International Institute of Information Technology, Bangalore

CS-513 System Software Mini Project

Title: Design and Development of Banking Management System

Submitted by: MT2025114 Shikhar Bhadreshkumar Mutta

Submission Date: 31st October 2025

PROJECT REPORT

----

1. INTRODUCTION

The Banking Management System (BMS) is a client-server application designed to simulate the operations of a banking institution.

It provides functionalities for customers, employees, managers, and administrators with secure and concurrent access to shared resources.

The project emphasizes the use of system-level programming concepts such as sockets, file management, and synchronization mechanisms.

----

2. OBJECTIVES

- Implement a secure and role-based banking system.
- Ensure concurrency and synchronization using semaphores and file locking.
- Maintain ACID properties across all financial transactions.

- Demonstrate modularity and use of inter-process communication.
- Provide a scalable file-based database structure.

----

3. SYSTEM ARCHITECTURE

The architecture follows a client-server model, where the server handles multiple client connections concurrently.
Each client represents a user role (Customer, Employee, Manager, Administrator).

Clients communicate with the server using TCP sockets.
The server performs operations like deposits, withdrawals, transfers, and loan processing while ensuring synchronization and security.

----

4. MODULE DESCRIPTION

4.1 CUSTOMER MODULE

- Login and authentication
- Deposit and withdraw funds
- Transfer funds between accounts
- Apply for loans and view transaction history
- Change password and logout

4.2 EMPLOYEE MODULE

- Login and authentication
- Add and modify customer details

- Process and approve/reject loans

- View customer transactions and loan requests

4.3 MANAGER MODULE

- Activate/deactivate accounts

- Assign loan applications to employees

- Review customer feedback

4.4 ADMINISTRATOR MODULE

- Manage employee and customer accounts

- Add or modify users and their roles

- System-wide monitoring and configuration

----

5. TECHNOLOGIES USED

- Programming Language: C

- Platform: Linux / UNIX

- System Calls: open(), read(), write(), lseek(), fcntl()

- Synchronization: File locks, semaphores

- Communication: Socket programming

- Build Tool: Makefile

----

6. IMPLEMENTATION DETAILS

- File-based storage for data persistence.

- Each user operation triggers read/write operations protected by locks.

- All I/O operations are performed using system calls, not standard I/O libraries.

- Multiple clients can operate concurrently without data corruption.

- Passwords are stored securely, and only one active session per user is permitted.

----

7. CONCURRENCY AND SYNCHRONIZATION

The project handles concurrent operations using file-level locking mechanisms.

When multiple clients perform transactions on the same account, locks prevent race conditions.

Semaphores ensure mutual exclusion during critical operations.

----

8. EVALUATION CRITERIA

- Working Code for all user modules.

- Proper synchronization ensuring ACID properties.

- UML Diagrams: Class, Component, and Sequence diagrams.

- Demonstration of concurrency and system-level programming.

----

9. UML DIAGRAMS (SUMMARY)

- Class Diagram: Represents entities and their attributes (Customer, Employee, Manager, Administrator).

- Component Diagram: Shows interactions between clients and the server.

- Sequence Diagram: Depicts transaction processes such as fund transfer.

----

10. TESTING AND VALIDATION

- Unit testing for each module (Customer, Employee, Manager, Administrator).

- Concurrent client testing to ensure data integrity.

- Validation of all functionalities per requirements.

- Verified correctness of transactions and synchronization mechanisms.

----

11. RESULTS AND DISCUSSION

The Banking Management System successfully simulates a realistic banking environment.

It demonstrates concurrency, synchronization, and role-based access control.

All transactions satisfy ACID properties and operate without race conditions.

----

12. FUTURE ENHANCEMENTS

- Integrate a relational database (MySQL/PostgreSQL).

- Develop a web-based GUI for better user experience.

- Implement encryption for all stored data.

- Add automated report generation and analytics dashboard.

----

13. CONCLUSION

The project fulfills the core objectives of the course demonstrating system programming concepts in a realistic application.

It effectively shows synchronization, concurrent file access, and secure role-based operations in a multi-user environment.

----

14. REFERENCES

- Linux Man Pages (System Calls Documentation)

- TCP/IP Socket Programming Guide

- IIIT Bangalore Course Material on System Software