**Assignment**

FUNDAMENTALS IN PROGRAMMING CSE4002

**Name:** Jayasuriya Arachchige Don Prabath Udayanga

**Student ID:** CL/HDCSE/CMU/128/14

**Module:** Fundamentals in Programming - CSE4002

**Batch:** 128

**Course:**Higher Diploma in Computing and Software Engineering - CMU

**Date of Submission:** 07/02/2025

**Feedback Form**

**International College of Business & Technology**

**Module:** CSE4002 - Fundamentals in Programming

**Student:**

**Assessor:**

**Assignment:** Restaurant breakfast billing system

**Strong features of your work:**

**Areas for improvement:**

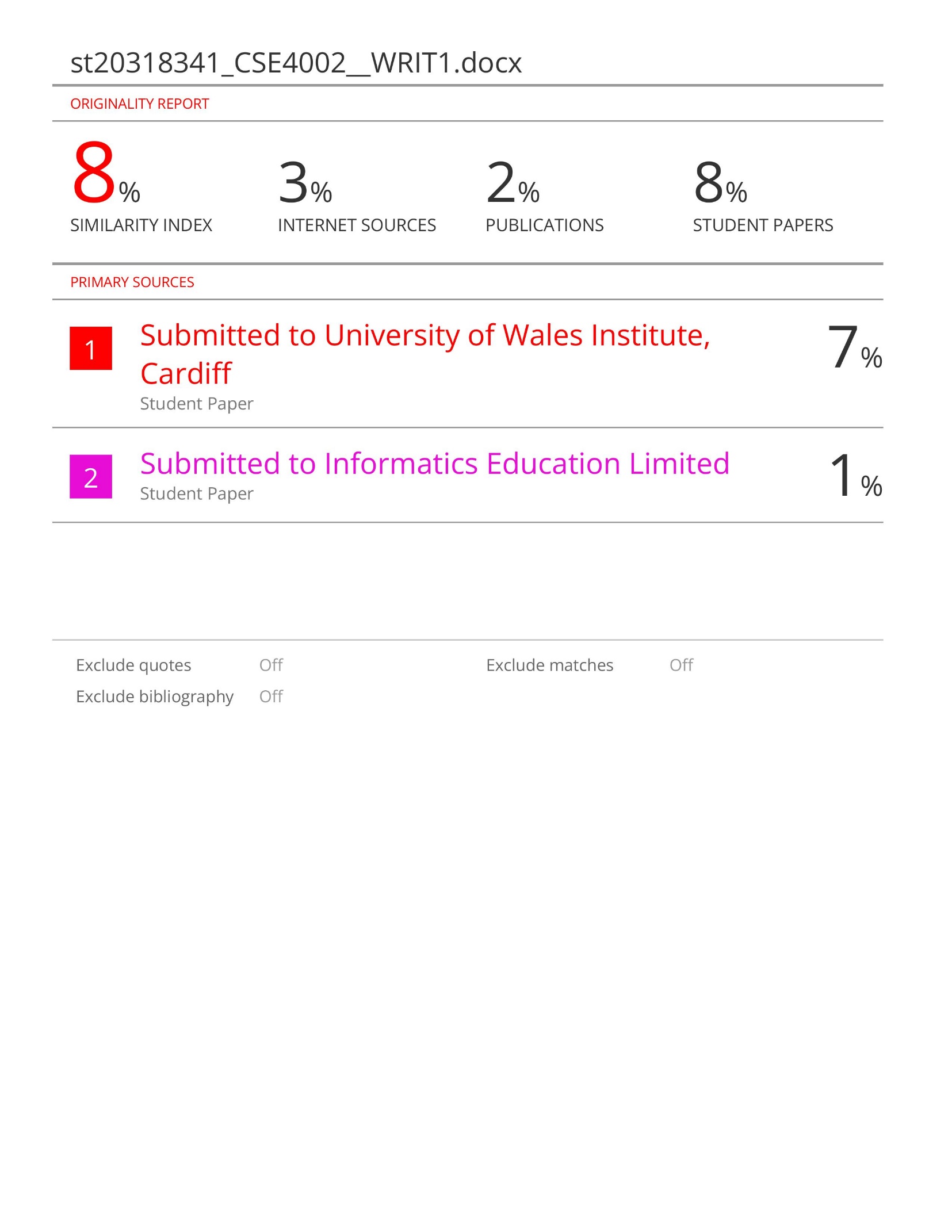
**Marks Awarded:**

**Assignment Cover Sheet**

|  |  |  |
| --- | --- | --- |
| **Qualification** | | **Module Number and Title** |
| HD in Computing and Software Engineering | | CSE4002 - **Fundamentals in Programming** |
| **Student Name & No.** | | **Assessor** |
| Prabath Udayanga (CL/HDCSE/CMU/128/14) | |  |
| **Hand out date** | | **Submission Date** |
|  | | 07/02/2025 |
| **Assessment type**  WRIT1-Coursework | **Duration/Length of**  **Assessment Type**  3000 Words | **Weighting of Assessment**  100% |

|  |
| --- |
| **Learner declaration** |
| I Prabath Udayanga (CL/HDCSE/CMU/128/14) certify that the work submitted for this assignment is my own and research sources are fully acknowledged. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Marks Awarded** | | | |
| First assessor | |  | |
| IV marks | |  | |
| Agreed grade | |  | |
| Signature of the assessor |  | Date |  |

****

**Acknowledgment**

First and foremost, we would like to thank our Fundamentals in Programming lecturer, Mr. Tharaka for his helpful guidance and support during the completion of this module. It might be impossible to have done this work if it were not for his confidence and guidance.

We would also like to thank our friends who have provided us with constant support needed in the completion of this project.

Table of Contents

Introduction.………………………………………………………………………… 5

Task 1 …………………………………………………………………......…………5

Task 2 ………………………………….………………………………………....…13

Full code………………………………………………………………………..14

Description on each part of code……………………………………………….18

Task 3 ….……………………………………………………………………………23

Test Plan……………………………………………………………………….24

Test Cases……………………………………………………………………...26

User Acceptance Testing………………………………………………………38

Conclusion ………………………………………….……………………………….39

References ……...…...………………………………………………………………40

**Restaurant breakfast billing system**

**Introduction**

The assignment about Breakfast Billing System for the local restaurant Meal Hut which particularly serves breakfast foods. The automated billing process will replace manual cash book practices by introducing an updated electronic system solution. Customers can easily choose menu items through the system interface and see their total bill get calculated with tax for improved speed and precision in service delivery.

**Task 1: Requirement Analysis and Design**

**Software Requirements Specification (SRS) for Meal Hut Breakfast Billing System**

**Project Scope**

**Use Case Diagram**

**Actors:**

* **Customer**: Selects menu items, views the bill.
* **Cashier**: Enters order or orders, generates the bill.
* **System**: Stores menu data, processes the orders, calculates the totals.

**Actor Descriptions**

* **Customer**: person who uses the ordering system to pick menu options while receiving their payment bill.
* **Cashier**: The employee responsibility for system order data entry and final bill production.
* **System**: The automated system that processes the order, which calculates prices with added tax before performing a print task.

**Use Case Scenarios and Alternative Scenarios**

**Use Case 1: Show Menu**

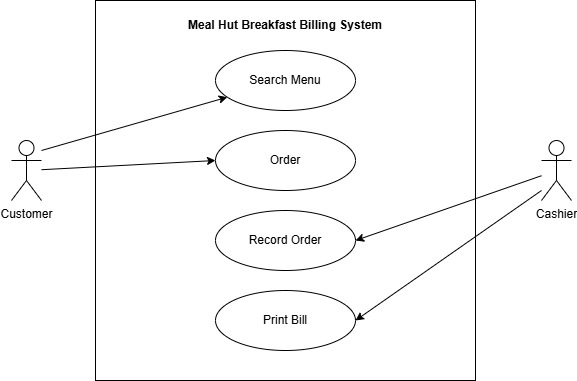
* **Scenario:**
  1. Customer requests to view the menu.
  2. System displays menu items with prices.
  3. Customer selects items to order.

**Use Case 2: Place Order**

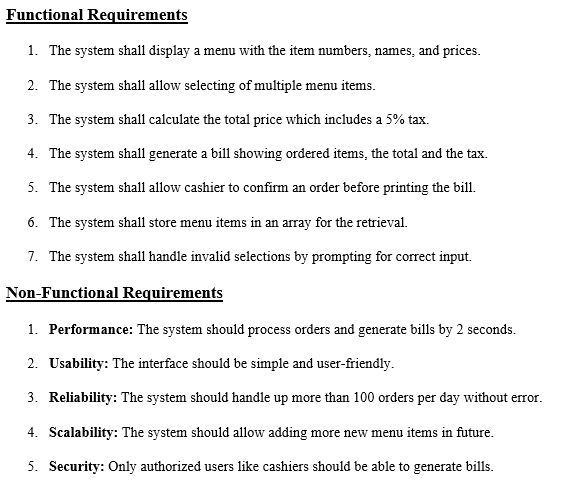
* **Scenario:**
  1. Cashier selects menu items based on customer’s order.
  2. System stores selected items.
  3. System calculates subtotal and tax.
  4. System generates the final bill.

**Use Case 3: Print Bill**

* **Scenario:**
  1. System retrieves ordered items and calculates the total.
  2. System applies a 5% tax.
  3. System displays and prints the final bill.

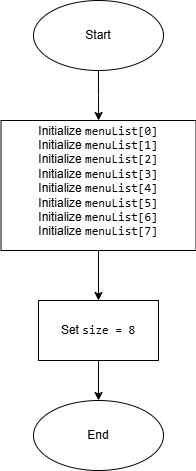
**Use Case Diagram**

**Business Rules Applied for Each Use Case**

1. **Show Menu:**
   * Only available menu items are displayed.
   * Prices must be updated dynamically in case of changes.
2. **Place Order:**
   * The cashier must confirm the order before finalizing it.
   * The system prevents duplicate orders unless explicitly requested.
   * Tax is automatically applied based on predefined percentages.
3. **Print Bill:**
   * The total bill must include a 5% tax.
   * The bill must display itemized pricing.
   * ****If an order is canceled, no bill is printed.

**Flowcharts**

1. **getData()**

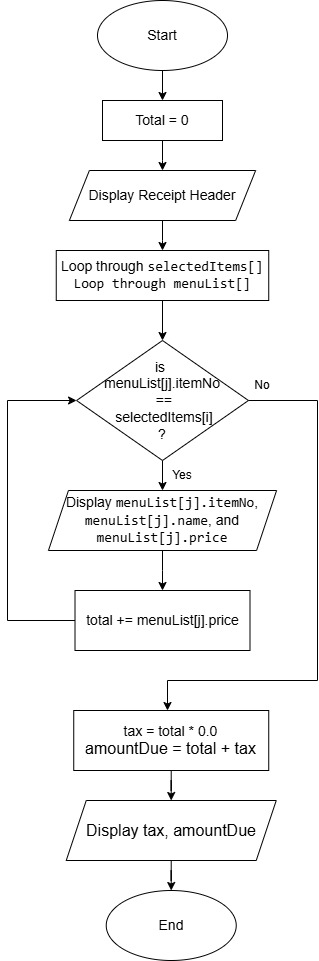


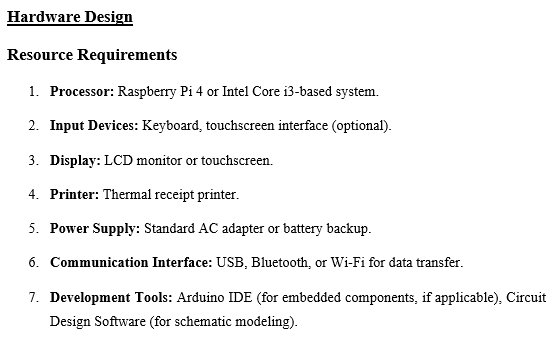
1. **showMenu()**

A diagram of a flowchart

Description automatically generated



1. **PrintCheck()**

**Approval**

This document is done for the development of the Meal Hut Breakfast Billing System. The document shows steps for implementing an efficient system.

**Prepared by:** Prabath Udayanga  
**Signature:** prabath  
**Date:** 2025/02/06

**Conclusion**

By automating billing processes the Meal Hut Breakfast Billing System increases operational efficiency while maintaining high accuracy rates and improves speed.

**Task 2: System Development**

**Introduction**

The quick-growing establishment Meal Hut prepares exclusively breakfast food items. At present Meal Hut maintains its billing process using manual cash book operations but this approach consumes valuable time while generating multiple errors. The breakfast billing system of Meal Hut receives a technological upgrade to enhance operational efficiency and improve the service quality for customers.

**Full Coding**

#include <iostream>

#include <iomanip>

using namespace std;

// Structure to store menu item details

struct MenuItem {

int itemNo;

string name;

double price;

};

// Function prototypes

void getData(MenuItem menuList[], int& size);

void showMenu(const MenuItem menuList[], int size);

void printCheck(const MenuItem menuList[], int size, int selectedItems[], int itemCount);

int main() {

const int MAX\_ITEMS = 8;

MenuItem menuList[MAX\_ITEMS];

int selectedItems[MAX\_ITEMS];

int size = 0, itemCount = 0;

getData(menuList, size);

int choice;

do {

cout << "\n\*\*\*\*\*\*\*\* Welcome to Meal Hut \*\*\*\*\*\*\*\*" << endl;

cout << " Breakfast Billing System" << endl;

cout << "1. Show Menu" << endl;

cout << "2. Order Items" << endl;

cout << "3. Print Bill" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

showMenu(menuList, size);

break;

case 2: {

int itemNo;

int tempCount = 0;

cout << "Enter item numbers to order (Please Enter -1 to stop):" << endl;

while (true) {

cin >> itemNo;

if (itemNo == -1) break;

bool valid = false;

for (int i = 0; i < size; i++) {

if (menuList[i].itemNo == itemNo) {

valid = true;

break;

}

}

if (valid) {

selectedItems[tempCount++] = itemNo;

}

else {

cout << "Invalid item number. Try again." << endl;

}

}

if (tempCount > 0) {

itemCount = tempCount;

}

else {

cout << "No items selected!" << endl;

}

break;

}

case 3:

if (itemCount == 0) {

cout << "No items selected!" << endl;

}

else {

printCheck(menuList, size, selectedItems, itemCount);

}

break;

case 4:

cout <<"Thank you for choosing us! Have a great day!"<< endl;

break;

default:

cout << "Invalid choice, please try again!" << endl;

}

} while (choice != 4);

return 0;

}

// Function to load menu data

void getData(MenuItem menuList[], int& size) {

menuList[0] = { 111, " Plain Egg", 1.45 };

menuList[1] = { 112, " Bacon and Egg", 2.45 };

menuList[2] = { 113, " Muffin\t", 0.99 };

menuList[3] = { 114, " French Toast", 1.99 };

menuList[4] = { 115, " Fruit Basket", 2.49 };

menuList[5] = { 116, " Cereal\t", 0.69 };

menuList[6] = { 117, " Coffee\t", 0.50 };

menuList[7] = { 118, " Tea\t", 0.75 };

size = 8;

}

// Function to display menu items

void showMenu(const MenuItem menuList[], int size) {

cout << "\n\*\*\*\*\*\*\*\* Welcome to Meal Hut \*\*\*\*\*\*\*\*" << endl;

cout << " Breakfast Billing System" << endl;

cout << "Item No Menu Item Price" << endl;

cout << "-----------------------------------" << endl;

for (int i = 0; i < size; i++) {

cout << menuList[i].itemNo << "\t" << menuList[i].name << "\t$" << menuList[i].price << endl;

}

}

// Function to calculate and print the bill

void printCheck(const MenuItem menuList[], int size, int selectedItems[], int itemCount) {

double total = 0.0;

cout << "\n\*\*\*\*\*\*\*\* Meal Hut Receipt \*\*\*\*\*\*\*\*" << endl;

cout << "Item No Menu Item Price" << endl;

cout << "-----------------------------------" << endl;

for (int i = 0; i < itemCount; i++) {

for (int j = 0; j < size; j++) {

if (menuList[j].itemNo == selectedItems[i]) {

cout << menuList[j].itemNo << "\t" << menuList[j].name << "\t$" << menuList[j].price << endl;

total += menuList[j].price;

}

}

}

double tax = total \* 0.05;

double amountDue = total + tax;

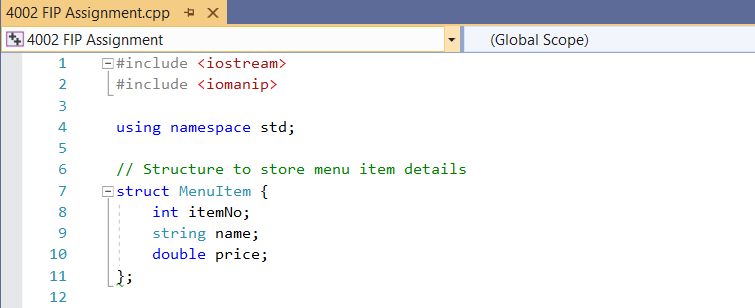
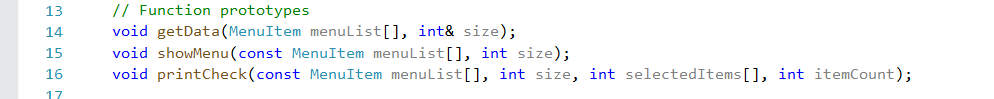
cout << "-----------------------------------" << endl;

cout << "Tax: " << "\t\t\t$" << tax << endl;

cout << "Amount Due: " << "\t\t$" << amountDue << endl;

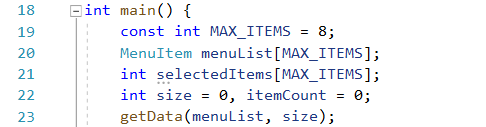
}

**Description on each part of code**

1. **Structure Definition**
2. **Function Prototypes**

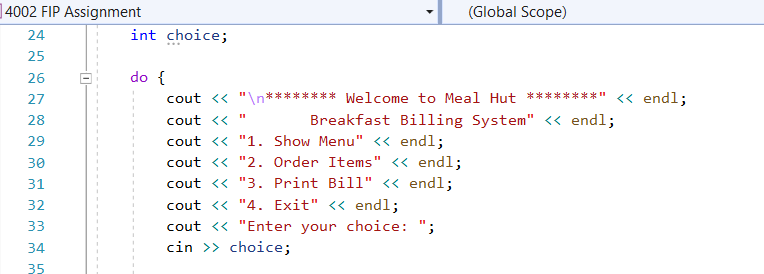
* **getData()** loads the menu data.
* **showMenu()** displays the menu.
* **printCheck()** calculates and print bill.

1. **main() Function**

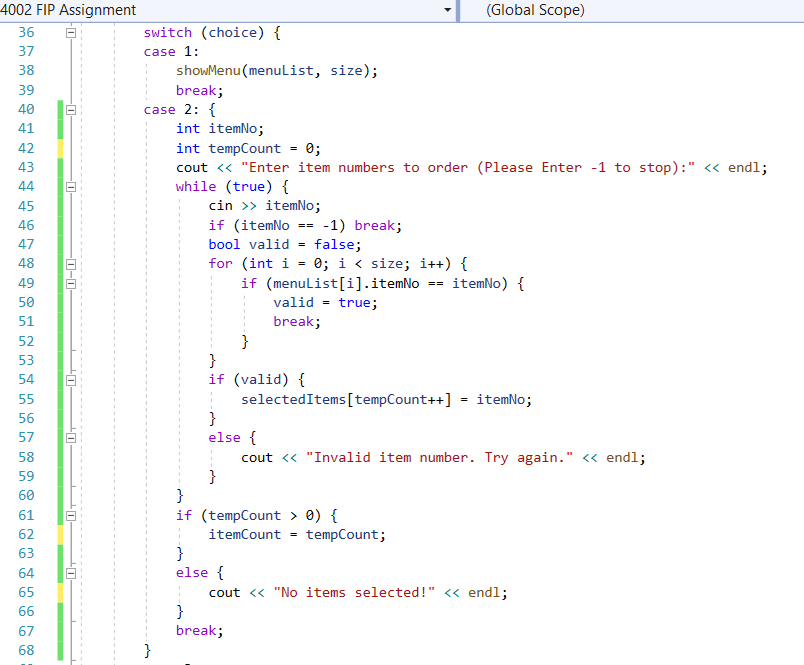


* Defines an array **menuList[]** to store the menu items.
* Defines an array **selectedItems[]** to store the selected items.
* **getData()** to initialize the menu.

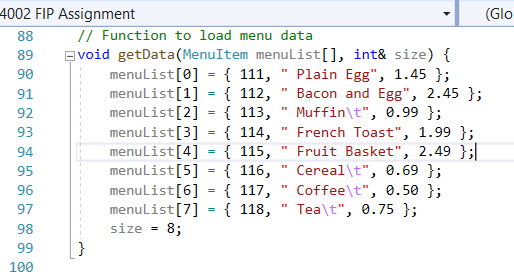
1. **Main Menu Loop**

****

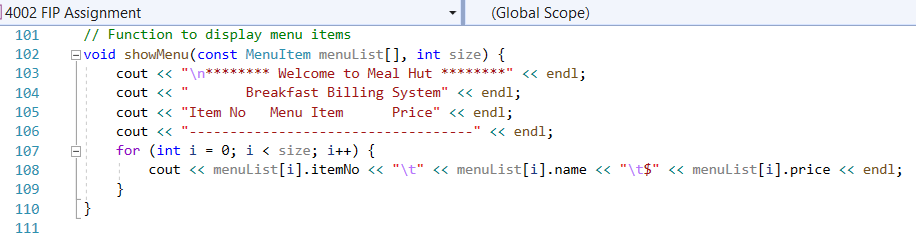
1. A screen shot of a computer program

   Description automatically generated**Handling User Choices**

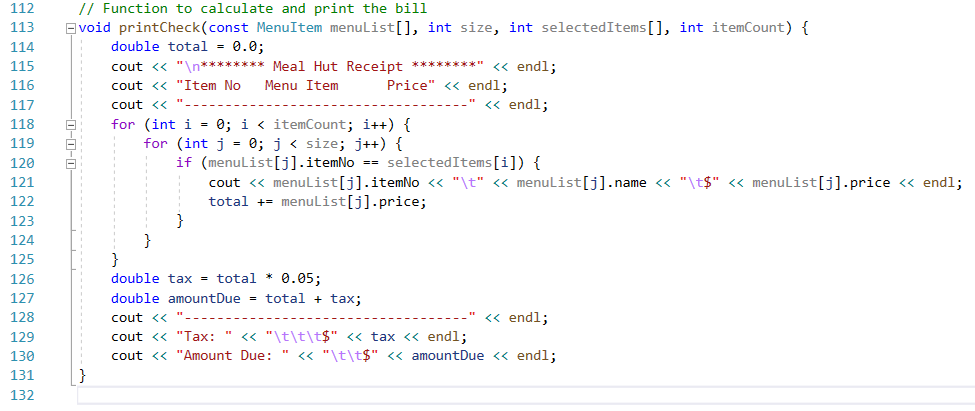
* Calls **showMenu()**, **printCheck()**, or handles ordering on user choice.
* Loops until the user chooses **Exit (4)** option.

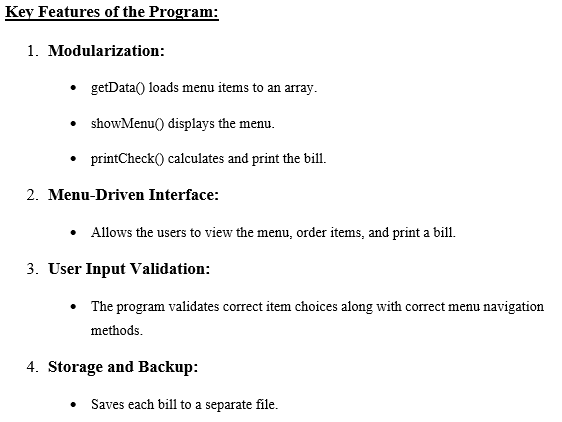
1. **getData() – Load Menu Data**

* **Store the breakfast menu** inside an array.
* Sets size = 8.

1. **showMenu() – Display Menu Items**

* **Prints the menu items**
* Uses **tabs (\t) for spacing**

1. **printCheck() – Calculate and Print Bill**

* **Loops through selected items** and prints them.
* **Calculates total, adds 5% tax**, and shows the final amount.

**Task 3: Testing and Documentation**

**Test Plan**

**Objectives**

* Verify the accuracy of the menu display.
* Make sure customers can select multiple items correctly.
* Validate bill calculations including tax.
* Check error handling for invalid inputs.
* Confirm stability and user-friendliness.

**Scope**

* **Unit Testing**: Testing each individual functions.
* **Integration Testing**: Make sure different modules work together.
* **System Testing**: Testing the entire system to determine its operational efficiency along with dependability features.
* **User Acceptance Testing:** Gathering feedback from users.

**Testing Environment**

* Operating System: Windows 11 or 10/Linux
* Hardware: Standard PC/Laptop

**Test Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Test Case Id** | **Test Case Name** | **Test Steps** | **Test Data** | **Expected Results** | **Test Result** |
| 1 | TC 1 | Main Menu | - | - | Main Menu display correctly | Pass |
| 2 | TC 2 | Exit Program | Select exit option | 4 | Program Exits with message “Thank you for choosing us! Have a great day!” | Pass |
| 3 | TC 3 | Main Menu Show Menu | Select Show Menu option | 1 | Menu should display correctly with all items | Pass |
| 4 | TC 4 | Incorrect Main Menu Selection | Enter invalid Choice | 5 | Error message “Invalid choice, please try again!” | Pass |
| 5 | TC 5 | Main Menu  Order Items | Select Order Items option | 2 | Request Order Item Numbers | Pass |
| 6 | TC 6 | Order Valid Items | Enter valid item numbers | 112, 113, 117, -1 | Items added to order | Pass |
| 7 | TC 7 | Order Invalid Item | Enter an invalid item number | 117, 199 | Show error message | Pass |
| 8 | TC 8 | Order All Items | Enter all menu item numbers | 111, 112, 113, 114, 115, 116, 117, 118, -1 | All items added successfully | Pass |
| 9 | TC 9 | Order Single Item | Enter one Item Number | 112, -1 | Item added successfully | Pass |
| 10 | TC 10 | Empty Order Bill | Proceed without selecting items | -1 | Show message "No items selected!" | Pass |
| 11 | TC 11 | Same Order Item | Enter same Item Number | 114, 111, 114, -1 | Correctly add multiple quantities | Pass |
| 12 | TC 12 | Calculate Bill | Add items, Select Print Bill option | 112, 114, 117, -1, 3 | Display Bill with correct total with tax | Pass |
| 13 | TC 13 | Print Bill option without Ordering | Select Print Bill option without adding items | 3 | Show message “No items selected!” | Pass |
| 14 | TC 14 | Calculate Large Bill | Add large number of items | 115, 116, 111, 118, 112, 114, 113, 117, 112, 111, -1 | Correct total with tax | Pass |
| 15 | TC 15 | Negative Input | Enter a negative number | -111 | Show error message | Pass |
| 16 | TC 16 | Ensure Menu Persistence | Open menu multiple times | - | Menu displays correctly | Pass |
| 17 | TC 17 | Valid Exit after printing bill | Select Exit option after getting the bill | 4 | System closes correctly | Pass |
| 18 | TC 18 | Reorder after printing the bill | Select Order Items option again | 2 | Allows to make multiple orders | Pass |
| 19 | TC 19 | Handle Zero Input | Enter 0 | 0 | Show error message | Pass |
| 20 | TC 20 | Make Order without viewing Menu | Select Order Items option | 2 | System works without error | Pass |
| 21 | TC 21 | Stop ordering by entering -1 | Enter -1 | -1 | Orders successfully added | Pass |
| 22 | TC 22 | Handle System Timeout | Leave system idle | - | System remains stable | Pass |
| 23 | TC 23 | Ensure No Crashes | Run multiple tests | - | System does not crash | Pass |

**Test Cases**

**Test Case 1: Main Menu**

|  |  |
| --- | --- |
| Test Case ID | 1 |
| Test Objectives | To check if Main Menu display correctly |
| Test Data | - |
| Expected Results | Main Menu should display correctly |
| Actual results | Main Menu should display correctly |
| Conclusion | Main Menu is working |
| Screenshot |  |

**Test Case 2: Exit Program**

|  |  |
| --- | --- |
| Test Case ID | 2 |
| Test Objectives | To check if program Exits with message “Thank you for choosing us! Have a great day!” |
| Test Data | 4 |
| Expected Results | Program should Exit without errors |
| Actual results | Program should Exit without errors |
| Conclusion | Program Exit successfully |
| Screenshot |  |

**Test Case 3: Show Menu**

|  |  |
| --- | --- |
| Test Case ID | 3 |
| Test Objectives | To check if Menu display correctly with all items |
| Test Data | 1 |
| Expected Results | Menu display correctly with all items |
| Actual results | Menu display correctly with all items |
| Conclusion | Menu display correctly |
| Screenshot |  |

**Test Case 4: Incorrect Main Menu Selection**

|  |  |
| --- | --- |
| Test Case ID | 4 |
| Test Objectives | To check if an error message display for invalid selection in Menu |
| Test Data | 5 |
| Expected Results | Error message “Invalid choice, please try again!” |
| Actual results | Error message “Invalid choice, please try again!” |
| Conclusion | Successfully displays an error for invalid selection |
| Screenshot |  |

**Test Case 5: Order Items**

|  |  |
| --- | --- |
| Test Case ID | 5 |
| Test Objectives | To check if a message display requesting order item numbers |
| Test Data | 2 |
| Expected Results | Message display requesting Order numbers |
| Actual results | Message display requesting Order numbers |
| Conclusion | The message successfully displays as expected |
| Screenshot |  |

**Test Case 6:** **Order Valid Items**

|  |  |
| --- | --- |
| Test Case ID | 6 |
| Test Objectives | To check if items are added to order |
| Test Data | 112, 113, 117 |
| Expected Results | Items added without any errors shown |
| Actual results | Items added without any errors shown |
| Conclusion | Items successfully added |
| Screenshot |  |

**Test Case 7:** **Order Invalid Item**

|  |  |
| --- | --- |
| Test Case ID | 7 |
| Test Objectives | To check if an error message is displayed for invalid order numbers |
| Test Data | 117,199 |
| Expected Results | Show error message “Invalid item number. Try again.” |
| Actual results | Show error message “Invalid item number. Try again.” |
| Conclusion | Error for invalid order numbers displayed successfully |
| Screenshot |  |

**Test Case 8:** **Order All Items**

|  |  |
| --- | --- |
| Test Case ID | 8 |
| Test Objectives | To check every item added successfully |
| Test Data | 111, 112, 113, 114, 115, 116, 117, 118 |
| Expected Results | All items added without any errors shown |
| Actual results | All items added without any errors shown |
| Conclusion | All items successfully added |
| Screenshot |  |

**Test Case 9: Order Single Item**

|  |  |
| --- | --- |
| Test Case ID | 9 |
| Test Objectives | To check any single item can be added successfully |
| Test Data | 112 |
| Expected Results | Item added without any errors shown |
| Actual results | Item added without any errors shown |
| Conclusion | Item is successfully added |
| Screenshot |  |

**Test Case 10: Empty Order Bill**

|  |  |
| --- | --- |
| Test Case ID | 10 |
| Test Objectives | To check if an error message is displayed when no order number entered |
| Test Data | - |
| Expected Results | Error message "No items selected!" |
| Actual results | Error message "No items selected!" |
| Conclusion | Error for empty order numbers displayed successfully |
| Screenshot |  |

**Test Case 11: Same Order Item**

|  |  |
| --- | --- |
| Test Case ID | 11 |
| Test Objectives | To check if multiple orders of same order number can be entered correctly |
| Test Data | 114, 111, 114 |
| Expected Results | Same Item numbers added without any errors shown |
| Actual results | Same Item numbers added without any errors shown |
| Conclusion | Items successfully added |
| Screenshot |  |

**Test Case 12: Calculate Bill**

|  |  |
| --- | --- |
| Test Case ID | 12 |
| Test Objectives | To check if display bill with correct total and tax |
| Test Data | 112, 114, 117, -1, 3 |
| Expected Results | Correct Total and Tax is shown |
| Actual results | Correct Total and Tax is shown |
| Conclusion | No errors in Total and Tax calculations |
| Screenshot |  |

**Test Case 13: Print Bill option without Ordering**

|  |  |
| --- | --- |
| Test Case ID | 13 |
| Test Objectives | To check if an error message is displayed for selecting print bill option without ordering |
| Test Data | 3 |
| Expected Results | Error message "No items selected!" |
| Actual results | Error message "No items selected!" |
| Conclusion | Error for selecting print bill option without ordering displayed successfully |
| Screenshot |  |

**Test Case 14: Calculate Large Bill**

|  |  |
| --- | --- |
| Test Case ID | 14 |
| Test Objectives | To check if display bill with correct total and tax for larger bill |
| Test Data | 115, 116, 111, 118, 112, 114, 113, 117, 112, 111, -1 |
| Expected Results | Correct bill with Total and Tax is shown |
| Actual results | Correct bill with Total and Tax is shown |
| Conclusion | No errors in Total and Tax calculations for a larger bill |
| Screenshot |  |

**Test Case 15:** **Negative Input**

|  |  |
| --- | --- |
| Test Case ID | 15 |
| Test Objectives | To check if an error message is displayed for selecting negative numbers |
| Test Data | -111 |
| Expected Results | Display error message |
| Actual results | Display error message |
| Conclusion | System doesn’t accept any negative inputs |
| Screenshot |  |

**Test Case 16: Ensure Main Menu Persistence**

|  |  |
| --- | --- |
| Test Case ID | 16 |
| Test Objectives | To check if main menu displays correctly by opening multiple times |
| Test Data | - |
| Expected Results | Main Menu displays correctly |
| Actual results | Main Menu displays correctly |
| Conclusion | Main menu can be open at any time without any error |
| Screenshot |  |

**Test Case 17:** **Valid Exit after printing bill**

|  |  |
| --- | --- |
| Test Case ID | 17 |
| Test Objectives | To check is system closes correctly after printing bill |
| Test Data | 4 |
| Expected Results | System closes correctly |
| Actual results | System closes correctly |
| Conclusion | No errors to exit from system after printing the bill |
| Screenshot |  |

**Test Case 18: Reorder after printing the bill**

|  |  |
| --- | --- |
| Test Case ID | 18 |
| Test Objectives | To check if user able to make multiple order attempts |
| Test Data | 2 |
| Expected Results | Allows to make multiple orders |
| Actual results | Allows to make multiple orders |
| Conclusion | Successfully able to reorder again after printing the bill |
| Screenshot |  |

**Test Case 19: Handle Zero Input**

|  |  |
| --- | --- |
| Test Case ID | 19 |
| Test Objectives | To check if an error message is displayed for selecting a 0 number |
| Test Data | 0 |
| Expected Results | Display error message |
| Actual results | Display error message |
| Conclusion | System doesn’t accept 0 inputs |
| Screenshot |  |

**Test Case 20: Make Order without viewing Menu**

|  |  |
| --- | --- |
| Test Case ID | 20 |
| Test Objectives | To check if its possible to make order without viewing the Menu first |
| Test Data | 2 |
| Expected Results | System works without error |
| Actual results | System works without error |
| Conclusion | Can make Orders without viewing menu |
| Screenshot |  |

**Test Case 21: Stop ordering by entering -1**

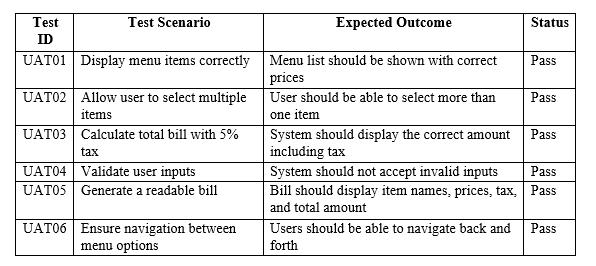
|  |  |
| --- | --- |
| Test Case ID | 21 |
| Test Objectives | To check if ordering stop and added once -1 entered |
| Test Data | -1 |
| Expected Results | All orders added once stopped |
| Actual results | All orders added once stopped |
| Conclusion | Ordering stops once -1 entered and successfully added |
| Screenshot |  |

**Test Case 22: Handle System Timeout**

|  |  |
| --- | --- |
| Test Case ID | 22 |
| Test Objectives | To check if system works well when it kept idle |
| Test Data | - |
| Expected Results | System remains stable |
| Actual results | System remains stable |
| Conclusion | Systems works well even if it is kept idle for anytime |
| Screenshot |  |

**Test Case 23: Ensure No Crashes**

|  |  |
| --- | --- |
| Test Case ID | 23 |
| Test Objectives | Run the full system multiple times to see if any errors visible |
| Test Data | - |
| Expected Results | No any crashes or errors found |
| Actual results | No any crashes or errors found |
| Conclusion | System works well without any error |
| Screenshot |  |

**User Acceptance Testing**

**User Feedback**

This is a survey that I did for the feedback. I did a questionnaire my friends to try my application and report back on their experiences.

Sample Questions and responses:

1. Is the system easy to navigate?

Responses: 10

Yes: 90%

No: 10%

1. Were you able to select multiple items without confusion?

Responses: 10

Yes: 100%

No: 0%

1. Did the bill calculation appear accurate and clear?

Responses: 10

Yes: 80%

No: 20%

1. Were there any difficulties in using the system?

Responses: 10

Yes: 80%

No: 20%

1. Is the system working without any error?

Responses: 10

Yes: 90%

No: 10%

1. How you would like rate the entire system?

Responses: 10

Excellent: 70%

Good: 30%

Worst: 0%

**Conclusion**

Through the Breakfast Billing System Meal Hut has achieved automated billing to provide rapid and precise service to consumers. The system makes operations more efficient through its menu-based approach combined with modular functions together with organized data storage mechanisms that improve both speed and usability experiences. Reliability of the system is confirmed through testing while accuracy is achieved by implementing proper input validation and tax calculation methods. The system has achieved its purposes and creates a platform for future developments that include order monitoring and digital payment capabilities.

**References**

1. *W3schools.com* (no date) *W3Schools Online Web Tutorials*. Available at: <https://www.w3schools.com/cpp/> (Accessed: 06 February 2025).
2. *Restaurant Billing System Project for final year student* (2022) *LovelyCoding.org*. Available at: <https://www.lovelycoding.org/resturant-billing-system/> (Accessed: 06 February 2025).
3. Soumyajit (2024) *How to manage complete billing structure for restaurants?*, *Taqtics*. Available at: <https://taqtics.co/manage-complete-billing-structure-for-restaurants/> (Accessed: 06 February 2025).
4. Gerhard Krüger                                and                Charles Lane (no date) *How to write an SRS document (software requirements specification document)*, *Perforce Software*. Available at: <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document> (Accessed: 06 February 2025).
5. Snappify (2023) *Software requirements specification sample: 101 expert guide*, *snappify*. Available at: <https://snappify.com/blog/software-requirements-specification-sample> (Accessed: 06 February 2025).
6. *Main function* (no date) *cppreference.com*. Available at: <https://en.cppreference.com/w/cpp/language/main_function> (Accessed: 06 February 2025).
7. *How to write test cases (with format & example)* (2024) *BrowserStack*. Available at: <https://www.browserstack.com/guide/how-to-write-test-cases> (Accessed: 06 February 2025).
8. *How to write test cases: A step-by-step qa guide* (no date) *Coursera*. Available at: <https://www.coursera.org/articles/how-to-write-test-cases> (Accessed: 06 February 2025).
9. *Qa best practices to improve software testing* (2025) *TestRail*. Available at: <https://www.testrail.com/blog/qa-best-practices/> (Accessed: 06 February 2025).
10. Team, K. (2025) *User acceptance testing: A complete guide*, *Katalon AI-augmented Test Automation Platform*. Available at: <https://katalon.com/resources-center/blog/user-acceptance-testing> (Accessed: 06 February 2025).