BANKING SYSTEM SIMULATOR USING C++ DATA STRUCTURES

Presented by Group I

GROUP 1: MEMBERS

Bautista, Mariela

Cabrera, Gabriel

Pizarro, Jeus

Quioyo, Angelo

INTRODUCTION

In the financial services industry, particularly within retail banking, there is a significant gap between theoretical knowledge and practical, real world application. New employees, such as bank tellers and branch managers, are required to master a wide array of complex procedures, adhere to strict security policies, and operate proprietary software systems where a single mistake can lead to financial loss and reputational damage. Traditional training methods, which often rely on passive observation, mentorship, and manual-based learning, can be inefficient and fail to provide a safe, repeatable environment for trainees to build confidence and competence, especially when handling irregular situations or critical errors.

CHALLENGES

TRAINING

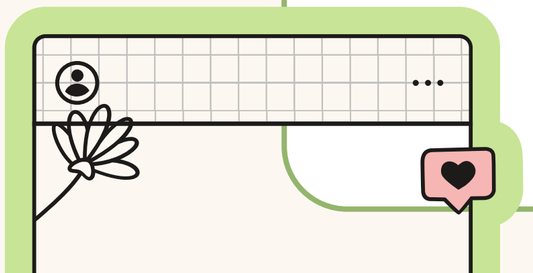
New tellers and managers need a sandboxed environment to practice core transactions, handle errors, and learn auditing procedures without risking real money.

PROTOTYPING

Developing new banking software requires a stable, functional backend to test against, which is often complex and time-consuming to build.

DATA INTEGRITY

Financial systems must be resilient. A system crash should never result in lost transactions or corrupted account data.



SOLUTION

This project is a console application that simulates the core functions of a retail bank, built with a focus on data persistence, auditing, and security.

It provides a complete, self-contained ecosystem where every transaction is processed, validated, and recorded accurately.
The system serves two primary purposes:

1. A training tool for practicing banking operations.
2. A functional prototype demonstrating the architecture of a robust financial application.

GENERAL OBJECTIVE

The overall objective of this project is to create and execute a working Bank Account Management System that operates through the console, ensuring the system is very safe in handling client accounts and very accurate in keeping comprehensive records of all financial transactions.

SPECIFIC OBJECTIVES

1. Provide users with the ability to execute the principal activities like depositing, withdrawing, and inquiring about the balance.
2. Establish a secure administrative module for system oversight and account management.
3. Make sure that the system will be capable of loading and saving account data through files persistently in order to keep state across sessions.

KEY FEATURES

CUSTOMER MODE

Authenticated users can deposit, withdraw, transfer funds, and view their balance and statement.

ADMIN MODE

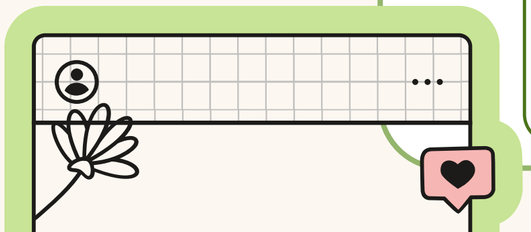
A password-protected area allows for account creation and deactivation, with a two-step confirmation process for critical actions.

DATA PERSISTENCE

All transactions and account changes are instantly saved to disk, ensuring no data is lost on an unexpected shutdown.

SYSTEM STATUS DASHBOARD

A real-time view of the system's operational status, including the currently served customer and total transactions processed.



DATA STRUCTURES USED

DATA STRUCTURE	PURPOSE	WHY IT WAS CHOSEN
VECTOR	Bank class to store all customer accounts.	Flexibility, Efficiency, Memory Management
QUEUE	To manage customers waiting to be served. It stores the account numbers of logged-in users.	First-in, first out principle
LINKED LIST	Implementation of stack	Dynamic data structure to store transactions
STACK	Each Account object has its own TransactionStack to maintain a history of all its transactions.	Last-in, first out principle to show transaction history chronologically

OTHER CONCEPTS USED

CLASSES & OBJECTS

The project is organized into logical units like Bank, Account, and TransactionStack. Each class encapsulates its own data and behavior.

ENCAPSULATION

Private members (like Account::balance and Account::pin) protect data from direct, uncontrolled access. Public methods provide a safe and controlled interface.

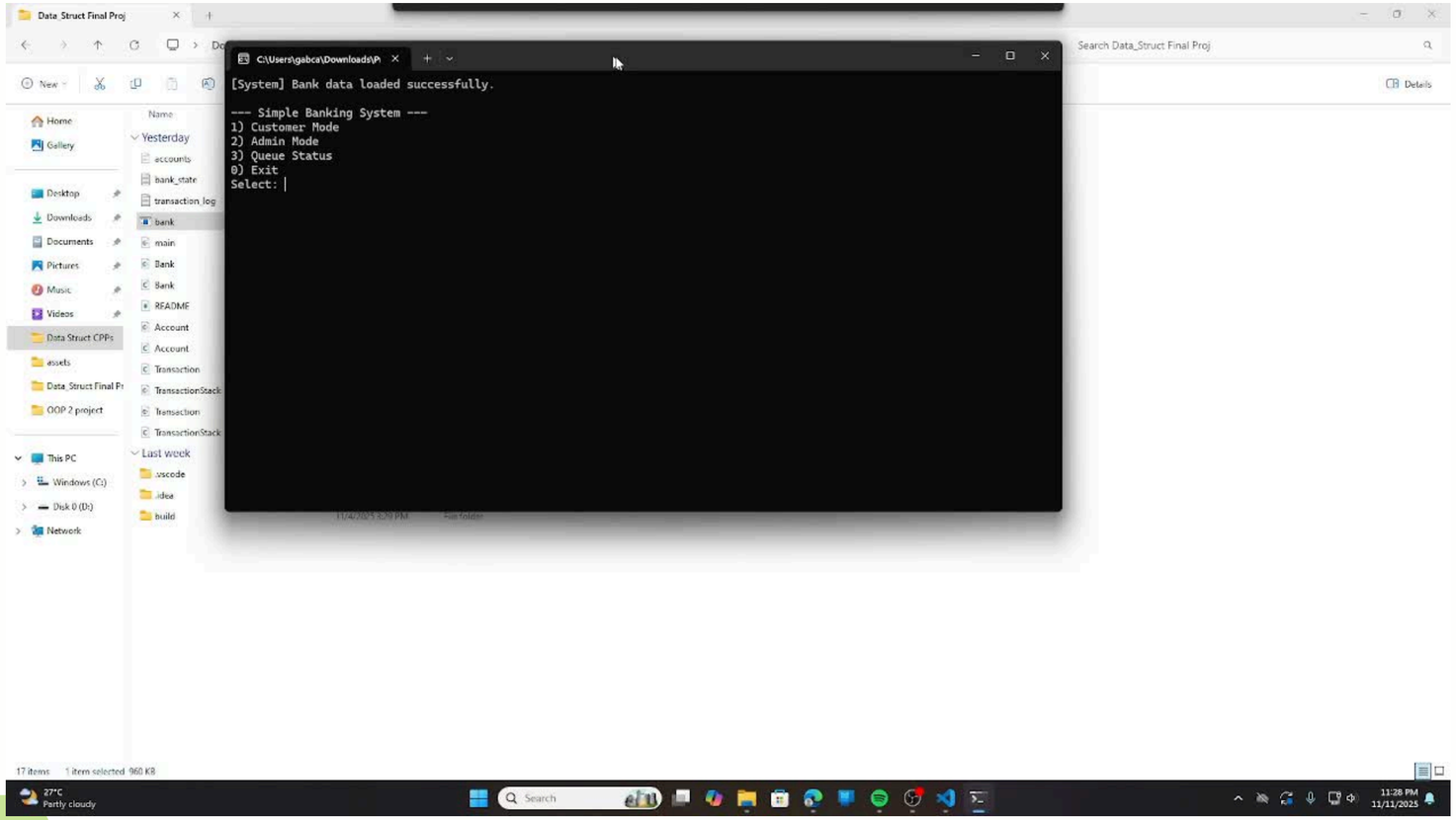
ABSTRACTION

Users of the Bank Simulator don't need to know how data is saved to a file or how accounts are stored in memory.

FILE I/O

The application uses streams for data persistence and processing.

PROJECT DEMO



CONCLUSION

The project successfully demonstrated the design and implementation of an efficient and persistent banking system simulation. By leveraging fundamental data structures and modern software engineering principles, the application effectively models core banking operations, including account management, financial transactions, and administrative oversight. The primary objective, which is to create a functional system with a strong emphasis on data integrity and persistence, was successfully achieved.

THANK YOU