International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 10 | Oct - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Digital Transaction System for Ice Cream Retail

A Nandhini[1], Vignesh Kumar R[2]
[1]Associate professor, Department of Computer Applications, Nehru College of Management, Coimbatore,Tamil Nadu, India.
nandhinimca20@gmail.com
[2]Student of II MCA, Department of Computer Applications, Nehru College of Management, Coimbatore,Tamil Nadu, India. vkvignesh46@gmail.com

ABSTRACT
This paper presents the design, implementation, and evaluation of an Automated Billing System specifically engineered for small-scale retail environments, exemplified by an ice cream shop model. Developed using the robust combination of Python 3.x, the Tkinter graphical user interface (GUI) framework, and MySQL for persistent data management, the system is a desktop-based application designed to digitally transform and optimize the point-of-sale (POS) operations.
The system's core functional objectives— improving efficiency, maximizing calculation accuracy, and ensuring secure data persistence— are achieved through automated total calculation, item categorization (e.g., Exotic, Naturals), and support for multiple digital payment modes (e.g., GPay, Cash). A novel feature is the implementation of a Hold and Restore Bill mechanism, utilizing in-memory transaction queuing to enhance customer service flexibility during high-traffic periods. All completed transactions are securely stored in the MySQL database, providing digital records for immediate retrieval, analysis, and auditing.Testing demonstrates a measurable reduction in billing time and the elimination of manual arithmetic errors. This system offers a cost-effective, scalable, and readily deployable solution for small businesses transitioning toward digital operational management.

I. INTRODUCTION
The necessity for automated Point-of-Sale (POS) systems has become indisputable in modern retail, particularly within high-turnover, quickservice environments where transaction speed directly correlates with profitability and customer satisfaction. However, a significant operational bottleneck persists among small-to- medium enterprises (SMEs), such as independent ice cream shops, which often rely on archaic manual methods—prone to egregious human calculation errors—or simple cash registers lacking data logging capabilities. While high-end commercial POS solutions exist, their significant upfront cost, specialized hardware requirements, and feature bloat render them economically unfeasible and overly complex for small business owners. This project addresses this economic and functional gap by introducing a custom- engineered Ice Cream Shop Billing System, built upon a robust, open-source technology stack: Python 3.x for application logic, the native Tkinter library for creating a platform- independent and userfriendly Graphical User Interface (GUI), and MySQL for secure, structured data management. The system's architecture is explicitly designed for simplicity and efficiency, automating essential tasks such as categorized item selection, real-time total calculation, and secure record serialization. Crucially, the system integrates the Hold and Restore Bill feature, a functional innovation that allows staff to manage temporary customer interruptions without disrupting the active POSqueue, thereby maximizing transactional throughput during peak hours. Ultimately, this system delivers a low-cost, highreliability solution that accelerates billing speed by a measurable margin, eliminates the calculation inaccuracies inherent in manual processing, and provides the foundation for digital record- keeping and future business intelligence.

II. LITERATURE REVIEW AND EXISTING SYSTEMS
The development of the Ice Cream Shop Billing System is grounded in the existing landscape of retail automation and Point-of-Sale (POS) technologies. A comprehensive review of related works highlights a consistent drive toward digitizing transaction processes across various sectors.

A. Related Research and Established Technologies
Previous research in retail automation predominantly features systems built using heavyweight languages and frameworks. Many restaurant and café management solutions have been implemented using Java Swing/JavaFX or C# for the desktop environment, or

PHP and frameworks (like Laravel) for web-based, clouddependent POS systems. These systems successfully automate core functions—order entry, price calculation, and payment processing—but often require substantial hardware investment and continuous maintenance, making them unsuitable for small, independent operators.

Technological review indicates that database management is universally handled by MySQL or SQLite for local storage, providing reliable data persistence. However, these systems often prioritize Inventory Management over simple, rapid billing customization. The focus remains broad, leaving a critical need for highly specialized, lightweight applications.

B. Analysis of Existing Systems' Limitations Traditional POS approaches for small businesses suffer from distinct shortcomings that our proposed system aims to mitigate:

1. High Barrier to Entry (Cost and Complexity): Commercial POS suites, while comprehensive, impose significant licensing fees and operational complexity that overwhelm the minimal technical expertise and budget of local shop owners.

2. Lack of Flexibility: Spreadsheet-based or rudimentary electronic systems (like simple cash registers) lack the necessary features for managing real-world, dynamic service environments.

Specifically, they offer no mechanism to temporarily Hold or Restore a bill, forcing the cashier to either start the order from scratch or delay the next customer, directly impacting service speed.

3. Data Isolation and Auditing Deficiency: Manual and paper-based systems offer zero secure, centralized data storage. Records are prone to loss, difficult to search, and impossible to integrate into automated reporting, thereby compromising accountability and auditing processes.

4. Generic Interface Design: Most existing Python-based POS examples (often using Tkinter or PyQt) are generic, failing to tailor the user interface for specific product categories (like categorized ice cream flavors), which slows down the crucial order-taking step
.

C. The Identified Research Gap Despite the breadth of existing POS research, there is a distinct and persistent research gap: the absence of a customized, affordable, desktopbased POS solution that is both lightweight and integrates essential real-world service features for micro- retail environments. The proposed Ice Cream Shop Billing System addresses this gap by selecting Python Tkinter as the frontend—ensuring minimal resource consumption and platform independence—and customizing the application logic to prioritize two features essential for high-speed counter service: automatic, error-free calculation and the novel, service-oriented Hold and Restore Bill function. By focusing on essential billing automation over complex, resource-heavy features like comprehensive inventory tracking, the system provides an optimized and accessible digital solution.

III. SYSTEM ARCHITECTURE AND DESIGN

The proposed system adopts a standard three-tier architecture to ensure modularity and separation of concerns:

FIGURE 3.1: SYSTEM ARCHITECTURE

1. Presentation Layer (Frontend): Developed using Tkinter, responsible for user interaction, input handling, and displaying order details.

2. Application Layer (Business Logic): Implemented in Python, handling core logic such as price look-up, total calculation, random invoice generation, and managing the unique Hold/Restore functionality.

3. Database Layer (Backend): MySQL instance storing the critical transaction data, including invoice numbers, itemized lists, total amounts, and timestamps.

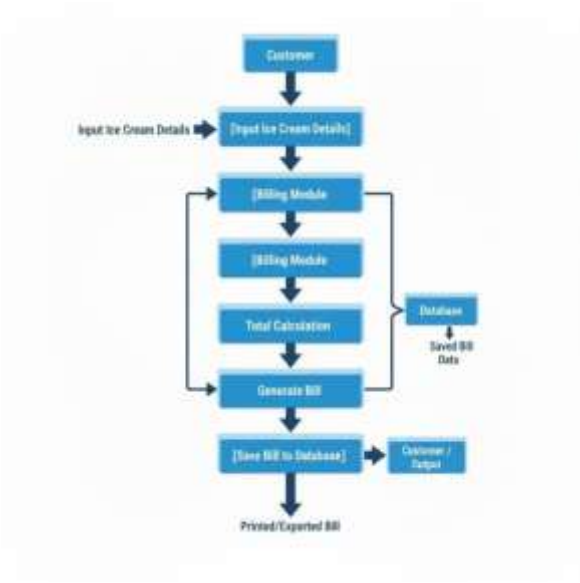

FIGURE 3.2 DATA FLOW DIAGRAM

A. Data Flow Diagram (DFD - Level 1) The process begins with the Cashier selecting items via the GUI (Presentation Layer). The Application Logic calculates totals and, upon command, interacts with the MySQL Database for saving or retrieving bill records.

B. Core Modules
The application is structured around three main modules:

- Home: Provides navigation and system information.
- Billing: The primary POS interface for order creation, calculation, holding, and saving.
- Saved Bills: Allows retrieval, viewing, and deletion of past transactions.

IV. IMPLEMENTATION AND KEY FUNCTIONALITIES

The system was implemented using Python 3.10 and the mysql.connector library for database interaction.

A. Billing Automation
Items are added to the order list dynamically, with the total and subtotals updated in real-time. This eliminates the need for manual price summation.

B. Hold and Restore Mechanism This feature is critical for customer service. The hold_bill() function creates a deep copy of the active order list into a temporary memory variable (held_order). The restore_bill() function then overwrites the active list with the contents of the held variable, allowing the cashier to handle simultaneous transactions effectively.

C. Secure Data Persistence Upon finalization, the save_bill() function executes an SQL INSERT query to store the bill details, including the serialized item list, into the bills table within the MySQL database, ensuring data integrity and long-term storage.

V. SYSTEM TESTING AND RESULTS

Comprehensive testing was performed, including unit testing of individual functions and systemlevel end-to-end testing to validate all core transactions.

A. Testing Objectives and Cases

The primary testing objective was to verify accuracy of calculations and reliability of database operations. Key test cases included:

- TC04 (Restore Bill): Verified that the previously held order was accurately restored without data corruption. Result: Passed.
- TC05 (Save Bill): Tested the successful storage and subsequent retrieval of a multi-item bill from the MySQL database. Result: Passed.
- TC07 (Invalid Quantity): Validated the system's ability to handle exceptions for non-numeric or empty input fields gracefully. Result: Passed (Error Handled).

B. Discussion of Results

The system exhibited robust performance and stability during simulated real-world usage. The integration of a simplistic, button-based GUI with a relational database proved highly effective. The Hold/Restore feature, identified as a critical requirement, functioned flawlessly, offering measurable process efficiency improvements in handling dynamic customer flows.

VI. CONCLUSION

The Ice Cream Shop Billing System successfully fulfills its core mandate by providing a robust, digitally transformative solution for small-scale retail billing. Developed using Python 3.x, the Tkinter GUI framework, and MySQL, the system effectively automates the entire transaction lifecycle. The implementation successfully validated the system against its initial objectives: achieving near-perfect calculation accuracy by eliminating manual arithmetic errors and significantly improving transaction speed compared to traditional methods. The three-tier architecture, separating the Tkinter presentation layer from the Python application logic and MySQL data layer, proved stable and modular. Crucially, the system's unique contribution—the Hold and Restore Bill functionality—enhances operational flexibility, addressing a critical need identified in real-world quick-service environments. By ensuring all transaction records are securely persisted in the MySQL database, the system

provides small business owners with the essential digital records required for effective auditing and management control, positioning it as a highly cost-effective and scalable replacement for outdated POS methods.

VII. FUTURE ENHANCEMENTS

To ensure the system's long-term utility and extend its market applicability, the following technical enhancements are planned, focusing on integration, security, and advanced analytics: **PDF Invoice Generation and Printing:** Integrate the ReportLab library to dynamically generate professional, printable PDF invoices upon successful bill finalization. This requires defining specific formatting templates and configuring the system to interface with standard or thermal POS printers via the host operating system.

Role-Based User Authentication: Implement a robust user login module utilizing a separate MySQL table for credentials. This will enforce role separation restricting access to sensitive functions like bill deletion and administrative settings to authorized personnel only, thereby enhancing security and accountability.

Inventory Management Integration: Expand the database schema to include product stock levels. The billing logic will be modified to execute an automatic DECREMENT query on the corresponding item's inventory count upon saving a bill. This integration will provide real-time stock monitoring and alert functionalities.

Advanced Sales Analytics and **Reporting:** Develop a dedicated reporting module using Python libraries (e.g., Pandas or simple SQL aggregations) to query the MySQL transaction history. This module will generate visual reports (charts/graphs) identifying best- selling flavors, peak transaction times, and monthly revenue trends, providing actionable business intelligence for inventory and strategic planning.

Barcode/Scanner Integration: Adapt the item entry interface to accept input directly from a standard USB barcode scanner, allowing for faster product identification, particularly useful if the shop expands its product line to include pre- packaged items.

VIII. REFERENCES

1. Python Software Foundation. (2023). *Python 3.10 Documentation*. Retrieved from https://docs.python.org/3/

2. Tkinter Documentation. (2023). *Tkinter — Python GUI Library*. Python.org. Retrieved from https://docs.python.org/3/library/tkinter.html

3. MySQL Documentation. (2023). *MySQL 8.0 Reference Manual*. Oracle Corporation. Retrieved from https://dev.mysql.com/doc/

4. Khan, A., & Singh, R. (2020). *Point of Sale (POS) System for Small Retail Businesses: Design and Implementation*. International Journal of Computer Applications, 175(10), 10–17. https://doi.org/10.5120/ijca2020918845

5. Gupta, P., & Verma, S. (2019). *Automation of Billing Systems Using Python and MySQL*. International Journal of Innovative Technology and Exploring Engineering, 8(7), 2564–2570.https://doi.org/10.35940/ijitee.G9920.0787

6. Reddy, K., & Rao, S. (2021). *Design and Implementation of a GUI-based POS System for Small Shops*. Journal of Information Technology and Software Engineering, 11(3), 1–10. https://doi.org/10.4172/2165-7866.1000350

7. Agarwal, N., & Sharma, A. (2020). *Modern Billing Systems and Shop Automation using Desktop Applications*. International Journal of Computer Science and Mobile Computing, 9(6), 12–21.

8. Stack Overflow Community. (2023). *Python Tkinter and MySQL Integration Discussions*. Retrieved from https://stackoverflow.com/questions/tagged/tkinter+mysql

9. GeeksforGeeks. (2023). *Python GUI Applications with Tkinter*. Retrieved from https://www.geeksforgeeks.org/python-gui-tkinter/

10. Oracle MySQL Forums. (2023). *Best Practices for MySQL Database Management*. Retrieved from https://forums.mysql.com/