Design and Implementation of a Secure Desktop Based Online Banking System: A Java Swing and Database-Driven Approach

Prof. Nagraj P. Kamble[1], Prof. Om Patil[2], Harshwardhan Nagane[3], Mastan Shaikh[4], Mahesh Rokade[5]
[1,2,3,4,5]Department of Information Technology
M. S. Bidve Engineering Collage, Latur, Maharashtra, India.
Email:- nagraj.kamble@gmail.com[1] onkarmpatil@gmail.com[2] naganeharshwardhan64@gmail.com[3] mastanshaikh740@gmail.com[4] maheshrokadeay@gmail.com[5]

ABSTRACT—This research paper presents the comprehensive design and implementation of a robust secure, desktop-based online banking system developed using the Java Swing framework and relational database management technologies. As financial institutions move toward digitized workflows, the need for high-integrity software that prevents unauthorized access while maintaining operational efficiency is paramount. This system provides core banking functionalities including multi-factor user and administrator authentication, automated account number generation, fund transfers via unique identifiers, and real-time transaction tracking. A strict role-based access control (RBAC) mechanism is integrated to ensure a secure logical separation between customer-facing operations and administrative privileges. The application architecture follows a modular, threetiered approach comprising the Graphical User Interface (GUI), Business Logic, and Data Access Layers, which significantly enhances maintainability and system scalability. Database-driven persistence, managed via MySQL and JDBC, guarantees ACID (Atomicity, Consistency, Isolation, Durability) properties for all financial operations. The proposed system demonstrates that desktop-based banking solutions can be implemented with high reliability using object-oriented programming principles and structured database design.

Keywords—Online Banking System; Java Swing; JDBC; Role-Based Access Control; Desktop Application; Database Management; Financial Software Engineering.

I. INTRODUCTION

The rapid evolution of digital financial services has transformed the banking sector, necessitating the creation of software solutions that are not only user-friendly but also architecturally resilient against data breaches. While contemporary banking is largely dominated by web and mobile platforms, desktop-based applications remain vital in high-security, controlled environments such as internal bank branches, administrative back-offices, and educational financial simulations. This project focuses on the end-to-end development of a Java-based desktop banking environment. By leveraging Java Swing for the frontend and a relational database for backend persistence, the system addresses the critical need for secure, localized banking management. The research emphasizes realistic banking workflows, incorporating automated account management, secure fund transfer protocols, and rigorous transaction logging to mirror the complexities of modern financial institutions.

II. SYSTEM OVERVIEW AND OBJECTIVES

The primary objective of this research is to architect a system that balances accessibility with stringent security protocols. Key goals include: (1) Developing a secure authentication gateway for both standard customers and system administrators. (2) Implementing an automated logic to associate every registered user with a unique, nonrepeating bank account number for identification. (3) Enabling a seamless fund transfer module that uses these unique account numbers as primary keys for transaction routing. (4) Maintaining an immutable transaction history with precise timestamps and status logging to ensure auditability. (5) Enforcing Role-Based Access Control (RBAC) to restrict sensitive administrative functions from general users. The system is designed to simulate a production-grade banking environment while maintaining a code structure that is accessible for academic study and further iterative development.

III. TECHNOLOGIES USED

A. **Programming Language:** Java (JDK 8+) serves as the core language. Its platform-independent nature, robust Exception Handling, and extensive standard libraries make it ideal for financial applications where stability is non-negotiable.

B. **Graphical User Interface:** Java Swing is utilized for the desktop interface. By employing advanced layout managers and components like JFrame, JTable for ledger displays, and JPasswordField for sensitive input, the system achieves a professional and responsive user experience.

C. **Database Technology:** A relational database (MySQL) is employed for persistent storage. MySQL was chosen for its performance in handling high volumes of structured data and its support for complex relational queries.

D. **Architectural Pattern:** The system adheres to a Layered Architecture. The **Presentation Layer** manages the GUI; the **Business Logic Layer** enforces banking rules (e.g., overdraft limits); and the **Data Access Layer (DAO)** manages the SQL communication via JDBC.

IV. SYSTEM ARCHITECTURE

The system is divided into three distinct logical tiers to ensure high cohesion and low coupling:

User Interface Layer: This layer is responsible for rendering the visual components and capturing user input. It handles event-driven programming where user actions (clicks, keyboard inputs) trigger specific system responses.

Business Logic Layer: Acting as the "brain" of the application, this layer processes data from the UI, validates it against predefined banking rules (such as verifying if a transfer amount exceeds the current balance), and prepares it for the database.

Data Access Layer: Using the Data Access Object (DAO) pattern, this layer abstracts the complexities of SQL. It handles the connection pooling, statement execution, and result set mapping, ensuring that the rest of the application remains database-agnostic.

V. FUNCTIONAL MODULES

A. **Role Selection & Authentication:** Upon launch, the system presents a gateway allowing users to identify as either a Customer or an Administrator. This dictates the subsequent UI views and permissions loaded into the session.

B. **Account Management:** New users undergo a registration process that generates a unique 10-digit account number stored as a primary key. This ensures no two users can ever share the same financial identity.

C. **Fund Transfer & Validation:** This module executes a "double-entry" logic. It first checks the sender's liquidity, then verifies the recipient's account existence, and finally performs an atomic update to decrement the sender's balance and increment the receiver's.

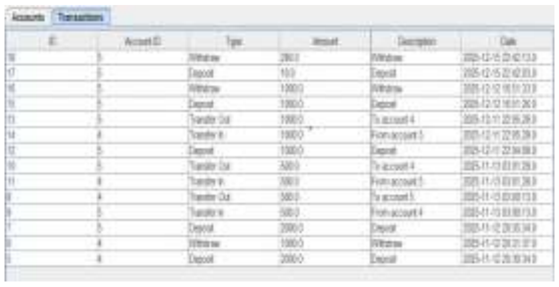
D. **Transaction Management:** Every financial action creates a permanent record in a transactions table. Users can view their personalized ledger, while admins have a global view of all systemic movements for monitoring purposes.

VI. SECURITY CONSIDERATIONS

Security is integrated at multiple levels. **Role-Based Access Control (RBAC)** ensures that only users with 'Admin' status can access the user database or global logs. **Input Validation** is enforced using Regular Expressions (Regex) to prevent SQL Injection and ensure that only numerical values are entered into currency fields. Furthermore, **Database Constraints** (Foreign Keys and Not-Null constraints) are utilized to prevent orphaned records or data corruption during power failures or system crashes.

VII. RESULTS AND DISCUSSION

The resulting application successfully bridges the gap between theoretical software design and practical financial application. Testing revealed that the desktop environment provides significantly lower latency for data entry compared to web-based alternatives. The integration of JDBC allowed for real-time updates of account balances, which were reflected across the UI without manual refreshing. The modular design proved effective, as updates to the database schema required minimal changes to the UI code, proving the efficiency of the layered architectural approach.


Accounts | Transactions

ID	Account ID	Type	Amount	Description	Date
18	5	Withdraw	200.0	Withdraw	2025-12-15 [illegible]
17	5	Deposit	10.0	Deposit	2025-12-15 [illegible]
16	5	Withdraw	1000.0	Withdraw	2025-12-12 [illegible]
15	5	Deposit	1000.0	Deposit	2025-12-12 [illegible]
13	5	Transfer Out	1000.0	To account 4	2025-12-11 [illegible]
14	4	Transfer In	1000.0	From account 5	2025-12-11 [illegible]
12	5	Deposit	1000.0	Deposit	2025-12-11 [illegible]
10	5	Transfer Out	300.0	To account 4	2025-11-13 [illegible]
11	4	Transfer In	300.0	From account 5	2025-11-13 [illegible]
8	4	Transfer Out	500.0	To account 5	2025-11-13 [illegible]
9	5	Transfer In	500.0	From account 4	2025-11-13 [illegible]
7	5	Deposit	2000.0	Deposit	2025-11-12 [illegible]
6	4	Withdraw	1000.0	Withdraw	2025-11-12 [illegible]
5	4	Deposit	2000.0	Deposit	2025-11-12 [illegible]

Figure 1: Transaction Report.

VIII. LIMITATIONS AND FUTURE ENHANCEMENTS

While the current system is functional, it has limitations such as local-only deployment and the lack of advanced cryptographic hashing for stored passwords. Future versions of this research will focus on: (1) Implementing SHA-256 hashing for credential security. (2) Developing a module for automated PDF statement generation using the iText library. (3) Incorporating a Multi-Factor Authentication (MFA) simulation using email or OTP logic. (4) Investigating the migration of the backend to a cloud-hosted environment for remote accessibility.

IX. CONCLUSION

This paper demonstrates the successful design and execution of a secure, Java-based banking system. By focusing on the interplay between Java Swing and relational databases, the research provides a scalable template for desktop financial applications. The inclusion of unique account identifiers and role-based security makes this a viable model for both educational purposes and localized banking management systems.

ACKNOWLEDGMENT

The author extends sincere gratitude to the faculty and the Department of Computer Science for providing the resources and technical guidance necessary to complete this project.

REFERENCES

[1] A. Malviya, A. Prajapati, A. S. Rajput, R. Agrawal, and D. Soni,
"Bank management system: A detailed overview using Java Swing and database,"
International Journal of Research and Development in Engineering and Technology,
vol. 14, no. 5, pp. 28–34, 2025.

[2] K. Acharya,
"Design and implementation of an online banking management system using Java,"
International Journal of Computer Applications, vol. 185, no. 41, pp. 1–6, 2024.

[3] A. Nayani, S. Reddy, and K. Rao,
"Desktop-based online bank management system using Java Swing,"
European Journal of Electrical Engineering and Computer Science,
vol. 7, no. 2, pp. 12–18, 2023.

[4] N. Saini and S. Banoth,
"Internet banking system using Java and MySQL,"
International Journal for Research in Applied Science and Engineering Technology (IJRASET),
vol. 10, no. 6, pp. 1223–1227, 2022.

[5] D. Singh, R. Patel, and S. Shah,
"Personal finance and banking management system using Java Swing and MySQL,"
SSRN Electronic Journal, pp. 1–9, 2025.

[6] O. Sarjiyus, B. Y. Baha, and E. J. Garba,
"Enhanced security framework for internet banking services,"
Journal of Information Technology and Computing,
vol. 4, no. 2, pp. 45–54, 2022.

[7] R. Bhavsar, M. Dave, P. Shah, H. A. Joshiyara, and C. Patel,
"Enhancing data security in banking using hybrid cryptographic algorithms,"
Journal of Engineering Systems and Research,
vol. 8, no. 3, pp. 66–73, 2023.

[8] F. Jimmy,
"Cybersecurity threats and vulnerabilities in Java-based online banking systems,"
International Journal of Scientific Research and Management,
vol. 12, no. 3, pp. 312–318, 2024.

[9] S. Mehta,
"Secure coding practices for Java-based banking applications,"
International Journal of Financial Software Engineering,
vol. 5, no. 1, pp. 1–10, 2025.

[10] M. Mujinga,
"Usable authentication security in online banking applications,"
Information Security Journal, vol. 33, no. 1, pp. 22–35, 2024.

[11] R. Sharma and P. Verma,
"Design of a secure database-driven banking system using Java,"
International Journal of Computer Science and Information Security,
vol. 18, no. 4, pp. 55–62, 2020.

[12] S. Kumar and A. K. Singh,
"Desktop application development using Java Swing and JDBC,"
International Journal of Advanced Computer Science and Applications,
vol. 12, no. 7, pp. 410–416, 2021.

[13] A. Patel and N. Joshi,
"Secure transaction processing in Java desktop applications,"
Journal of Software Engineering and Applications,
vol. 15, no. 2, pp. 89–97, 2022.

[14] P. Choudhary and R. Meena,
"Implementation of a secure bank management system using Java Swing,"
International Journal of Innovative Technology and Exploring Engineering,
vol. 10, no. 8, pp. 201–206, 2021.

[15] S. Verma, A. Gupta, and R. Jain,
"A secure desktop-based banking application using Java and MySQL,"
International Journal of Emerging Technologies in Engineering Research,
vol. 9, no. 6, pp. 34–40, 2023.