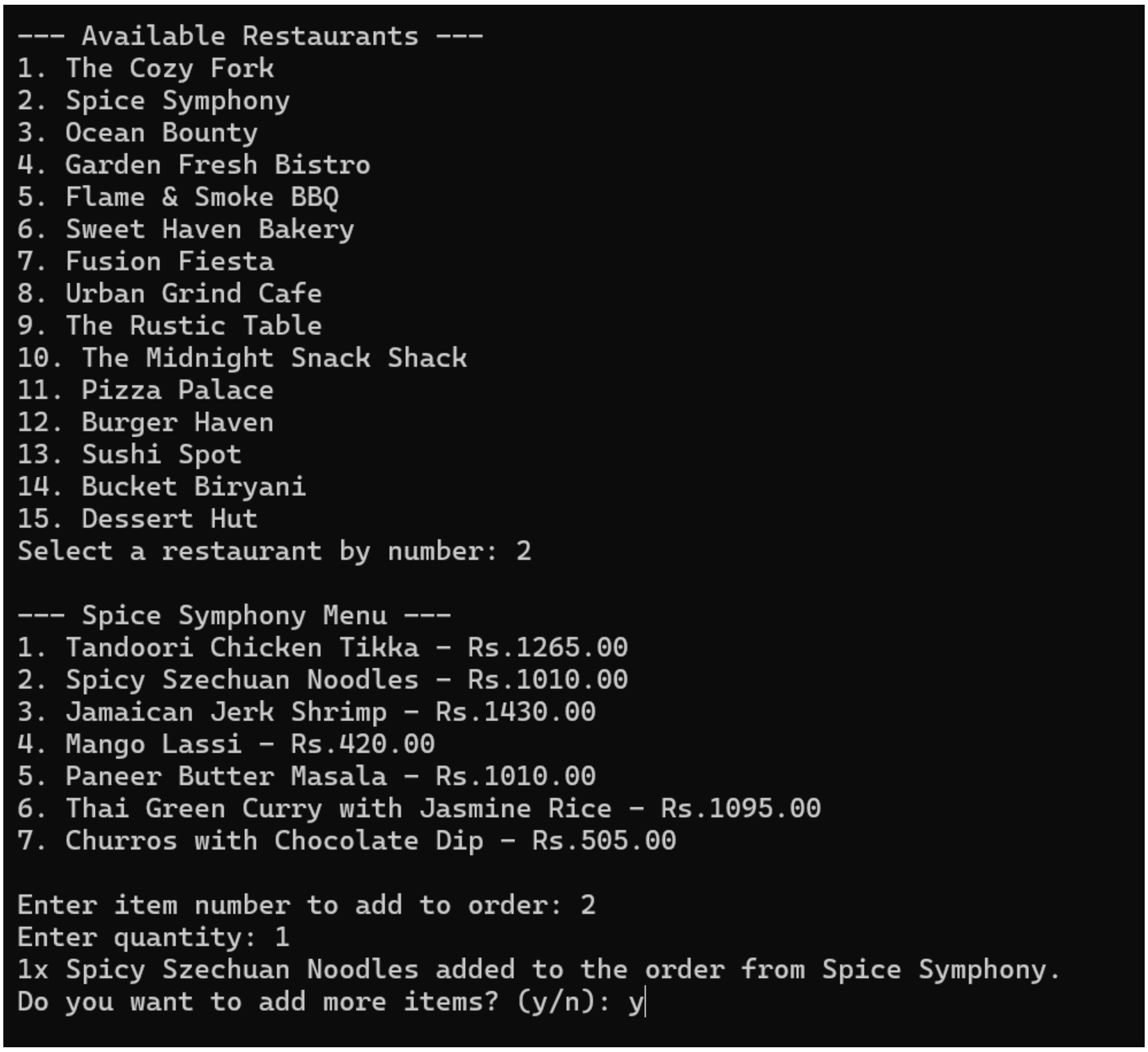
```
--- Available Restaurants ---
1. The Cozy Fork
2. Spice Symphony
3. Ocean Bounty
4. Garden Fresh Bistro
5. Flame & Smoke BBQ
6. Sweet Haven Bakery
7. Fusion Fiesta
8. Urban Grind Cafe
9. The Rustic Table
10. The Midnight Snack Shack
11. Pizza Palace
12. Burger Haven
13. Sushi Spot
14. Bucket Biryani
15. Dessert Hut
Select a restaurant by number: 1
```

This part of the output shows the user the list of available restaurants and asks the user to select a restaurant in the code.



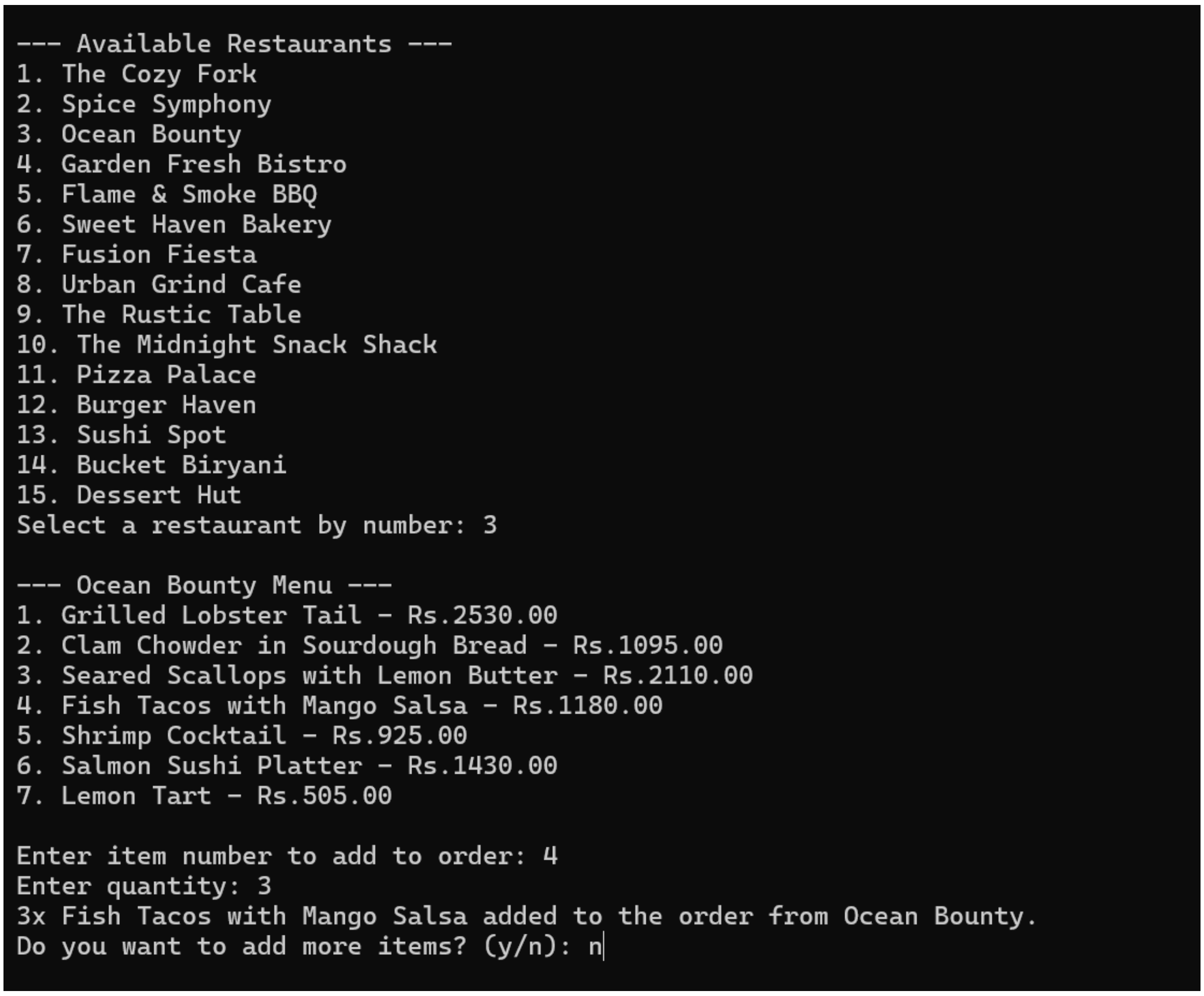
```
--- The Cozy Fork Menu ---
1. Creamy Tomato Basil Soup - Rs.590.00
2. Classic Caesar Salad - Rs.760.00
3. Grilled Herb Chicken Sandwich - Rs.1095.00
4. Triple Chocolate Brownie - Rs.420.00
5. Cheese Stuffed Garlic Bread - Rs.505.00
6. Baked Mac and Cheese - Rs.925.00
7. Classic Pancakes with Maple Syrup - Rs.675.00

Enter item number to add to order: 5
Enter quantity: 2
2x Cheese Stuffed Garlic Bread added to the order from The Cozy Fork.
Do you want to add more items? (y/n): y
```

After selecting the restaurant it redirects to the list of items present in that restaurant.



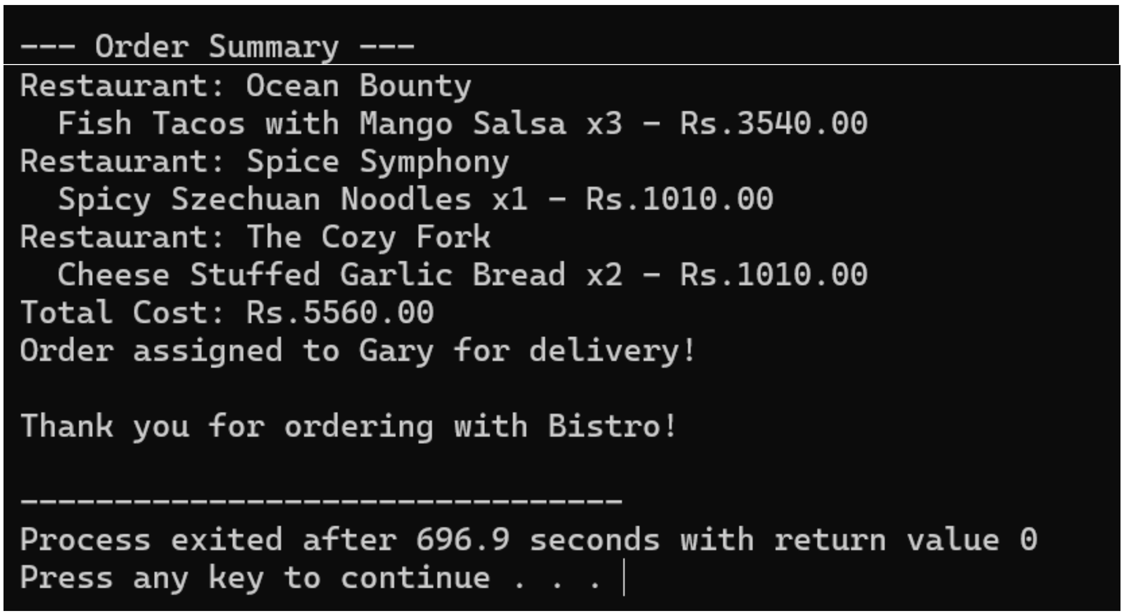
```
--- Available Restaurants ---
1. The Cozy Fork
2. Spice Symphony
3. Ocean Bounty
4. Garden Fresh Bistro
5. Flame & Smoke BBQ
6. Sweet Haven Bakery
7. Fusion Fiesta
8. Urban Grind Cafe
9. The Rustic Table
10. The Midnight Snack Shack
11. Pizza Palace
12. Burger Haven
13. Sushi Spot
14. Bucket Biryani
15. Dessert Hut
Select a restaurant by number: 2

--- Spice Symphony Menu ---
1. Tandoori Chicken Tikka - Rs.1265.00
2. Spicy Szechuan Noodles - Rs.1010.00
3. Jamaican Jerk Shrimp - Rs.1430.00
4. Mango Lassi - Rs.420.00
5. Paneer Butter Masala - Rs.1010.00
6. Thai Green Curry with Jasmine Rice - Rs.1095.00
7. Churros with Chocolate Dip - Rs.505.00

Enter item number to add to order: 2
Enter quantity: 1
1x Spicy Szechuan Noodles added to the order from Spice Symphony.
Do you want to add more items? (y/n): y
```

It again asks the user to select from item from other restaurant and adds it to the final recipet.The user can repeat this process till the user completes his/her order.



```
--- Available Restaurants ---
1. The Cozy Fork
2. Spice Symphony
3. Ocean Bounty
4. Garden Fresh Bistro
5. Flame & Smoke BBQ
6. Sweet Haven Bakery
7. Fusion Fiesta
8. Urban Grind Cafe
9. The Rustic Table
10. The Midnight Snack Shack
11. Pizza Palace
12. Burger Haven
13. Sushi Spot
14. Bucket Biryani
15. Dessert Hut
Select a restaurant by number: 3

--- Ocean Bounty Menu ---
1. Grilled Lobster Tail - Rs.2530.00
2. Clam Chowder in Sourdough Bread - Rs.1095.00
3. Seared Scallops with Lemon Butter - Rs.2110.00
4. Fish Tacos with Mango Salsa - Rs.1180.00
5. Shrimp Cocktail - Rs.925.00
6. Salmon Sushi Platter - Rs.1430.00
7. Lemon Tart - Rs.505.00

Enter item number to add to order: 4
Enter quantity: 3
3x Fish Tacos with Mango Salsa added to the order from Ocean Bounty.
Do you want to add more items? (y/n): n
```

After the user has entered all the items in the final bill it given the order summary and gives the final bill is generated.



```
--- Order Summary ---
Restaurant: Ocean Bounty
  Fish Tacos with Mango Salsa x3 - Rs.3540.00
Restaurant: Spice Symphony
  Spicy Szechuan Noodles x1 - Rs.1010.00
Restaurant: The Cozy Fork
  Cheese Stuffed Garlic Bread x2 - Rs.1010.00
Total Cost: Rs.5560.00
Order assigned to Gary for delivery!

Thank you for ordering with Bistro!

--------------------------------
Process exited after 696.9 seconds with return value 0
Press any key to continue . . .
```

The order is assigned to a delivery person.

10.Future Scope

The future scope for the Bistrobee application code is promising, with several avenues for enhancement and expansion. Integrating advanced features such as real-time order tracking, personalized recommendations based on user preferences, and loyalty programs could significantly improve user engagement. Additionally, incorporating machine learning algorithms for demand forecasting and inventory management can optimize restaurant operations and reduce food waste.

Furthermore, expanding the platform to include a wider variety of cuisines and dietary options, such as vegan or gluten-free meals, would cater to a broader audience. Implementing a rating and review system for both restaurants and delivery personnel can enhance transparency and trust within the community.

The introduction of a mobile application would also facilitate easier access for users, allowing them to place orders on-the-go. Collaborating with local farms and suppliers could promote sustainability and support local economies, aligning with growing consumer preferences for ethical sourcing.

Moreover, exploring partnerships with third-party delivery services could enhance logistics and expand delivery areas. Finally, integrating payment options such as cryptocurrencies could attract tech-savvy users and streamline transactions. Overall, the Bistrobee application has the potential to evolve into a comprehensive food marketplace that not only meets current demands but also anticipates future trends in the food delivery industry.

11.Conclusion

This project set out to develop a simplified food ordering system using C++ and Object-Oriented Programming (OOP) concepts, aiming to demonstrate the practical use of classes, inheritance, polymorphism, and file I/O in a real-world context. The key components of the project included user account management, city route representation, fare calculation, and driver-rider matching, all of which were successfully implemented in a modular and extensible manner.

In conclusion, the Bistrobee application represents a comprehensive and user-friendly platform designed to streamline the process of ordering food from a variety of restaurants while ensuring an efficient delivery system. By integrating a robust login system that authenticates users and personalizes their experience, Bistrobee enhances customer engagement and satisfaction. The application showcases a diverse selection of restaurants and menu items, allowing users to easily navigate and customize their orders. Furthermore, the inclusion of delivery personnel ensures timely and reliable service, making the dining experience seamless from selection to delivery. As a result, Bistrobee not only meets the needs of food enthusiasts but also fosters a sense of community among users and local restaurants, ultimately contributing to a vibrant culinary ecosystem.