A

PROJECT REPORT ON

"CALENDER APPLICATION"

SUBMITTED BY:

Mr. Omkar Dnyaneshwar Gaikwad (2124UCEM1110)

SUBJECT:

C++ PROGRAMMING

Under the guidance of

Miss. ISHWARI TIRSE

Department of Computer Science and Engineering

Sanjivani Rural Education Society's

SANJIVANI UNIVERSITY

KOPARGAON-423603, DIST: AHMEDNAGAR

2024-2025

INTRODUCTION

Today's fast environment means customers require convenient, easy ordering processes for food. In this case, the shop has trouble managing orders and tracking inventory. Inaccuracies and, most of all, delay cause disappointment for the customers . By implementing this system in C++, we can efficiently manage large data sets while maintaining ease of use and accessibility.

CODE

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>

class MenuItem {
public:
    std::string name;
    double price;

    MenuItem(const std::string& n, double p) : name(n),
price(p) {}

    void display() const {
        std::cout << std::setw(20) << name << ": $" << price <<
"\n";
    }
};
```

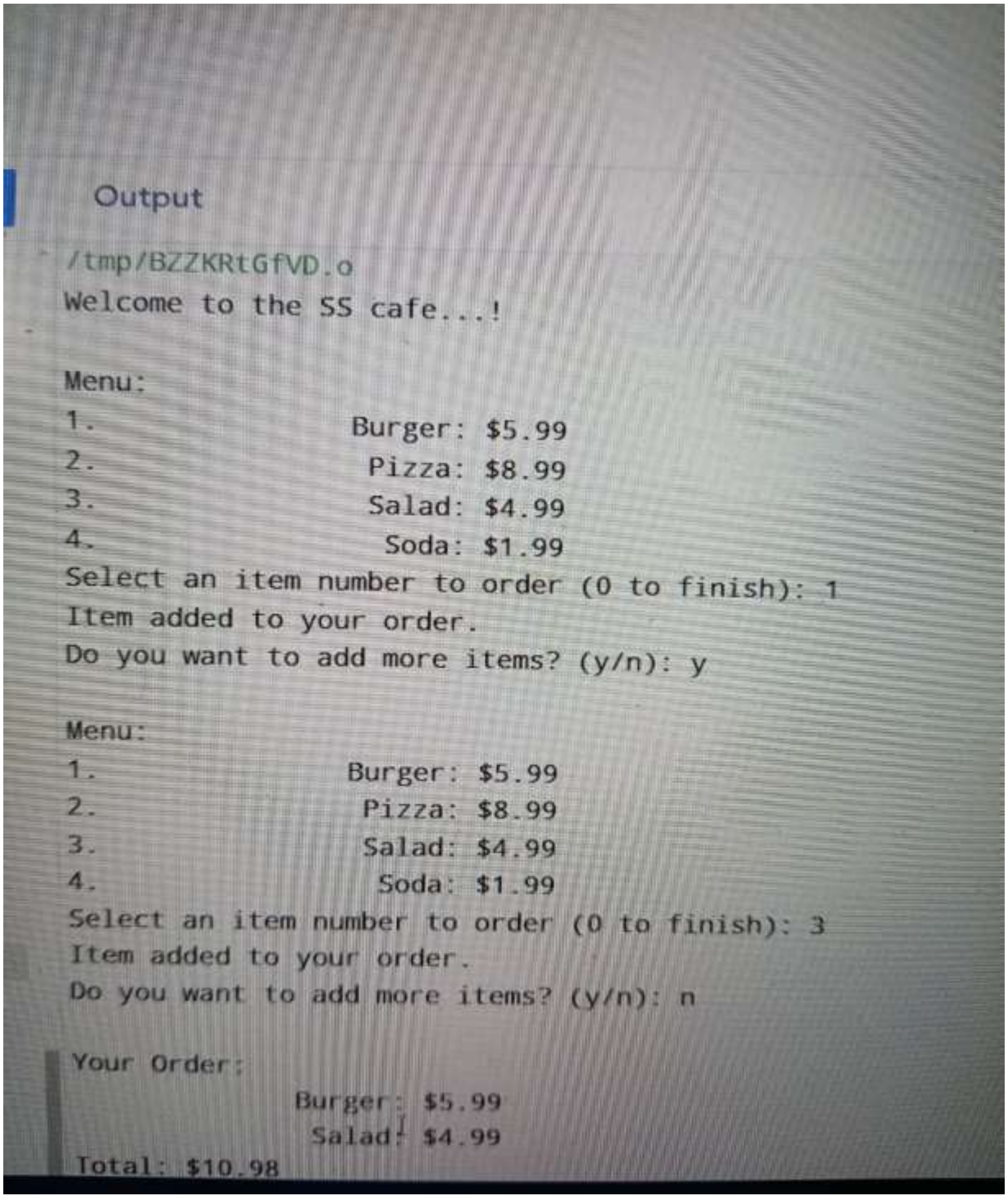
```
class Order {
private:
    std::vector<MenuItem> items;

public:
    void addItem(const MenuItem& item) {
        items.push_back(item);
    }

    double calculateTotal() const {
        double total = 0;
        for (const auto& item : items) {
            total += item.price;
        }
        return total;
    }

    void displayOrder() const {
        std::cout << "\nYour Order:\n";
        for (const auto& item : items) {
            item.display();
        }
        std::cout << "Total: $" << std::fixed <<
std::setprecision(2) << calculateTotal() << "\n";
    }

    const std::vector<MenuItem>& getItems() const {
        return items;
    }
};

class Payment {
public:
    bool processPayment(double amount) {
        std::string cardNumber;
        std::cout << "Enter your card number: ";
        std::cin >> cardNumber;

        // Simulate payment processing
        std::cout << "Processing payment of $" << std::fixed <<
std::setprecision(2) << amount << "...\n";
```

```
        // In a real application, you would integrate with a payment gateway here
        std::cout << "Payment successful!\n";
        return true;
    }
};

class FoodManagementSystem {
private:
    std::vector<MenuItem> menu;

    void loadMenu() {
        menu.push_back(MenuItem("Burger", 150));
        menu.push_back(MenuItem("Pizza", 200));
        menu.push_back(MenuItem("Salad", 100));
        menu.push_back(MenuItem("Soda", 50));
    }

public:
    FoodManagementSystem() {
        loadMenu();
    }

    void displayMenu() const {
        std::cout << "\nMenu:\n";
        for (size_t i = 0; i < menu.size(); ++i) {
            std::cout << i + 1 << ". ";
            menu[i].display();
        }
    }

    void takeOrder() {
        Order order;
        int choice;
        char more;

        do {
            displayMenu();
            std::cout << "Select an item number to order (0 to
finish): ";
```

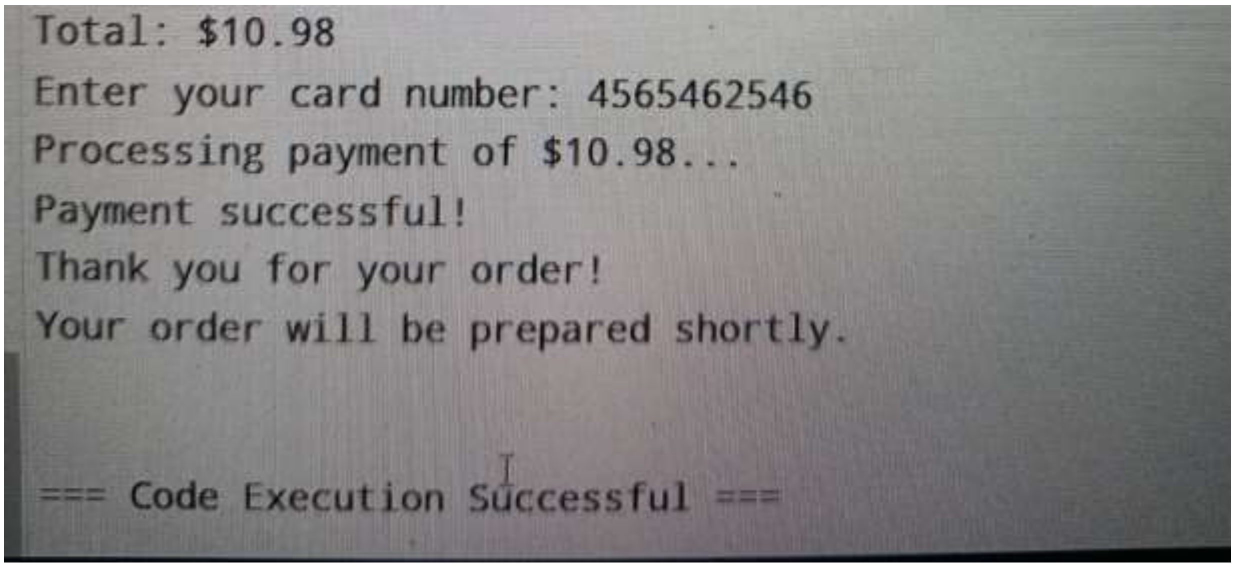
```
        std::cin >> choice;

        if (choice > 0 && choice <= menu.size()) {
            order.addItem(menu[choice - 1]);
            std::cout << "Item added to your order.\n";
        } else if (choice == 0) {
            break;
        } else {
            std::cout << "Invalid choice, please try again.\n";
        }

        std::cout << "Do you want to add more items? (y/n): ";
        std::cin >> more;
    } while (more == 'y');

    order.displayOrder();

    // Payment processing
    Payment payment;
    if (payment.processPayment(order.calculateTotal())) {
            std::cout << "Thank you for your order!\n";
            std::cout << "Your order will be prepared shortly.\n";
        }
    }
};

int main() {
    FoodManagementSystem system;

    std::cout << "Welcome to the SS cafe!\n";
    system.takeOrder();

    return 0;
}
```

OUTPUT

1.With Yes Option:

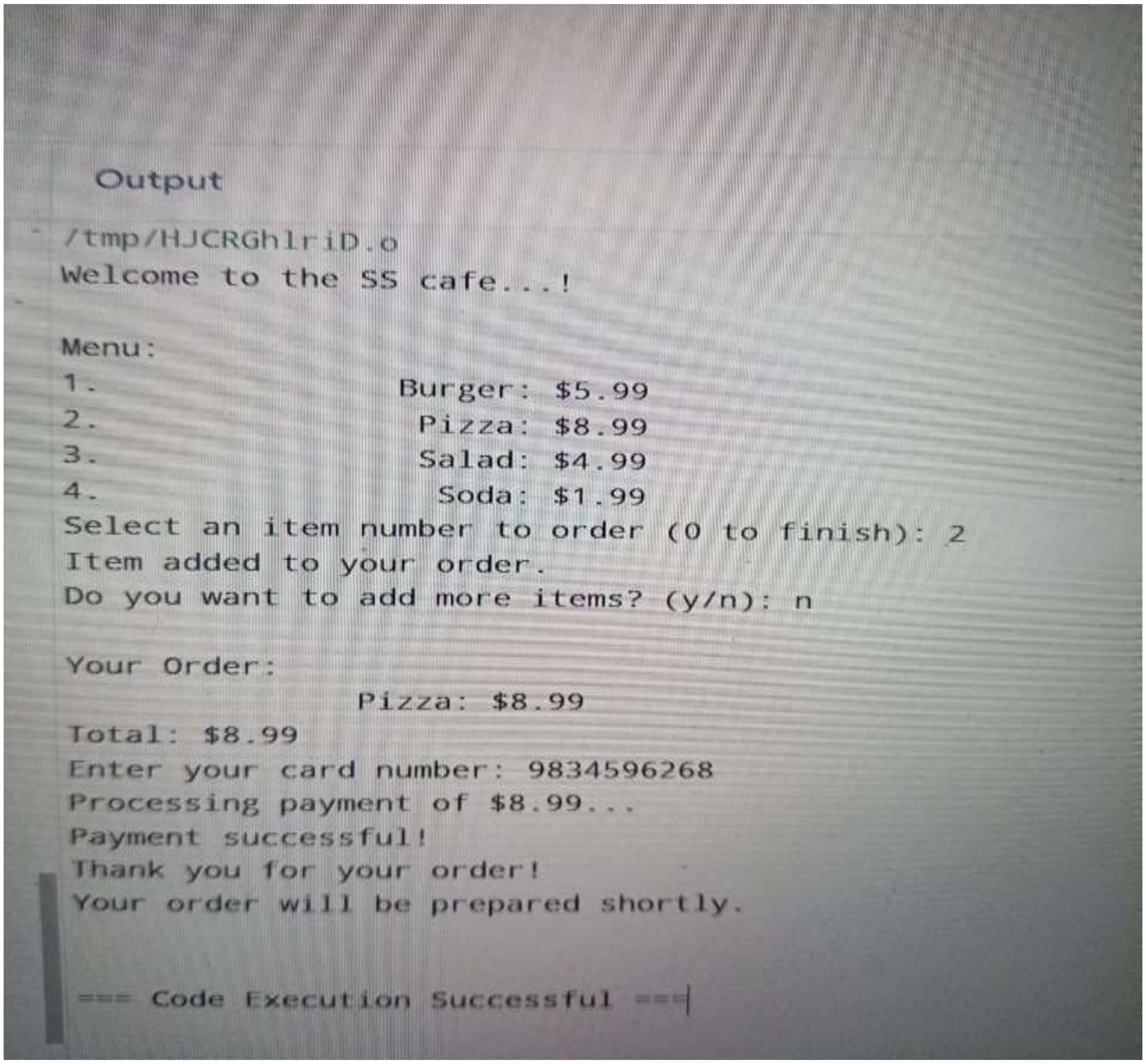


```
Output
/tmp/BZZKRtGfVD.o
Welcome to the SS cafe...!

Menu:
1.                 Burger: $5.99
2.                  Pizza: $8.99
3.                  Salad: $4.99
4.                   Soda: $1.99
Select an item number to order (0 to finish): 1
Item added to your order.
Do you want to add more items? (y/n): y

Menu:
1.                 Burger: $5.99
2.                  Pizza: $8.99
3.                  Salad: $4.99
4.                   Soda: $1.99
Select an item number to order (0 to finish): 3
Item added to your order.
Do you want to add more items? (y/n): n

Your Order:
              Burger: $5.99
               Salad: $4.99
Total: $10.98
```



```
Total: $10.98
Enter your card number: 4565462546
Processing payment of $10.98...
Payment successful!
Thank you for your order!
Your order will be prepared shortly.


=== Code Execution Successful ===
```

2. With No Option:



```
Output
/tmp/HJCRGhlriD.o
Welcome to the SS cafe...!

Menu:
1.                 Burger: $5.99
2.                  Pizza: $8.99
3.                  Salad: $4.99
4.                   Soda: $1.99
Select an item number to order (0 to finish): 2
Item added to your order.
Do you want to add more items? (y/n): n

Your Order:
               Pizza: $8.99
Total: $8.99
Enter your card number: 9834596268
Processing payment of $8.99...
Payment successful!
Thank you for your order!
Your order will be prepared shortly.


=== Code Execution Successful ===
```

CONCLUSION

It shows that SS Cafe Online Food Management System is an upgraded ability towards efficiently and effectively serving its customers. Among the many key takeaways, include the following.

It provides users with an interactive menu which can easily navigate customers, enabling them to place orders, and process payments online. Customer satisfaction grows, as people find it very convenient and easy.

Smoothened Order Management: The orders taken through the system are not more prone to errors as those taken manually.

Dynamic Updating of Menus: The system enables menu updation in real time for SS Cafe to vary the offerings according to the availability, seasonal items, or based on the tastes of the customers that makes it remain relevant in the market.

Secure payment processing: Payment processing within the system makes it more secure, ensuring the transactions are in place so that customers will trust the purchase. It gives assurance and can drive the sale.

Operational Efficiency: This system provides customers ordering pattern information, popular items, and revenue trends, making data available for use in decision making, inventory, and marketing.

Scalability: The online system can be expanded to incorporate features such as loyalty programs, promotions, or integration with delivery services that SS Cafe may offer as the cafe grows.