

**BMS COLLEGE OF ENGINEERING BENGALURU**

Autonomous Institute, Affiliated to VTU



**SPC AAT Report on  
FOOD SERVICE RECEIPT GENERATOR**

*Submitted in partial fulfillment of the requirements for AAT*

Bachelor of Engineering  
in  
Computer Science and Engineering

*Submitted by:*

**YATHARTH GUPTA**

**SYED DANISH PASHA**

**Department of Computer Science and Engineering**

**BMS College of Engineering  
Bull Temple Road, Basavanagudi, Bangalore 560 019  
2025-2026**

**B.M.S COLLEGE OF ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



***DECLARATION***

We, YATHARTH GUPTA AND SYED DANISH PASHA students of 1<sup>st</sup> Semester, B.E, Department of AIML, BMS College of Engineering, Bangalore, hereby declare that, this AAT Project entitled "**FOOD SERVICE RECEIPT GENERATOR**" has been carried out in Department of CSE, BMS College of Engineering, Bangalore during the academic semester Oct 2024 – Jan 2025. We also declare that to the best of our knowledge and belief, the AAT Project report is not from part of any other report by any other students.

**Student Name**

- 1.YATHARTH GUPTA**
- 2.SYED DANSH PASHA**

**Student Signature**

**BMS COLLEGE OF ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND**  
**ENGINEERING**



***CERTIFICATE***

This is to certify that the AAT Project titled "**FOOD SERVICE RECIEPT GENERATOR**" has been carried out by **YATHARTH GUPTA(1BM25AI302)** and **SYED DANISH PASHA (1BM25AI043)** during the academic year 2025-2026.

Signature of the Faculty in Charge

## **Table of Contents**

Sl. No.	Title	Page no.
1	Introduction	1
2	Algorithm	4
3	Flowchart	7
4	Source code	9
5	Results (screenshots)	19
6	References	23

## 1. INTRODUCTION

### 1. Context and Background

The efficient and accurate management of financial transactions is critical for the sustainability and legal compliance of any food service establishment, particularly small cafes and eateries. Many small-scale operations still rely on manual or outdated methods for generating customer receipts, which leads to inefficiencies such as slow transaction times, high probability of human error in calculating totals, and difficulties in record-keeping. This project addresses this operational gap by developing a dedicated, robust, and resource-light solution. The choice of **C language** and a **console application** ensures the resulting program is highly efficient, has a minimal memory footprint, and provides a direct, reliable interface suitable for rapid deployment in constrained environments.

### 2. Project Aim (What is the Report About?)

The primary aim of this project is to **design, implement, and evaluate a functional Food Service Receipt Generator** using the C programming language. This system is specifically targeted at aiding small food service businesses to automate and standardize the process of generating receipts. The key objective is to provide an accurate, fast, and reliable digital system that can calculate itemized totals, apply dynamic tax rates, and present a clear, itemized receipt output.

### 3. Project Scope and Limitations (What is Included?)

The scope of this report and the corresponding application is defined by the following key functional requirements:

- **Itemized Input:** The system must allow for the input of multiple menu items, quantities, and their individual prices.
- **Calculation Engine:** Core functionality includes accurate calculation of subtotal, application of a defined Goods and Services Tax (GST) rate, and computation of the grand total.
- **Output Format:** The final receipt must be clearly displayed to the console, detailing the business name, itemized list, subtotal, tax amount, and final total.

- **Implementation:** The entire application is developed using structured programming principles in **ANSI C** and executed within a console environment.

The current version is limited to single-user operation (no multi-terminal networking) and is focused strictly on the core transaction logic (it does not include complex features like inventory management or database integration).

## 4. Report Structure (The Roadmap)

The remainder of this report details the development and evaluation of the Food Service Receipt Generator. Section 2 provides the full System Design, including flowcharts and data structures.

### 4.1. Context and Background

The efficient and accurate management of financial transactions is critical for the sustainability and legal compliance of any food service establishment, particularly small cafes and eateries. Many small-scale operations still rely on manual or outdated methods for generating customer receipts, which leads to inefficiencies such as slow transaction times, high probability of human error in calculating totals, and difficulties in record-keeping. This project addresses this operational gap by developing a dedicated, robust, and resource-light solution. The choice of **C language** and a **console application** ensures the resulting program is highly efficient, has a minimal memory footprint, and provides a direct, reliable interface suitable for rapid deployment in constrained environments.

### 4.2. Project Aim (What is the Report About?)

The primary aim of this project is to **design, implement, and evaluate a functional Food Service Receipt Generator** using the C programming language. This system is specifically targeted at aiding small food service businesses to automate and standardize the process of generating receipts. The key objective is to provide an accurate, fast, and reliable digital system that can calculate itemized totals, apply dynamic tax rates, and present a clear, itemized receipt output.

### 4.3. Project Scope and Limitations (What is Included?)

The scope of this report and the corresponding application is defined by the following key functional requirements:

- **Itemized Input:** The system must allow for the input of multiple menu items, quantities, and their individual prices.

- **Calculation Engine:** Core functionality includes accurate calculation of subtotal, application of a defined Goods and Services Tax (GST) rate, and computation of the grand total.
- **Output Format:** The final receipt must be clearly displayed to the console, detailing the business name, itemized list, subtotal, tax amount, and final total.
- **Implementation:** The entire application is developed using structured programming principles in ANSI C and executed within a console environment.

The current version is limited to single-user operation (no multi-terminal networking) and is focused strictly on the core transaction logic (it does not include complex features like inventory management or database integration).

#### **4.4. Report Structure (The Roadmap)**

The remainder of this report details the development and evaluation of the Food Service Receipt Generator. Section 2 provides the full System Design, including flowcharts and data structures. Section 3 discusses the Implementation and coding methodologies used in C. Section 4 outlines the Test Plan and presents the results of functional and stress testing. Finally, Section 5 concludes the report by evaluating the project's success against its initial aims and suggesting future enhancements.

## **2.ALGORITHM**

### **STEP 1: Initialization**

- Initialize variables to store the current date, time, customer details, order items (name, quantity, price, total), and set the initial **subtotal** to 0.0.
- Define constants: **GST\_RATE** (0.05) and **MAX\_ITEMS** (20).

### **STEP 2: Get Time and Bill Number**

- Call the **getDateTime** function to fetch the current system date and time and store them in **dateStr** and **timeStr**.
- Call the **getBillNumber** function, which reads the last bill number from **bill\_count.txt**, increments it by one, saves the new number back to the file, and returns the new unique bill number (**billNo**).

### **STEP 3: Prepare Receipt File**

- Create a filename string using the new **billNo** (e.g., **bill\_101.txt**).
- Open this file for writing ("w") and assign the file pointer to **receiptFile**.

### **STEP 4: Display Menu and Take Order**

- Call **printMenu** to display the fixed menu items and their prices to the user.
- Call **takeOrder** to begin the interactive ordering process:
  - Prompt the user to enter an item number (1-5) or 0 to finish.
  - Validate the item choice and quantity, repeating prompts until valid input is received.
  - If the item is new, add its details (name, quantity, price, total) to the item arrays and increment **itemCount**.
  - If the item is a repeat, update the existing quantity and total in the arrays.

- Continuously update the running **subtotal**.

### STEP 5: Get Customer and Order Details

- Call `getCustomerDetails` to prompt the user for their **Customer Name**.
- Prompt the user to choose the **Order Type** (Dine In or Online) and store the selection in `orderTypeStr`.

### STEP 6: Calculate Taxes and Total

- Calculate the Goods and Services Tax (`gst`) by multiplying `subtotal` by `GST_RATE` (0.05):  $gst = subtotal \times GST\_RATE$ .
- Calculate the final **total** amount:  $total = subtotal + gst$ .

### STEP 7: Get Payment Choice

- Call `getPaymentChoice` to display the payment options (Cash, Card, UPI).
- Prompt the user to select one (1, 2, or 3) and return the choice (`paymentChoice`).

### STEP 8: Print and Save Receipt Details

- Call `printReceipt`, passing all collected data (`billNo`, date, time, customer details, item arrays, totals, `subtotal`, `gst`, `total`, and the `receiptFile`).
- This function prints the full, formatted receipt to the **console** and simultaneously writes the same details to the `receiptFile`.

### STEP 9: Handle Payment Transaction

- Call `handlePayment` with the `paymentChoice`, `total`, and `receiptFile`.
- *If the payment choice is **Cash** (1):*
  - It enters a loop that prompts the user for cash amount until the total amount is covered.

- It calculates and displays the **Balance Returned**.
- It records the payment method and balance on the receipt file.

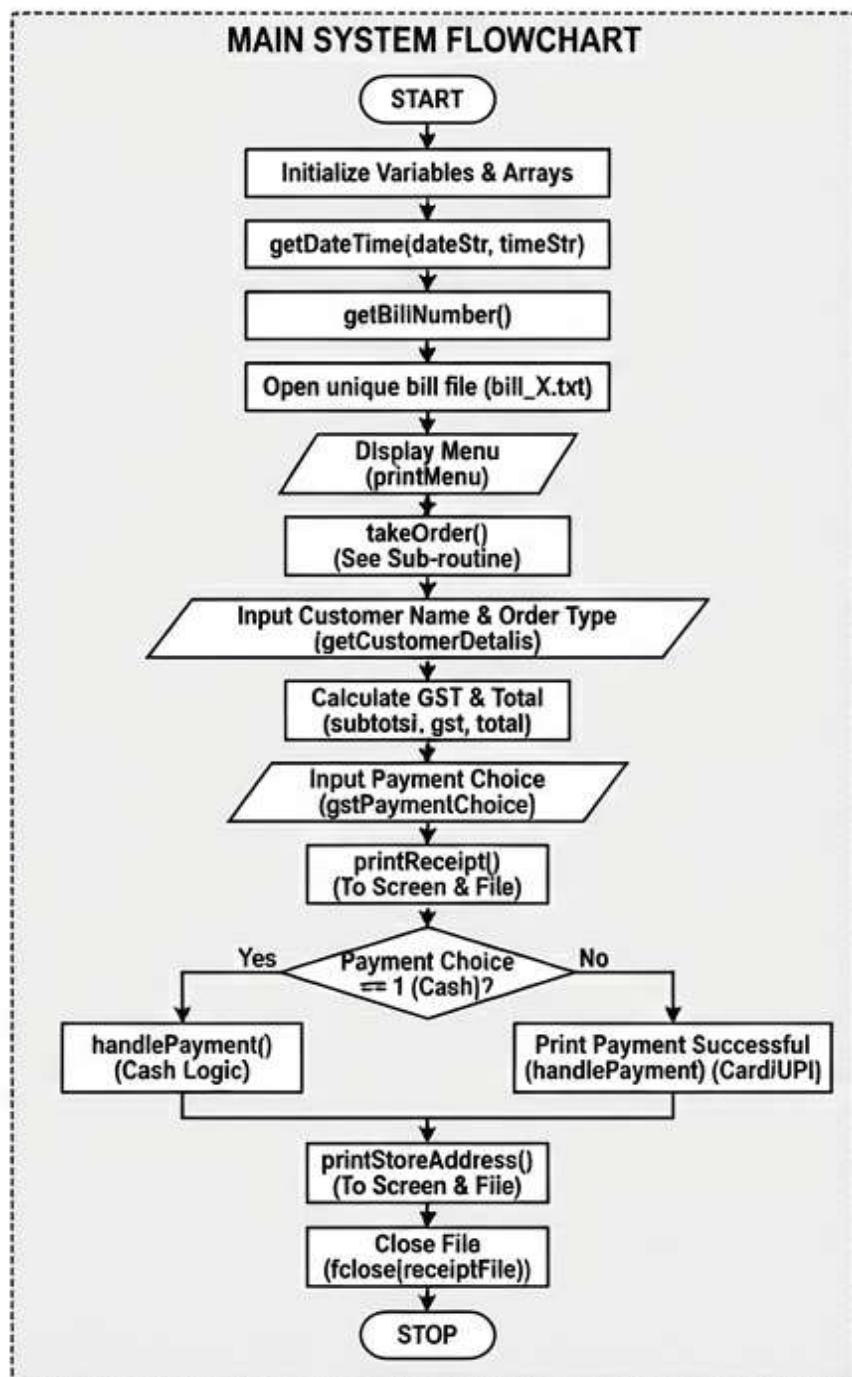
#### **STEP 10: Print Store Information**

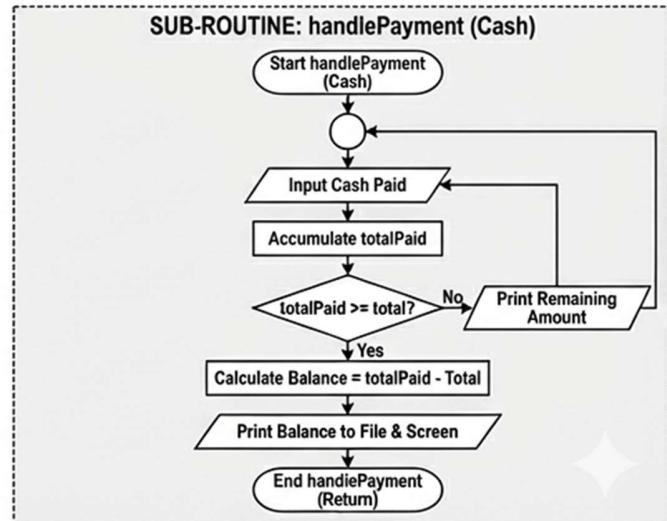
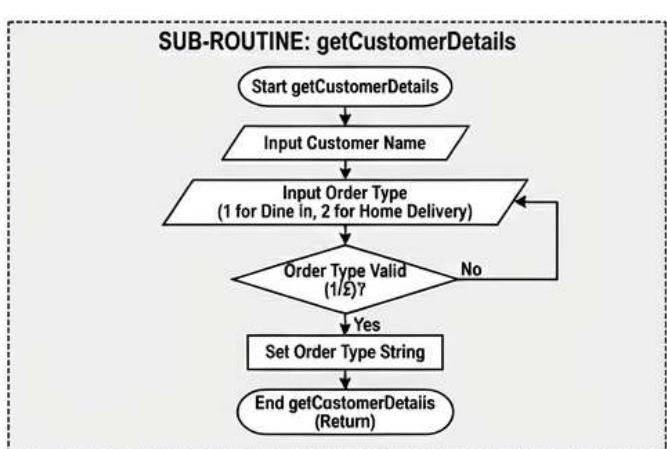
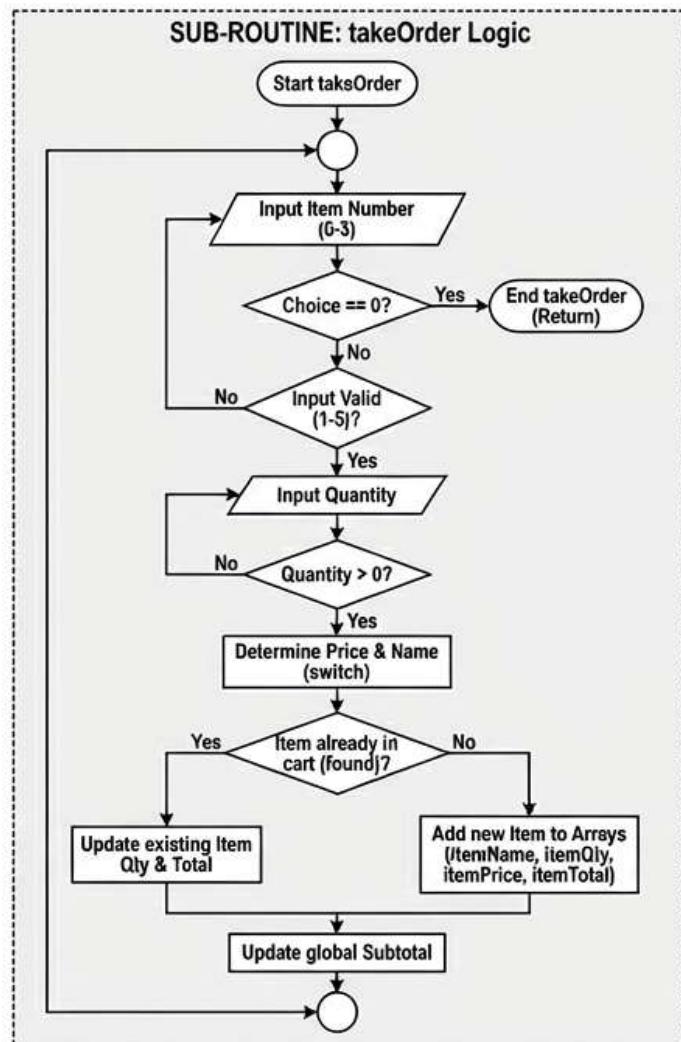
- Call `printStoreAddress` to display the store address on the **console** and append it to the `receiptFile`.

#### **STEP 11: Finalize**

- Close the receipt file using `fclose(receiptFile)`.
- Display a confirmation message on the console, indicating the filename where the receipt was saved.
- The program successfully terminates (`return 0`).

### 3.FLOWCHART





## 4. SOURCE CODE

```
#include <stdio.h>
#include <time.h>
#include <string.h>

#define GST_RATE 0.05
#define MAX_ITEMS 20

/* ----- FUNCTION DECLARATIONS ----- */
void getDateTime(char dateStr[], char timeStr[]);
int getBillNumber();
void printMenu();
void takeOrder(char itemName[] [30], int itemQty[],
float itemPrice[],
        float itemTotal[], int *itemCount,
float *subtotal);
void getCustomerDetails(char customerName[], char
orderTypeStr[]);
int getPaymentChoice();
void printReceipt(int billNo, char dateStr[], char timeStr[],
                char customerName[], char orderTypeStr[],
                char itemName[] [30], int itemQty[],
                char paymentTypeStr[],
                float itemPrice[], float itemTotal[],
                int itemCount, float subtotal, float gst,
float total,
                FILE *receiptFile);
void handlePayment(int paymentChoice, float total, FILE
*receiptFile);
void printStoreAddress(FILE *receiptFile);

/* ----- MAIN ----- */
int main() {
    char dateStr[20], timeStr[20];
    char customerName[50], orderTypeStr[20];
    char itemName[MAX_ITEMS] [30];
    int itemQty[MAX_ITEMS], itemCount = 0;
    float itemPrice[MAX_ITEMS], itemTotal[MAX_ITEMS];
    float subtotal = 0.0, gst, total;

    getDateTime(dateStr, timeStr);
```

```
int billNo = getBillNumber();

char fileName[30];
sprintf(fileName, "bill_%d.txt", billNo);
FILE *receiptFile = fopen(fileName, "w");

printMenu();
takeOrder(itemName, itemQty, itemPrice, itemTotal,
&itemCount, &subtotal);
getCustomerDetails(customerName, orderTypeStr);

gst = subtotal * GST_RATE;
total = subtotal + gst;

int paymentChoice = getPaymentChoice();

char paymentTypeStr[20];

if (paymentChoice == 1)
    strcpy(paymentTypeStr, "Cash");
else if (paymentChoice == 2)
    strcpy(paymentTypeStr, "Card");
else
    strcpy(paymentTypeStr, "UPI");

printReceipt(billNo, dateStr, timeStr, customerName,
orderTypeStr,
        itemName, itemQty,
        paymentTypeStr, itemPrice, itemTotal,
        itemCount, subtotal, gst, total, receiptFile);

handlePayment(paymentChoice, total, receiptFile);
printStoreAddress(receiptFile);

fclose(receiptFile);
printf("\nReceipt saved as %s\n", fileName);

return 0;
}

/* ----- FUNCTION DEFINITIONS ----- */
```

```
void getDateTime(char dateStr[], char timeStr[]) {
    time_t t = time(NULL);
    struct tm *tm_info = localtime(&t);
    strftime(dateStr, 20, "%d-%m-%Y", tm_info);
    strftime(timeStr, 20, "%I:%M %p", tm_info);
}

int getBillNumber() {
    FILE *file = fopen("bill_count.txt", "r");
    int billNo = 1;

    if (file != NULL) {
        fscanf(file, "%d", &billNo);
        billNo++;
        fclose(file);
    }

    file = fopen("bill_count.txt", "w");
    fprintf(file, "%d", billNo);
    fclose(file);

    return billNo;
}

void printMenu() {
    printf("=====\\n");
    printf("          WELCOME TO FOOD HUB      \\n");
    printf("=====\\n");
    printf("1. Burger      - Rs 120\\n");
    printf("2. Pizza       - Rs 250\\n");
    printf("3. Sandwich    - Rs 100\\n");
    printf("4. French Fries - Rs 80\\n");
    printf("5. Soft Drink   - Rs 50\\n");
    printf("-----\\n");
}

void takeOrder(char itemName[][30], int itemQty[], float
itemPrice[],
               float itemTotal[], int *itemCount, float
*subtotal) {

    int choice, quantity;
    float price;
```

```
int found, i;

while (1) {

    /* Validate item choice */
    while (1) {
        printf("Enter item number (1 - 5) (0 if done): ");

        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n'); // clear invalid
input
            printf("\nInvalid input! Please enter a number
between 0 and 5.\n\n");
            continue;
        }

        if (choice < 0 || choice > 5) {
            printf("\nInvalid choice! Please enter a number
between 0 and 5.\n\n");
            continue;
        }

        break; // valid input
    }

    if (choice == 0) {
        break; // finish ordering
    }

    /* Validate quantity */
    while (1) {
        printf("Enter quantity: ");

        if (scanf("%d", &quantity) != 1) {
            while (getchar() != '\n');
            printf("\nInvalid input! Please enter a valid
quantity.\n\n");
            continue;
        }

        if (quantity <= 0) {
            printf("\nQuantity must be greater than
0.\n\n");
        }
    }
}
```

```
        continue;
    }

    break; // valid quantity
}

/* Set item price and name */
switch (choice) {
    case 1: price = 120; strcpy(itemName[*itemCount],
"Burger"); break;
    case 2: price = 250; strcpy(itemName[*itemCount],
"Pizza"); break;
    case 3: price = 100; strcpy(itemName[*itemCount],
"Sandwich"); break;
    case 4: price = 80; strcpy(itemName[*itemCount],
"French Fries"); break;
    case 5: price = 50; strcpy(itemName[*itemCount],
"Soft Drink"); break;
}

/* Check if item already exists */
found = 0;
for (i = 0; i < *itemCount; i++) {
    if (strcmp(itemName[i], itemName[*itemCount]) == 0)
{
    itemQty[i] += quantity;
    itemTotal[i] += price * quantity;
    *subtotal += price * quantity;
    found = 1;
    break;
}
}

/* If new item, add it */
if (!found) {
    itemQty[*itemCount] = quantity;
    itemPrice[*itemCount] = price;
    itemTotal[*itemCount] = price * quantity;
    *subtotal += itemTotal[*itemCount];
    (*itemCount)++;
}

printf("Item added successfully!\n\n");
```

```
    }

}

void getCustomerDetails(char customerName[], char
orderTypeStr[]) {
    int orderType;
    char orderChar;

    printf("\nEnter Customer Name: ");
    scanf(" %[^\n]", customerName);

    printf("\nOrder Type:\n");
    printf("1. Dine In\n");
    printf("2. Online\n");

    do {
        printf("Choose order type (1/2): ");
        scanf(" %c", &orderChar);

        if (orderChar != '1' && orderChar != '2') {
            printf("\nInvalid choice! Please enter 1 or
2.\n\n");
        }
    } while (orderChar != '1' && orderChar != '2');

    /* Convert char to int */
    orderType = orderChar - '0';
    if (orderType == 1)
        strcpy(orderTypeStr, "Dine In");
    else
        strcpy(orderTypeStr, "Home Delivery");
}

int getPaymentChoice() {
    char choice;

    printf("\nPayment Options:\n");
    printf("1. Cash\n");
    printf("2. Card\n");
    printf("3. UPI\n");

    do {
```

```
printf("Choose payment method (1/2/3) : ");
scanf(" %c", &choice);

if (choice != '1' && choice != '2' && choice != '3') {
    printf("\nInvalid choice! Please enter 1, 2, or
3.\n\n");
}

} while (choice != '1' && choice != '2' && choice != '3');

return choice - '0'; // converts '1' → 1, '2' → 2, '3' →
3
}

void printReceipt(int billNo, char dateStr[], char timeStr[],
                  char customerName[], char orderTypeStr[],
                  char itemName[][30], int itemQty[],
                  char paymentTypeStr[],
                  float itemPrice[], float itemTotal[],
                  int itemCount, float subtotal, float gst,
float total,
FILE *receiptFile) {

printf("\n=====\\n");
printf("          FOOD HUB RECEIPT           \\n");
printf("          Bill No : %d             \\n",
billNo);
printf("=====\\n");
printf("Date : %s    Time : %s\\n", dateStr, timeStr);
printf("Customer : %s\\nOrder Type : %s\\n", customerName,
orderTypeStr);
printf("Payment : %s\\n", paymentTypeStr);
fprintf(receiptFile, "Payment : %s\\n", paymentTypeStr);
printf("-----\\n");

//for saving
fprintf(receiptFile,
"=====\\n");
fprintf(receiptFile, "          FOOD HUB
RECEIPT           \\n");
fprintf(receiptFile, "          Bill No :
%d             \\n", billNo);
```

```

        fprintf(receiptFile,
"=====\\n");
        fprintf(receiptFile, "Date : %s    Time : %s\\n", dateStr,
timeStr);
        fprintf(receiptFile, "Customer    : %s\\nOrder Type : %s\\n",
customerName, orderTypeStr);
        fprintf(receiptFile, "-----\\n");

        printf("%-15s %-6s %-10s %-10s\\n", "Item", "Qty", "Price",
"Total");
        printf(receiptFile, "%-15s %-6s %-10s %-10s\\n", "Item",
"Qty", "Price", "Total");

        for (int i = 0; i < itemCount; i++) {
            printf("%-15s %-6d Rs %-7.2f Rs %-7.2f\\n",
                   itemName[i], itemQty[i], itemPrice[i],
itemTotal[i]);
            fprintf(receiptFile, "%-15s %-6d Rs %-7.2f Rs %-
7.2f\\n",
                   itemName[i], itemQty[i], itemPrice[i],
itemTotal[i]);
        }

        printf("-----\\n");
        printf("Subtotal :                         Rs %.2f\\nGST (5%)
:");
                    Rs
%.2f\\nTotal   :                         Rs %.2f\\n",
                     subtotal, gst, total);
        fprintf(receiptFile, "-----\\n");
        fprintf(receiptFile, "Subtotal :                         Rs
%.2f\\n", subtotal);
        fprintf(receiptFile, "GST (5%) :                         Rs
%.2f\\n", gst);
        fprintf(receiptFile, "Total      :                         Rs
%.2f\\n", total);
    }

void handlePayment(int paymentChoice, float total, FILE
*receiptFile) {
    float cashPaid, balance;

```

```
if (paymentChoice == 1) {  
  
    float totalPaid = 0.0;  
  
    while (totalPaid < total) {  
        printf("Enter Cash Amount: Rs ");  
        scanf("%f", &cashPaid);  
  
        if (cashPaid <= 0) {  
            printf("Invalid amount! Please enter a valid  
amount.\n");  
            continue;  
        }  
  
        totalPaid += cashPaid;  
  
        if (totalPaid < total) {  
            printf("Insufficient amount!\n");  
            printf("Remaining amount to be paid : Rs  
%.2f\n\n", total - totalPaid);  
        }  
    }  
  
    balance = totalPaid - total;  
    printf("\nPayment Successful\n");  
    printf("Balance Returned: Rs %.2f\n", balance);  
  
    fprintf(receiptFile, "Payment Method: Cash\n");  
    fprintf(receiptFile, "Total Paid: Rs %.2f\n",  
totalPaid);  
    fprintf(receiptFile, "Balance Returned: Rs %.2f\n",  
balance);  
}  
else {  
    printf("\nPayment Successful\n");  
    fprintf(receiptFile, "Payment Successful via %s\n",  
           paymentChoice == 2 ? "Card" : "UPI");  
}  
  
}  
  
void printStoreAddress(FILE *receiptFile) {
```

```
printf("\n=====\\n") ;
printf("                STORE ADDRESS          \\n") ;
printf("                FOOD HUB              \\n") ;
printf("                123, MG Road         \\n") ;
printf("                Bengaluru - 560001    \\n") ;
printf("                Karnataka, India     \\n") ;
printf("=====\\n") ;

fprintf(receiptFile, "\nSTORE ADDRESS\\nFOOD HUB\\n123, MG
Road\\nBengaluru - 560001\\nKarnataka, India\\n");
}
```

## 5. RESULTS

```
=====
WELCOME TO FOOD HUB
=====

1. Burger      - Rs 120
2. Pizza       - Rs 250
3. Sandwich    - Rs 100
4. French Fries - Rs 80
5. Soft Drink   - Rs 50
-----
Enter item number (1 - 5) (0 if done): 1
Enter quantity: 2
Item added successfully!

Enter item number (1 - 5) (0 if done): 3
Enter quantity: 1
Item added successfully!

Enter item number (1 - 5) (0 if done): 5
Enter quantity: 1
Item added successfully!

Enter item number (1 - 5) (0 if done): 0

Enter Customer Name: XYZ

Order Type:
1. Dine In
2. Online
Choose order type (1/2): 1

Payment Options:
1. Cash
2. Card
3. UPI
Choose payment method (1/2/3): 3
```

```
=====
FOOD HUB RECEIPT
Bill No : 71
=====
Date : 06-01-2026  Time : 11:40 AM
Customer : XYZ
Order Type : Dine In
Payment : UPI
-----
Item          Qty   Price   Total
Burger        2     Rs 120.00  Rs 240.00
Sandwich      1     Rs 100.00  Rs 100.00
Soft Drink    1     Rs 50.00   Rs 50.00
-----
Subtotal :                               Rs 390.00
GST (5%) :                             Rs 19.50
Total   :                               Rs 409.50
-----
Payment Successful
-----
STORE ADDRESS
FOOD HUB
123, MG Road
Bengaluru - 560001
Karnataka, India
-----
Receipt saved as bill_71.txt
PS E:\C\AAT> |
```

```
=====
WELCOME TO FOOD HUB
=====

1. Burger      - Rs 120
2. Pizza       - Rs 250
3. Sandwich    - Rs 100
4. French Fries - Rs 80
5. Soft Drink   - Rs 50

-----
Enter item number (1 - 5) (0 if done): 3
Enter quantity: 2
Item added successfully!

Enter item number (1 - 5) (0 if done): 4
Enter quantity: 2
Item added successfully!

Enter item number (1 - 5) (0 if done): 5
Enter quantity: 2
Item added successfully!

Enter item number (1 - 5) (0 if done): 0

Enter Customer Name: XYZ

Order Type:
1. Dine In
2. Online
Choose order type (1/2): 1

Payment Options:
1. Cash
2. Card
3. UPI
Choose payment method (1/2/3): 2
```

```
=====
FOOD HUB RECEIPT
Bill No : 73
=====

Date : 06-01-2026 Time : 11:45 AM
Customer : XYZ
Order Type : Dine In
Payment : Card

-----
Item          Qty  Price   Total
Sandwich      2    Rs 100.00  Rs 200.00
French Fries   2    Rs 80.00   Rs 160.00
Soft Drink     2    Rs 50.00   Rs 100.00
-----
Subtotal :                               Rs 460.00
GST (5%) :                             Rs 23.00
Total   :                               Rs 483.00

Payment Successful

-----
STORE ADDRESS
FOOD HUB
123, MG Road
Bengaluru - 560001
Karnataka, India
=====

Receipt saved as bill_73.txt
PS E:\C\AAT> []
```

```
=====
WELCOME TO FOOD HUB
=====

1. Burger      - Rs 120
2. Pizza       - Rs 250
3. Sandwich    - Rs 100
4. French Fries - Rs 80
5. Soft Drink   - Rs 50

Enter item number (1 - 5) (0 if done): 1
Enter quantity: 1
Item added successfully!

Enter item number (1 - 5) (0 if done): 6
Invalid choice! Please enter a number between 0 and 5.

Enter item number (1 - 5) (0 if done): 0

Enter Customer Name: XYZ

Order Type:
1. Dine In
2. Online
Choose order type (1/2): 3

Invalid choice! Please enter 1 or 2.

Choose order type (1/2): 1

Payment Options:
1. Cash
2. Card
3. UPI
Choose payment method (1/2/3): 1
```

```
=====
FOOD HUB RECEIPT
Bill No : 75
=====

Date : 06-01-2026    Time : 11:52 AM
Customer : XYZ
Order Type : Dine In
Payment : Cash

-----
Item          Qty     Price    Total
Burger        1       Rs 120.00  Rs 120.00

-----
Subtotal :                               Rs 120.00
GST (5%) :                             Rs 6.00
Total :                                 Rs 126.00
Enter Cash Amount: Rs 120
Insufficient amount!
Remaining amount to be paid : Rs 6.00

Enter Cash Amount: Rs 6

Payment Successful
Balance Returned: Rs 0.00

=====
STORE ADDRESS
FOOD HUB
123, MG Road
Bengaluru - 560001
Karnataka, India

=====
Receipt saved as bill_75.txt
PS E:\C\AAT> |
```

## Food Service Receipt Generator

\*bill\_71.txt - Notepad

File Edit Format View Help

Payment : UPI

=====

FOOD HUB RECEIPT  
Bill No : 71

=====

Date : 06-01-2026 Time : 11:40 AM  
Customer : XYZ  
Order Type : Dine In

=====

Item	Qty	Price	Total
Burger	2	Rs 120.00	Rs 240.0
Sandwich	1	Rs 100.00	Rs 100.0
Soft Drink	1	Rs 50.00	Rs 50.00

=====

Subtotal : Rs 390.00  
GST (5%) : Rs 19.50  
Total : Rs 409.50

Payment Successful via UPI

STORE ADDRESS  
FOOD HUB  
123, MG Road  
Bengaluru - 560001  
Karnataka, India

\*bill\_73.txt - Notepad

File Edit Format View Help

Payment : Card

=====

FOOD HUB RECEIPT  
Bill No : 73

=====

Date : 06-01-2026 Time : 11:45 AM  
Customer : XYZ  
Order Type : Dine In

=====

Item	Qty	Price	Total
Sandwich	2	Rs 100.00	Rs 200.0
French Fries	2	Rs 80.00	Rs 160.0
Soft Drink	2	Rs 50.00	Rs 100.0

=====

Subtotal : Rs 460.00  
GST (5%) : Rs 23.00  
Total : Rs 483.00

Payment Successful via Card

STORE ADDRESS  
FOOD HUB  
123, MG Road  
Bengaluru - 560001  
Karnataka, India

\*bill\_75.txt - Notepad

File Edit Format View Help

Payment : Cash

=====

FOOD HUB RECEIPT  
Bill No : 75

=====

Date : 06-01-2026 Time : 11:52 AM  
Customer : XYZ  
Order Type : Dine In

=====

Item	Qty	Price	Total
Burger	1	Rs 120.00	Rs 120.0

=====

Subtotal : Rs 120.00  
GST (5%) : Rs 6.00  
Total : Rs 126.00

Payment Method: Cash  
Total Paid: Rs 126.00  
Balance Returned: Rs 0.00

STORE ADDRESS  
FOOD HUB  
123, MG Road  
Bengaluru - 560001  
Karnataka, India

## **6. REFERENCES**

- Gemini AI
- C Programming
- Dennis M. Ritchie C programming language textbook