

# Project Title

## **FAST FOOD MANAGEMENT SYSTEM (GUI + Console Hybrid)**

Assembly (MASM) program with enhanced popups via MessageBoxA and console interaction using Irvine32 library.

### **Group Members**

- Member 1: Areeba Zubair — 24k-0596
- Member 2: Maryam — 24k-3011
- Member 3: Aqsa Zubair — 24k-0599

**Submission Date:** 25 Nov, 2025

---

## 1. Executive Summary

### **Overview:**

This project implements a simple Fast Food Management System using x86 Assembly (MASM) that combines console-based interaction (Irvine32 routines) with Windows GUI popups (MessageBoxA) for welcome and billing. The system lets a user select a menu (Breakfast, Lunch, Dinner), add multiple items with quantities, computes a detailed bill, applies conditional discounts (5% or 10%), accepts payment, computes change, and displays a formatted bill in a MessageBox.

### **Key Findings:**

- The hybrid GUI/console approach provides a straightforward user experience while demonstrating low-level programming and Windows API usage.
  - Discounts are correctly applied based on subtotal thresholds ( $\geq 500$  for 5%,  $\geq 1000$  for 10%).
  - A dynamically built bill string (via `wsprintfA`) enables a readable popup summary.
  - The program demonstrates array indexing, loops, conditional branching, integer arithmetic, and basic input validation in assembly.
-

## 2. Introduction

### Background:

This project is relevant to Object-Oriented Programming coursework as a systems/programming project that emphasizes program structure, modular design (procedures), and user interaction logic. Although implemented in assembly rather than a typical OOP language, the project demonstrates software engineering principles: modularity (separate procedures), state management (global variables), and separation of concerns (input, processing, output).

### Project Objectives:

- Implement a menu-driven ordering system with three menus (Breakfast, Lunch, Dinner).
  - Allow multiple item entries with quantities.
  - Calculate item totals, subtotal, discounts, net total, accept payment, and compute change.
  - Present a final bill using a Windows MessageBox (dynamic formatted string).
  - Demonstrate proper use of MASM, Irvine32.inc helpers, and basic Win32 API functions.
- 

## 3. Project Description

### Scope:

#### Included:

- Menu display (console).
- Order-taking loop supporting multiple items.
- Price lookup from arrays for each menu category.
- Detailed bill printing to console.
- Discount rules and net total computation.
- Payment input and change computation.
- Final detailed bill popup (MessageBox using wsprintfA to build string).

#### Excluded (not implemented):

- Persistent storage (file save/load of orders).
- A graphical window with controls (the GUI is limited to popups).
- Multi-user concurrency or networking.
- Complex input sanitization for non-numeric entries.

### Technical Overview:

- Language / Assembler: MASM (x86).
- Helper library: Irvine32.inc (console I/O and helpers).
- Windows API: MessageBoxA, wsprintfA.
- Linker libraries: user32.lib, kernel32.lib.

- Development environment: Visual Studio (MASM) or any MASM-capable IDE. (Specify the exact IDE/version used when finalizing the report.)
- 

## 4. Methodology

### Approach:

- Break project into modules/procedures: DisplayMenu, TakingOrder, calcTotal, CalcDiscount, and main.
- Iteratively develop and test each procedure: first menu display, then order collection, then price calculations, then discount/payment, then popup formatting.
- Use sample test cases (many small orders, large orders) to validate arithmetic and branching.

### Roles and Responsibilities:

- Member 1 — Core assembly logic, menu display, order-taking and calcTotal procedure, in addition to assisting in report writing.
  - Member 2 — Developed price arrays, enhanced all procedures with GUI popup integration (MessageBoxA, wsprintfA), and performed testing.
  - Member 3 — Implemented the discount procedure and integrated GUI popups using MessageBoxA and wsprintfA, in addition to assisting in testing.
- 

## 5. Project Implementation

### Design and Structure:

- main shows welcome popup and menu popup, then calls DisplayMenu, calcTotal, and CalcDiscount.
- DisplayMenu prints menu options and jumps to a chosen menu's listing, then calls TakingOrder.
- TakingOrder loops to collect item codes and quantities, storing them into orderCode[] and quantity[] arrays.
- calcTotal iterates recorded orders, selects item names for printing (menu-specific), looks up prices in BreakfastCosts, LunchCosts, or DinnerCosts arrays, multiplies by quantity, and accumulates subTotal.
- CalcDiscount applies conditional discount logic, updates NetTotal, requests payment via ReadInt, computes change, and populates variables used by wsprintfA for the MessageBox.

### Functionalities Developed:

- Menu selection (Breakfast / Lunch / Dinner).
- Multiple-item ordering with continuation prompt.
- Price lookup and line-item total calculation.
- Subtotal and conditional discount (0%, 5%, 10%) application.
- Payment capturing and change calculation.
- Formatted bill popup using `wsprintfA` and `MessageBoxA`.

### Challenges Faced & Resolutions:

- String formatting for the popup: `wsprintfA` requires correct format and matching parameter types; solution: prepare `billPopupFmt` and pass `dword` values in correct order.
- Integer percentage calculations: Avoided floating point by using `imul` followed by `idiv` with multiplier (100) for percent. Careful ordering and preserving registers were needed to avoid corrupting important values.
- Array indexing in assembly: Corrected byte vs. `dword` indexing and used `dec` to convert 1-based input codes to 0-based arrays.
- Ensuring correct buffer size: Allocated `billPopupBuffer` with 256 bytes to safely contain the final formatted string.

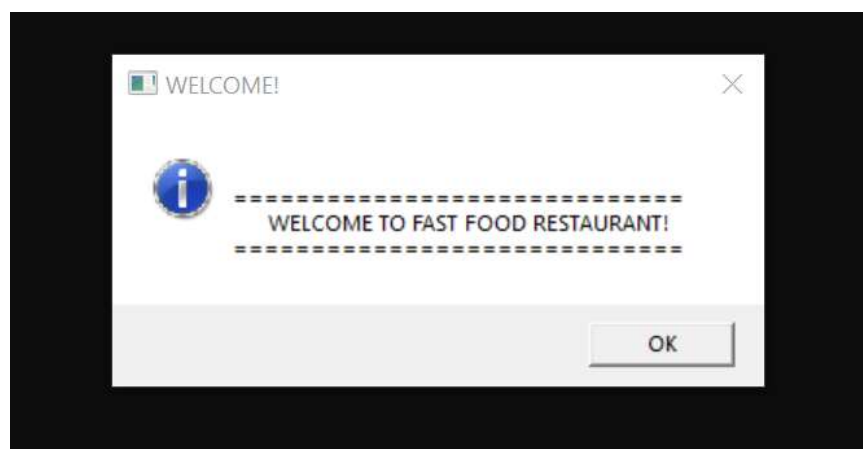
---

## 6. Results

### Project Outcomes:

- Functional ordering system that meets the stated objectives and correctly computes discounts and change.
- User receives both console feedback (detailed bill printed) and a GUI popup summarizing the bill.

### Screenshots and Illustrations:



```

-----
                        CHOOSE A MENU OPTION
-----

1 - BREAKFAST
2 - LUNCH
3 - DINNER

Enter Choice (1-3): 2

----- LUNCH MENU -----
ITEM                COST (Rs)                CODE
1) Biryani          Rs.250                1
2) Karahi            Rs.400                2
3) Roti              Rs.20                 3
4) Salad             Rs.100                4
5) Soup              Rs.80                 5

Enter item code (1-5): 1
Enter the desired quantity: 2

Do you wish to continue ordering? (1=Yes / 0=No): 1
Enter item code (1-5): 2
Enter the desired quantity: 1

Do you wish to continue ordering? (1=Yes / 0=No): 0

```

[illegible]



### Testing and Validation:

- Test case 1 — Small order: Breakfast Paratha x1 (Rs.50). Subtotal Rs.50; discount 0%; Net Rs.50; payment Rs.100; change Rs.50. Passed.
- Test case 2 — Medium order: Lunch Biryani x2 + Karahi x1 (subtotal >500 but <1000). Expected 5% discount. Verified console discount and popup values.
- Test case 3 — Large order: Multiple Dinner items producing subtotal  $\geq 1000$ . Expected 10% discount. Verified computation and change calculation.
- Edge cases: invalid menu choices and invalid item codes handled by loops that re-prompt. Non-numeric inputs are not fully handled by the current ReadDec/ReadInt routines; for robust input sanitization additional checks or string parsing would be required.

---

## 7. Conclusion

### Summary of Findings:

The FAST FOOD MANAGEMENT SYSTEM successfully demonstrates a hybrid console + popup GUI ordering and billing system implemented in MASM. It effectively illustrates assembly-level program structure, array-based price lookup, arithmetic operations for totals and percentage discounts, and Windows API usage for improved user interaction.

### Final Remarks & Recommendations:

- **Persistence:** Add file I/O to save receipts/orders (e.g., write bill records to a file).
  - **Input validation:** Implement stronger parsing to handle non-numeric or out-of-range entries gracefully.
  - **GUI upgrade:** Replace MessageBox popups with a simple Win32 window or a higher-level GUI library for richer interaction.
  - **Modularity & Maintainability:** Consider refactoring duplicated menu/name selection code into parameterized procedures to reduce repeated code.
  - **Internationalization & Currency:** Make currency and text configurable for other locales.
  - **Extensibility:** Add features like order cancellation, order editing before finalization, order history, and inventory tracking.
-