“Development of a Cost-Effective Desktop Cashiering System Using C++: Addressing Manual Inefficiencies in Sales and Payment Management for Small Printing Shops”

Flora, Imee
Guarin, Jan Mikaela C.
Moradillo, Eunice E.
Mendoza, Allyson E.
Realubit, Seth Andrew

Technological Institute of the Philippines
Quezon City
November 2025

Table of Contents

Introduction

Ikeu Shi Printing Shop is a family-owned microenterprise established in 2022 that provides essential printing services such as document printing, photocopying, and scanning to students, professionals, and local community residents. The business is operated solely by the owners, with no additional employed staff. Since its inception, they have relied on a fully manual approach to managing sales transactions and payment processing. All transactions are recorded in notebooks or logbooks, and payment computations are carried out using calculators or mobile devices. While cash remains the primary mode of payment, an increasing number of customers prefer to pay via e-wallet platforms such as GCash and Maya. For these transactions, they simply confirm that the payment has been sent. Receipts, when issued, are handwritten, and daily or weekly sales totals are calculated manually at the end of the business day. Although this approach is manageable for a small volume of customers, it becomes inefficient, time-consuming, and highly prone to human error as transaction volume increases, thereby affecting the accuracy and organization of financial records.

The general problem faced by Ikeu Shi Printing Shop is that its manual cashiering and sales recording processes are inefficient and prone to inaccuracies, making it difficult to maintain reliable transaction records and ensure effective payment management.

More specifically, the business encounters the following main issues:

1. **Manual Transaction Recording** – All sales are recorded by hand, increasing the likelihood of inaccurate entries, incomplete documentation, and potential loss or misplacement of records.

2. **Inefficient Digital Payment Verification** – Payments made through e-wallet platforms such as GCash and Maya require manual confirmation that the payment has been sent, which slows down the process and increases the possibility of errors.

3. **Lack of Automated Sales Reporting** – Sales summaries are generated manually, resulting in delays and potential inaccuracies in daily or periodic financial reporting, limiting the business's ability to monitor its financial performance effectively.

To address these challenges, this study proposes the development of a cost-effective desktop cashiering system using C++. The system will digitally record all sales transactions, reducing human error and improving accuracy. It will support e-wallet payments by confirming transactions without requiring manual entry of reference numbers, enabling faster and more reliable payment processing. In addition, the system will automatically generate sales reports, allowing the owners to monitor daily, weekly, and monthly performance efficiently. By implementing this solution, Ikeu Shi Printing Shop can improve operational efficiency, ensure accurate transaction tracking, and provide a more professional and convenient service for customers, particularly those preferring cashless payments.

The Project

System Entry and Navigation

The cashiering system begins with a straightforward interface that presents users with three primary options: sign-up, log-in, and exit. This menu acts as the central navigation point, allowing users to either register a new account, authenticate an existing one, or terminate the session. This design ensures that users are guided through a logical entry point before accessing the system's core functions (Cambria Software, 2024).

User Registration

When a user selects the sign-up option, the system initiates a registration process that prompts for a username and password. It performs validation checks to ensure the username is not empty and that the password matches its confirmation (Soulslicer, 2023). If the username already exists in the system's records, the user is notified with a message indicating that the username is taken (rnshalinda, 2024). Otherwise, the system hashes the password and securely stores the credentials. This approach aligns with modern security practices in cashiering systems, ensuring data protection and preventing unauthorized access (TylerMSFT, 2025)

User Authentication

Upon selecting the log-in option, the system verifies the entered username against stored records. If the username is not found, the system returns a "no such user" message (Studocu, 2024). If the username exists, the system prompts for a password and compares the hashed input with the stored hash. A successful match grants access to the main interface, while a mismatch redirects the user to the main menu (TylerMSFT, 2025). This authentication process ensures that only authorized personnel can access sensitive sales data and administrative functions.

Cashiering Operations

Once authenticated, users are directed to the cashiering module, which displays a list of available services such as;

Photocopy - Short	₱2
Photocopy - A4/Long	₱4
Print B&W - Short	₱3
Print B&W - A4/Long	₱5
Print Colored - Short	₱10
Print Colored - A4/Long	₱15

Each service is associated with a fixed price. The cashier selects a service, inputs the quantity, and the system calculates the subtotal (SourceCodester, 2023). It also updates internal records by incrementing the quantity sold and total revenue for each service (Asim et. al., 2022)

. This real-time transaction tracking supports efficient inventory and financial management, which is essential for small businesses (Pekkala, 2024).

Payment Processing

After finalizing the order, the system prompts the user to select a payment method, either cash or e-wallet (Pekkala, 2024). If the e-wallet option is chosen, the system requests a reference number, simulating QR code verification (Genie & Sharma, 2024). The user then enters the payment amount, and the system calculates any change due. If the payment is insufficient, the transaction is canceled. Otherwise, a receipt is generated showing the total amount, payment method, and reference number if applicable (PeerDH, 2024). This integration of digital payment options reflects the growing trend of cashless transactions in small enterprises.

Sales Reporting

For business owners, the system includes a secure sales reporting feature accessible via owner login. This module displays a detailed summary of each service, including the quantity sold and total revenue (Asim et. al., 2022). It also shows the overall sales total, helping owners monitor performance and make informed decisions (Cambria Software, 2024). The report is formatted using C++ I/O tools for clarity and precision, making it easy to interpret even for users with limited technical expertise (Pekkala, 2024).

System Termination

Users can exit the system by selecting the exit option from the main menu or cashiering interface. The system then displays a closing message and safely terminates the session (TylerMSFT, 2025). This ensures a clean shutdown and preserves system integrity.

PROJECT MODEL

The Ikeu Shi Printing Shop Desktop Cashiering System was developed using an iterative life cycle model, selected primarily to address the project's limited timeframe and the evolving nature of business requirements. This model emphasizes continual refinement and adaptation, with each cycle enabling the team to revisit and improve upon earlier stages based on feedback and emerging needs. Iterative development also encouraged collaboration and practical problem-solving, ensuring that the system would fully meet the operational challenges encountered by the shop.

Each phase focused on a core aspect of the cashiering and sales management workflow, allowing the team to systematically address inefficiencies, implement solutions, and validate improvements in real-world scenarios. By applying the principles of the iterative model, problems could be identified early and solutions could be quickly refined, ultimately resulting in a reliable and easy-to-use system that greatly improved transaction speed and reporting accuracy for the shop.

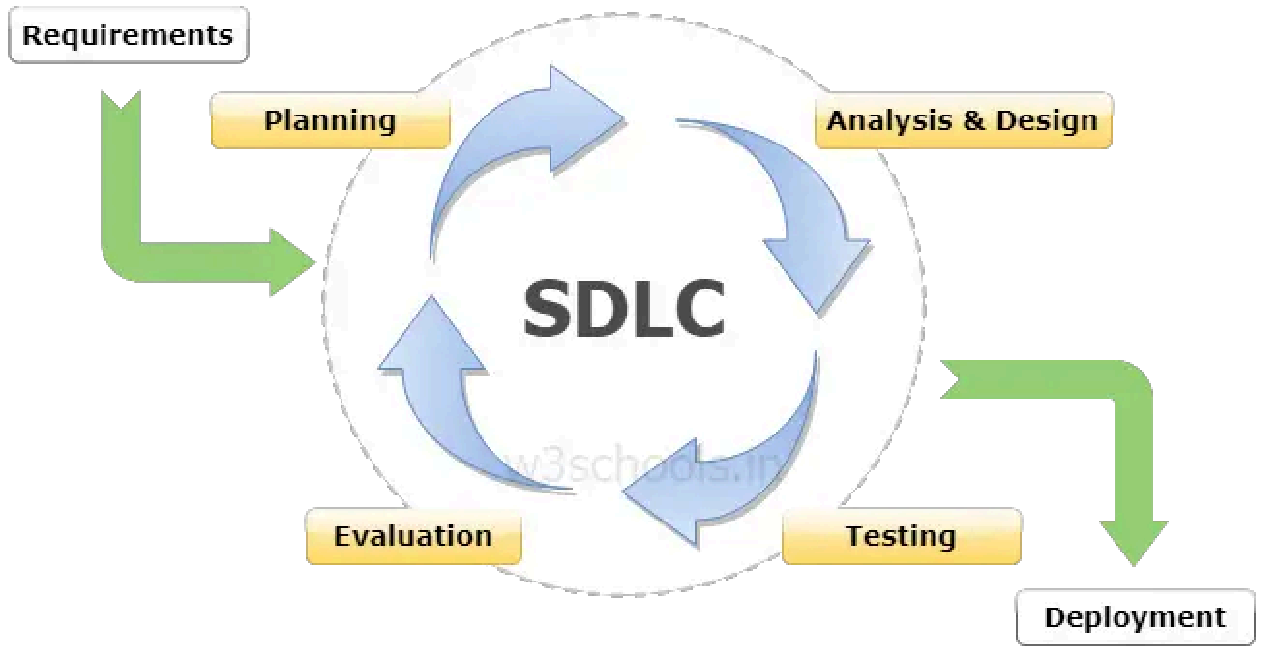

Fig: SDLC Iterative Model

FIGURE 1 : SDLC Iterative Model.

Requirements Phase

- The team began by investigating the shop's operational workflow, focusing on inefficiencies such as manual sales logging, slow payment confirmations, and lack of automated reporting. Input was collected directly from daily operations and in discussions with owners to ensure all real-world issues were addressed.
- Requirements such as fast cashiering, secure owner login, e-wallet reference validation, and menu-driven navigation were set as the foundation for the planned system. These needs were carefully documented to guide the upcoming design and implementation work.

Iteration 1:

Planning and Analysis:

- The team began by mapping out the manual workflow, finding inefficiencies and the risk of human error in everyday sales tasks. Key metrics for transaction speed, accuracy, and reporting clarity were set as initial benchmarks. Observational notes and user input helped shape the first round of objectives.

Designing:

- Initial system flowcharts and pseudocode were crafted to visualize and standardize cash handling and sales recording. The design emphasized a simple, friendly menu-driven interface operated via the terminal. A foundational data structure for service and payment information was outlined in a basic data dictionary.

Implementation and Testing:

- The first iteration of code established modules for core cashiering functions like service selection, payment entry, and sales recording. Simple unit tests were conducted after each module, ensuring features operated as planned. Usability was reviewed during dry runs to confirm that staff could quickly adapt.

Evaluation:

- The earliest user tests highlighted issues with input reliability and made clear the benefits of persistent storage and owner-controlled settings. Observed errors and improvement ideas were systematically logged for the next round of development. Lessons learned here informed updates in security, navigation, and error-proofing for future iterations.

Iteration 2:

Planning and Analysis:

- Based on first-round feedback, attention shifted to securing owner access and making payment validation more rigorous, particularly for digital transactions. Additional analysis looked at file-based storage solutions to strengthen data retention and daily reporting. User journey mapping was revisited to reduce steps and avoid confusion during busy periods.

Designing:

- Flowcharts were revised and a new pseudocode was developed for secure owner login, improved e-wallet handling, and advanced reporting. The team expanded the data structures to support richer sales analytics and easier updates. Visuals and workflows were refined for better intuitiveness and fewer user errors.

Implementation and Testing:

- New modules were layered onto the core, including persistent data management, enhanced validation routines, and modular payment-processing logic. Comprehensive test cases simulated various user scenarios, validating both positive and negative flows. Continuous debugging helped maintain system stability during incremental updates.

Evaluation:

- The enhanced application underwent user walkthroughs and targeted scenario tests to check for seamless operation across features. System feedback loops allowed prompt fixes and minor tweaks before finalization. The researchers consolidated all feedback in a summary report to prepare for deployment.

Review/Deployment:

After completion, the full system was reviewed against all functional and business goals, and deployment documentation was created. Staff and owner training included hands-on use to ensure comfort and readiness for live transactions. The digital system officially replaced the manual process, delivering faster, more accurate, and user-friendly shop operations

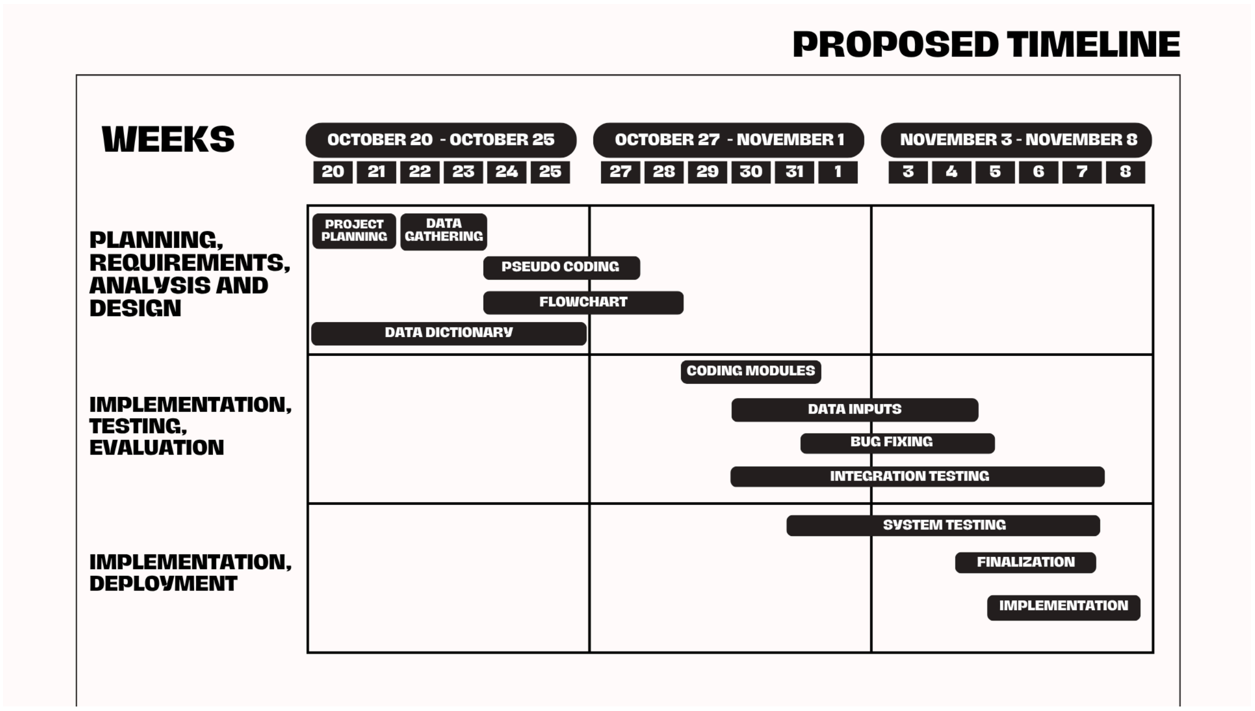

FIGURE 2: Proposed Project Timeline

The proposed timeline provides a structured schedule for a project, breaking down tasks according to weekly milestones. During the first week (October 20–25), the focus is on planning, requirements gathering, analysis, and design, including project planning, data gathering, pseudocoding, creating flowcharts, and compiling a data dictionary. The second week (October 27–November 1) transitions into coding modules, initiating data input, and beginning bug fixing.

The final week (November 3–8) intensifies implementation tasks, with activities such as integration testing, system testing, finalization, and deployment. This timeline ensures each phase which is the design, evaluation, and deployment that progresses systematically, highlighting overlaps where tasks like system testing and implementation run concurrently for efficient project delivery.

Hardware and Software Requirements

Hardware:

- **Desktop or Laptop Computer:** A system with at least an Intel Core i3 processor (or equivalent AMD), 4 GB RAM, and 100 GB free storage space to ensure smooth operation of the C++-based cashiering application.
- **Keyboard and Mouse:** Standard USB or wireless keyboard for data entry and a USB or wireless mouse for navigation in the console-based interface.
- **Monitor:** A display with a minimum resolution of 1024x768 (e.g., 19-inch LCD or LED screen) to clearly show transaction details, receipts, and reports.

Software:

- **Compatible Operating System:** Windows 10/11 or Linux distributions such as Ubuntu 18.04 or later, ensuring compatibility with C++ compilers.
- **C++ Compiler and Runtime Environment:** GCC (GNU Compiler Collection) version 7.0 or later (available via MinGW on Windows or native on Linux), or Microsoft Visual C++ 2019 or later, for compiling and running the application.
- **Standard C++ Libraries**: Core libraries like <iostream> for input/output operations, <fstream> for file handling (e.g., saving transaction logs), and <string> for data manipulation, all included with the C++ standard library (C++11 or higher)

Statement of the Problem

This research aims to address these operational issues to develop a cost-effective and user-friendly desktop cashiering system using C++ language. Most cashiering systems are dependent on manual entry wherein it can make the transactions prone to human errors and the transaction tracking is time consuming. Generating daily and weekly sales reports requires difficult manual computation. Additionally, the lack of backups and data protection in manual systems increases the possibilities of manipulation and data loss.

1. Due to manual entry and the lack of automated validation processes, transaction tracking is inconsistent and inaccurate.
2. Customers that prefer cashless transactions would find this difficult to use as this lacks the use of internet connection when using e-wallet transactions .
3. Adoption to the current cashiering system presents usability and technical issues as this requires internet connectivity.

Objectives

This study aims to improve transaction and payment efficiency as well as the operation accuracy in printing businesses. A cost-effective desktop cashiering system C++ technology must be designed, developed, and evaluated.

1. Is to design a cost-effective desktop cashiering system that will automates all sales reports and make the transaction efficient.
2. Is to develop a C++ based system that has an integrated payment service that supports both cash and e-wallet transactions with its reference number.

Flowchart of the System

The flowchart for the system named **"Development of a Cost-Effective Desktop Cashiering System Using C++: Addressing Manual Inefficiencies in Sales and Payment Management for Small Printing Shops"** shows a clear and systematic flow, starting from the main menu and moving towards the core function of the system which is inside the main interface. The flowchart starts from right to left with the following features:

- Login
- Cashiering
- Sales Tracking

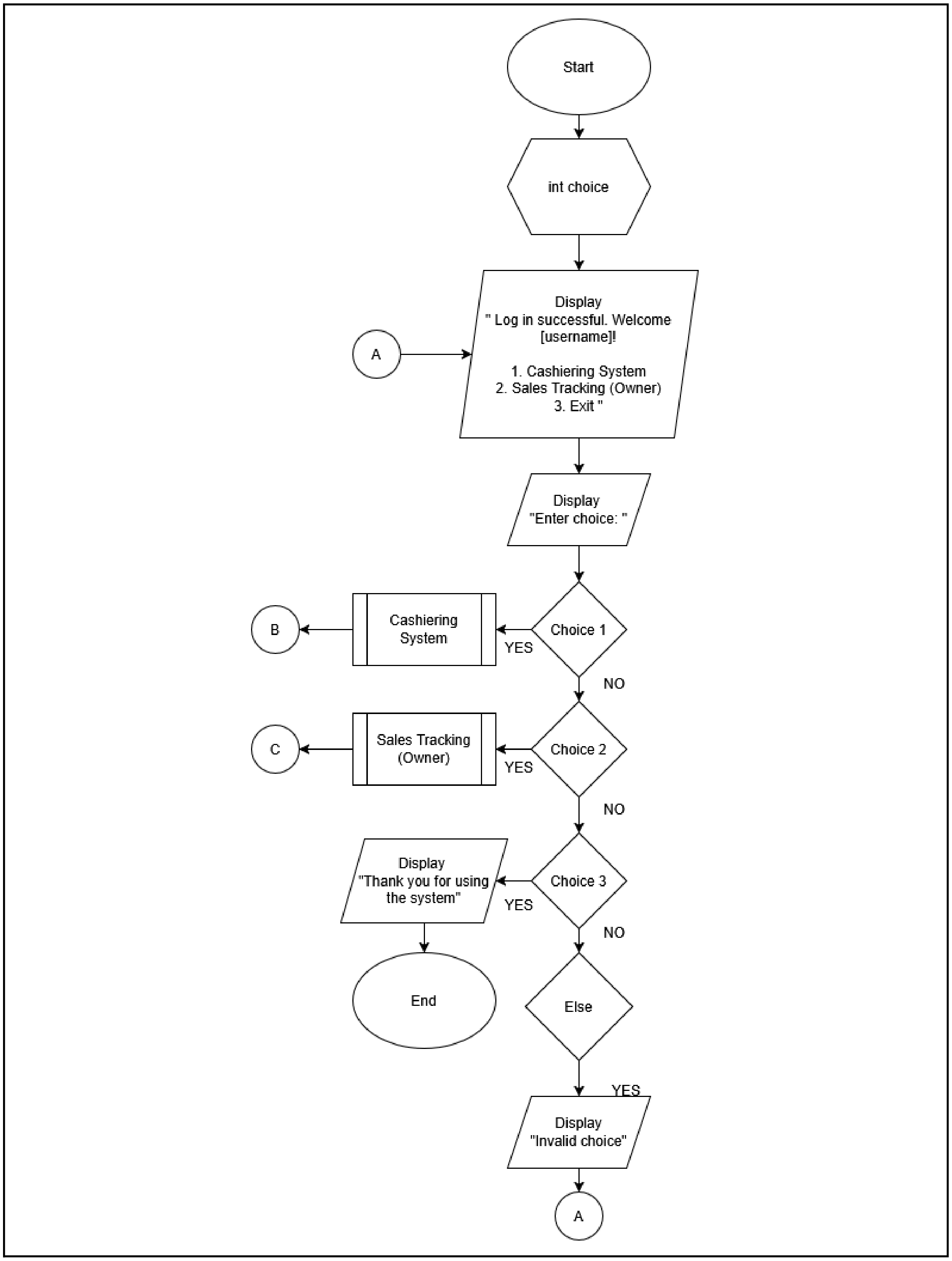
Start
int choice
Display
" Log in successful. Welcome [username]!
1. Cashiering System
2. Sales Tracking (Owner)
3. Exit "
A
Display
"Enter choice: "
Choice 1
YES
NO
Cashiering System
B
Choice 2
Sales Tracking (Owner)
C
Choice 3
Display
"Thank you for using the system"
End
Else
Display
"Invalid choice"
A

Figure 1: <Menu>

Figure 1 illustrates the menu of the system which involves the user choosing between three main functions. The system displays “Enter choice” to clarify the user in which of the following options he/she wants to do. Following that are 4 consecutive decisions the system makes in order to know which route/ path it should take. Choosing 1 will take the user to the “Cashiering System”, 2 will take the user to the “Sales Tracking”, 3 will then take the user to a thank you message and ending the program. Meanwhile if the user chooses any number higher than 3 it will display “Invalid choice” and take the user back to the main three functions of the system.

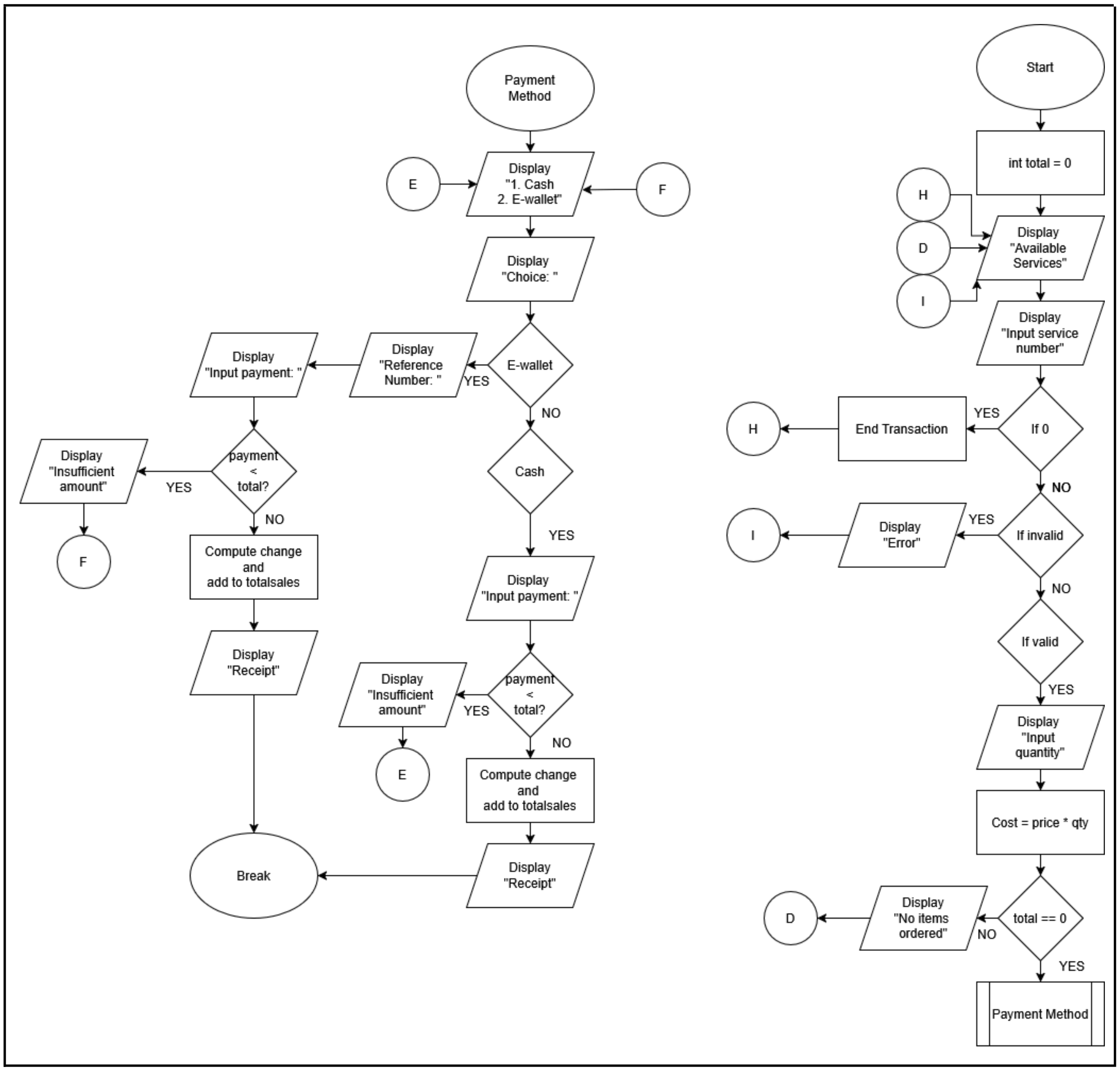
Payment Method
E
Display "1. Cash 2. E-wallet"
F
Display "Choice: "
Display "Input payment: "
Display "Reference Number: "
E-wallet
YES
NO
Display "Insufficient amount"
payment < total?
YES
NO
F
Compute change and add to totalsales
Cash
YES
Display "Input payment: "
Display "Receipt"
Display "Insufficient amount"
payment < total?
YES
NO
E
Compute change and add to totalsales
Break
Display "Receipt"
Start
int total = 0
H
D
I
Display "Available Services"
Display "Input service number"
H
End Transaction
YES
If 0
NO
I
Display "Error"
YES
If invalid
NO
If valid
YES
Display "Input quantity"
Cost = price * qty
D
Display "No items ordered"
NO
total == 0
YES
Payment Method

Figure 2: < Cashiering System >

Going from right to left for figure 2 we start off with the initialization total being zero followed by the display of the available printing types, photocopy along with their corresponding prices. Continuing further is the display of “Input service number” for the system to know what to run. If the user chooses “0” it will end the transaction and return to the display of available services. After that it will display an “Error” if the user chooses any number higher than 6. Lastly, the system continues on displaying an “Input quantity” if the user chooses a number between 1-6.

After entering a quantity it will compute the total cost, but if the entered quantity is “0” it will display “No items ordered” and return to the available services. Else It will proceed with the payment method which is split into two, e-wallet or cash. While the two have the same structure the only difference is that there would be a verification on the reference number for the e-wallet but other than that the structure of the payment method does not have much of a difference which ends to displaying the “Receipt” and breaking the sub-function right after.

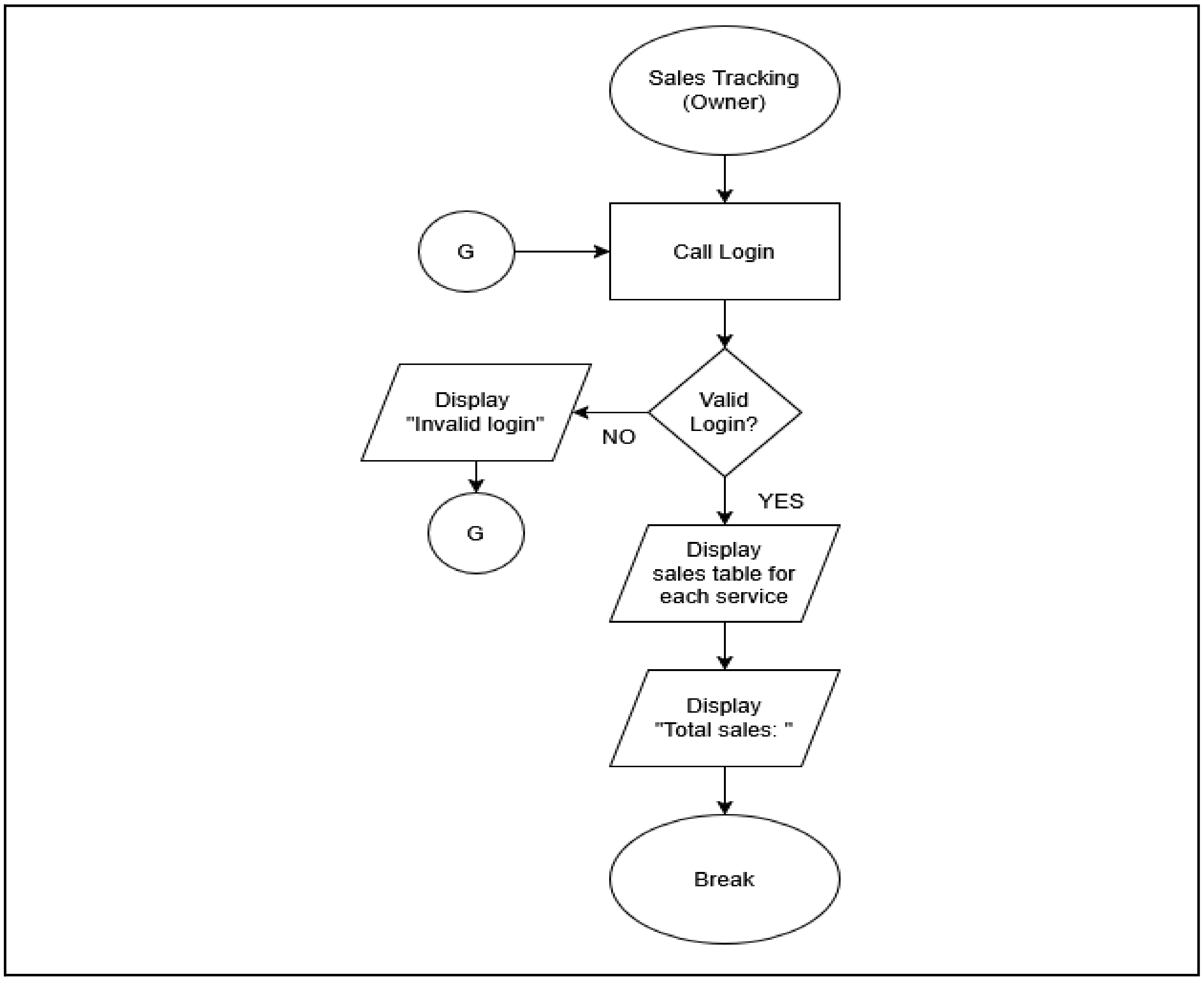

Figure 3: < Sales Tracking (Owner)>

The 3rd figure represents the shortened system of “Sales Tracking”. It starts off with a quick login system to check the validity of the entered details, if it is not valid it would then display “Invalid login” followed by going directly back to the login system. If its valid i would then display the tabular format of the sales for each format and would display the “Total sales: “ at the bottom of the table which would then lead to a break on the sub-function of the sales tracking.

Pseudocode

The pseudo code of the system follows a specific but very simple way of presenting itself. Beginning with a “start” followed by the menu with its following choices. Just like in the flowchart it goes through a lot of decisions to continue in which the majority uses “if” to separate and decide.

```
Start
  Display "Ikeu Printing Shop"
    Call Mainmenu
  Display "Enter choice: "
    If == 1
      Call Cashiering
        Display "Choose a service number: "
      If choice == 0
        Return to main menu
      If choice > 7
        Display "Invalid choice"
          Return to menu
      If choice <= 6
        Display "Enter quantity"
          Process cost = price * qty
          Process add cost to total amount
      REPEAT until finished with adding services
        Display "Total amount: "
      Display "Payment method (1-Cash),(2-E-Wallet): "
        If choice == 1
          Display "Enter amount"
            If payment < total amount
              Display "Insufficient payment. Transaction cancelled."
            Else
              Process calculate and show the change
          Display "Receipt"
            Return to menu

        If choice == 2
          Display "Enter reference number: "
          Display "Enter amount: "
            If payment < total mount
              Display "Insufficient payment. Transaction cancelled."
            Else
              Process calculate and show the change
          Display "Receipt"
            Return to menu
```

```
If choice == 2
  Display "Username: \n Password: "
    If details are incorrect
      Display "Invalid login"
    If details are correct
      Display sales report
      Display total sales
        Return to menu

If choice == 3
  Display "Thank you for using the system!"
    End the program

If choice > 3
  Display "Invalid choice!"
    Return to menu
```

Figure 4: Pseudo code of <system>

The pseudocode for the Ikeu Printing Shop System was designed to outline the flow of the program in a way that could be efficient and simple. It starts with defining the structure, beginning from the display of the menu and user choices, followed by decisional steps that determine which part of the program will run. Either the cashiering system, sales tracking, or exit. Each section of the pseudocode is written in a simple manner while still involving some technical terms like if-else for people who do not completely understand coding. The cashiering focuses on repetitive actions, such as entering services and calculating the total amount, while the sale tracking includes a login and report generation.

Data Dictionary

This Study shows the different Data Dictionary of cashiering system for Ikeu Printing shop. The Data dictionary explains all the important data that is used in the program for cashiering system, including their names, sizes, types, and functions. It aims to provide the readers to understand different data that is used in the cashiering program.

The table below shows the list of all the different data that is used in the program. Each item is described based on what it does in the system.

Table 1: Data Dictionary

Data Name	Size	Data Type	Description
1. Cashier ()		void	The transaction process, from purchasing an item to payment and printing receipt.

2. charge	8 bytes	double	Amount of money returned to the customer after the payment for the items.
3. choice	4 bytes	int	Used by the user to choose menu option or select service.
4. Data	64 bytes	Structure (struct)	Stores info about each service the shop offers, like name, price, sold items, and total sales.
5. Login ()		bool	Used to verify the owner's username and password to access the sale report.
6. Main ()		int	Function that serves as the system's starting point and displays the main menu.
7. name	32 bytes	string	The name or type of service.
8. numServices	4 bytes	int	The total number of services available in the printing shop.
9. pass	32 bytes	string	The password of the owner to login.
10. payType	4 bytes	int	Type of payment method (1 for Cash, 2 for E-Wallet).
11. payment	8 bytes	double	The amount of money entered by the customer to pay for their purchase.
12. price	8 bytes	double	The cost of one unit (fixed standard of quality) of that service.

13. qty	4 bytes	int	Number of items or quantity of service that the customer wants to purchase.
14. ref	32 bytes	string	The reference number for E-Wallet transaction.
15. salesReport		void	Track sales, displaying total sales and number of services.
16. Sales	8 bytes	double	Total amount of money earned from a specific service.
17. Service []	432 bytes	Array	List that stores all the available services offered by the printing shop.
18. showMenu()		void	Displays the list of available services and prices.
19. Sold	4 bytes	int	Tells how many times the service has been purchased or used.
20. total	8 bytes	double	The total amount to be paid by the customer for their chosen services.
21. totalSales	8 bytes	double	Total income of all services combined.
22. user	4 bytes	string	Username for the owner to login for checking sales report.

Code

The program is a simple system that allows the owner and his/her employees to manage transactions and monitor the sales. It uses a menu design, allowing the printing shop to easily choose which function it should run. The menu provides three choices:

1. Cashiering System
2. Sales Tracking (Owner)
3. Exit

Each part of the program is for a specific function which displays loops, conditions, and functions.

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

struct Service {
    string name;
    double price;
    int sold;
    double sales;
};

Service services[] = {
    {"Photocopy - Short", 2, 0, 0},
    {"Photocopy - A4/Long", 4, 0, 0},
    {"Print B&W - Short", 3, 0, 0},
    {"Print B&W - A4/Long", 5, 0, 0},
    {"Print Colored - Short", 10, 0, 0},
    {"Print Colored - A4/Long", 15, 0, 0}
};

int numServices = 6;
double totalSales = 0;

// --- FUNCTION PROTOTYPES ---
void showMenu();
void cashier();
void salesReport();
bool login();

int main() {
    int choice;
    do {
        cout << "\n=== Ikeu Printing Shop ===\n";
        cout << "1. Cashiering System\n";
        cout << "2. Sales Tracking (Owner)\n";
        cout << "3. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        if (choice == 1) cashier();
        else if (choice == 2) salesReport();
        else if (choice == 3) cout << "Thank you for using the system!\n";
        else cout << "Invalid choice!\n";
    } while (choice != 3);
    return 0;
}

void showMenu() {
    cout << "\n--- Available Services ---\n";
    for (int i = 0; i < numServices; i++)
        cout << i + 1 << ". " << left << setw(25)
             << services[i].name << "₱" << services[i].price << endl;
    cout << "0. Finish Transaction\n";
}

void cashier() {
    int choice, qty;
    double total = 0;

    showMenu();
    while (true) {
        cout << "\nEnter service number (0 to finish): ";
        cin >> choice;
        if (choice == 0) break;
        if (choice < 1 || choice > numServices) {
            cout << "Invalid choice!\n";
            continue;
        }
        cout << "Enter quantity: ";
        cin >> qty;
        double cost = services[choice - 1].price * qty;
        total += cost;
        services[choice - 1].sold += qty;
        services[choice - 1].sales += cost;
        cout << "Added " << qty << " x " << services[choice - 1].name
             << " = ₱" << cost << endl;
    }

    if (total == 0) {
        cout << "No items ordered.\n";
        return;
    }

    cout << "\nTotal amount: ₱" << total << endl;
    int payType;
    string ref = "N/A";
    cout << "Payment Method (1-Cash, 2-E-Wallet): ";
    cin >> payType;
    if (payType == 2) {
        cout << "Enter E-Wallet Reference Number: ";
        cin >> ref;
    }

    double payment;
    cout << "Enter payment: ₱";
    cin >> payment;
    if (payment < total) {
        cout << "Insufficient payment. Transaction cancelled.\n";
        return;
    }

    double change = payment - total;
    totalSales += total;

    cout << "\n--- RECEIPT ---\n";
    cout << "Total: ₱" << total << endl;
    cout << "Payment: ₱" << payment << endl;
    cout << "Change: ₱" << change << endl;
    if (payType == 2) cout << "E-Wallet Ref: " << ref << endl;
    cout << "Transaction Complete!\n";
}

bool login() {
    string user, pass;
    cout << "\n=== OWNER LOGIN ===\n";
    cout << "Username: ";
    cin >> user;
    cout << "Password: ";
    cin >> pass;
    if (user == "Ikeu" && pass == "123") return true;
    cout << "Invalid login!\n";
    return false;
}

void salesReport() {
    if (!login()) return;

    cout << "\n--- SALES REPORT ---\n";
    cout << left << setw(25) << "SERVICE"
         << setw(10) << "SOLD"
         << setw(10) << "SALES(₱)\n";
    cout << "---------------------------------\n";

    for (int i = 0; i < numServices; i++) {
        cout << left << setw(25) << services[i].name
             << setw(10) << services[i].sold
             << setw(10) << fixed << setprecision(2)
             << services[i].sales << endl;
    }

    cout << "---------------------------------\n";
    cout << "TOTAL SALES: ₱" << totalSales << endl;
}
```

Figure 3. <name of figure>

Code discussion

At the beginning of the code it uses a few libraries which includes:

<iostream> for input and output operations,
<iomanip> for text formatting, and
<string> for string data types.

A structure named "Service" is declared to store details about the services offered by the printing shop. It contains the service's name, price, number of items sold, and total sales. An array of "Service" objects stores six different services, such as "Photocopy - Short" and "Print Colored - A4/Long." Two variables such as "numServices" and "totalSales" are used to track the number of services and the shop's total earnings. This allows the program to handle multiple services without using complex databases.

The cashiering function is responsible for processing customer transactions. It displays all the available services and their prices using the "showMenu()" function. The program would then repeatedly ask the user to enter service number and quantity. Where in each valid input updates the number of items sold and calculates the total cost for that transaction.

Once all services have been selected, the program asks for a payment method which shows "cash" or "E-wallet." If the customer chooses an E-wallet, they are required to input a reference number. Afterward, the program accepts the payment amount, checks if it is sufficient, and computes the change. A receipt is then displayed, showing the total payment, change, and E-wallet reference number if applicable.

The sales tracking system is designed for the owner to view sales performance. Before accessing the sales report, the program requires the owner to log in using a predefined username ("Ikeu") and password ("123"). This feature is done by the "login()" function, which verifies the input credentials. If the login fails, the program displays an "Invalid login" message and returns to the main menu. Otherwise, it proceeds to display the report.

The "salesReport()" function generates a table showing each service's name, the total quantity sold, and the total amount earned. It uses the "<iomanip>" library to properly align the text and numbers in columns. The report ends by showing the overall total sales computed from all transactions.

This function serves as the control center of the program. It uses a do-while loop to repeatedly display the main menu until the user chooses to exit. Inside the loop, the program uses an if-else structure to determine whether to open the cashiering system, sales report, or exit the program. When the user chooses to exit, the program displays a short thank-you message and ends.

Results and Discussion

This chapter presents the results obtained after the development and testing of the Cashiering System created for Ikeu Printing Shop. The system was programmed using C++ with the purpose of improving the cashiering process by minimizing manual tasks and providing a faster and more organized way of handling customer transactions. The discussion below highlights how each major feature of the system functioned during the testing phase and how it contributed to achieving the intended objectives of the project.

Throughout the testing phase, the system effectively displayed the range of services available in the printing shop, including photocopying and printing options, whether in colored or black and white formats. Users could select the services they required, input the number of copies, and the system automatically computed the total amount due. Prior to completing the transaction, the user was prompted to select their preferred mode of payment, choosing between cash and a cashless option. When cashless payment was chosen, the system processed the transaction through an e-wallet method. After the payment was confirmed, a receipt was generated containing the summary of the purchased services, quantities, total cost, and the selected payment method. This automated process reduced potential human error, ensured pricing accuracy, and made transaction handling more convenient for both the cashier and customer.

The Sales tracking feature was also tested to verify its effectiveness in recording daily sales. This function could only be accessed through a secure login intended for the shop owner. Once logged in, the system displayed the total number of services rendered and total earnings accumulated within the day. This feature provided the owner with an overview of business performance and served as a reference for monitoring sales trends. The stored sales data could also be reviewed when needed for documentation or evaluation purposes, making it easier for the owner to keep track of daily income without relying on manual record-keeping.

Conclusion

In conclusion, this C++ software successfully illustrates the menu-driven programming to carry out crucial business tasks. Each choice has its own purposes and implementations wherein the Cashiering System enables users to manage transactions effectively while the Sales Tracking offers insights regarding the overall total sales and for the performance monitoring. Lastly, is the Exit that guarantees a simple and user-friendly way to safely quit the program. Overall, this project emphasizes data processing, logical flow control, and user interaction in console-based systems, demonstrating the useful application of C++ in creating effective business management tools. Future efforts should focus on integrating advanced technologies, expanding digital infrastructure, and tailoring interventions to enhance the growth of e-commerce and digital payments in the Philippines (Anne, 2024).

References

Asim, A. A. R., Baer, R. J. B., Cisneros, B., & Genelza, E. D. (2022). Point of sale system for proper inventory management for Mac's Food Business: A project proposal for partial fulfillment of the subject CpET 8L – Software Design for Technological University of the Philippines. Technological University of the Philippines.https://www.studocu.com/ph/document/technological-university-of-the-philippines/computer-engineering-technology/chapter-1-pos-system-and-inventory-management-for-business-industry-and-so-on/28337601

Cambria Software. (2024). Business app projects with C & C++ programming. Cambria Corporation.

https://www.cambria.com/learn-programming/cpp-business-application-projects

cplusplus.com. (2024). C++ data types. https://cplusplus.com/doc/tutorial/variables/

cppreference.com. (2024). Fundamental types (C++). https://en.cppreference.com/w/cpp/language/types

Gabriela Anne. (2024). E-Commerce Growth and Digital Payments in the Philippines. Research Gate. https://www.researchgate.net/publication/382727808_E-Commerce_Growth_and_Digital_Payments_in_the_Philippines

GeeksforGeeks. (2025, September 20). Structures in C++.

https://www.geeksforgeeks.org/cpp/structures-in-cpp/

Genie, N. A., & Sharma, N. (2024, June 20). Nimble AppGenie. Nimbleappgenie.

https://www.nimbleappgenie.com/blogs/ewallet-apis/

PeerDH. (2024). Building a simple point of sale system in C++.

https://peerdh.com/blogs/programming-insights/building-a-simple-point-of-sale-system-in-c

Pekkala, A. (2024, September 26). Cash management for 2025 and beyond. Nomentia.com; Nomentia.

https://www.nomentia.com/blog/the-future-of-corporate-cash-management-2025

Pekkala, A. (2024, October 31). Cash management trends in 2025. Nomentia.com; Nomentia.

https://www.nomentia.com/blog/the-cash-management-trends-of-2025

rnshalinda. (2024). Restaurant bill calculator and purchase system in C++ [Source code]. GitHub.

https://github.com/rnshalinda/Restaurant-Bill-Calculator-Cashier-system

SDLC Iterative Model. (2025). W3schools.in. https://www.w3schools.in/sdlc/iterative-model soulslicer. (2023). Cashier system written in C++ [Source code]. GitHub.

https://github.com/soulslicer/Cashier-System

SourceCodester. (2023). Simple POS system in C free source code.

https://www.sourcecodester.com/cc/16735/simple-pos-system-c-free-source-code.html

TylerMSFT. (2025, September 6). Create a Traditional Windows Desktop Application (C++). Microsoft.

https://learn.microsoft.com/en-us/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-170

W3Schools. (2024). C++ arrays.

https://www.w3schools.com/cpp/cpp_arrays.asp

W3Schools. (2024). C++ data types.

https://www.w3schools.com/cpp/cpp_data_types.asp

W3Schools. (2024). C++ functions.

https://www.w3schools.com/cpp/cpp_functions.asp

W3Schools. (2024). C++ strings.

https://www.w3schools.com/cpp/cpp_strings.aspp