**Bank Management System - Project Report**

**BY:**

**Hasnain Wazir (2024222)**

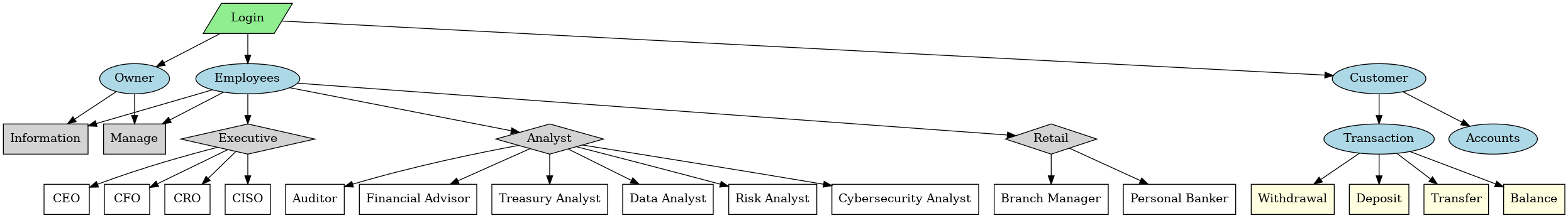
**Abdullah Munir (2024042)**

# 1. Introduction and Problem Statement

The Bank Management System (BMS) is a console-based software designed to streamline and manage banking operations using Object-Oriented Programming (OOP) in C++. The main aim of this project is to demonstrate how core banking functions like customer registration, account management, transaction processing, and employee role-based operations can be implemented efficiently using OOP principles. This system supports multiple roles like customers, managers, and higher-tier employees like CEO, CFO, CRO, and CISO, each with specific access and control features.

# 2. System Design and Architecture

The architecture of the BMS consists of multiple classes representing different roles and functionalities:  
- `User` (base class)  
- `Customer`, `Employee` (derived from User)

- `Manager`(derived from Employee)  
- Separate functions for each employee role (CEO, CFO, CRO, CISO)  
- File-based storage for persistence (e.g., users.txt, transactions.txt)  
  
UML Diagram:  


This is the diagram we created initially while thinking the idea to execute.

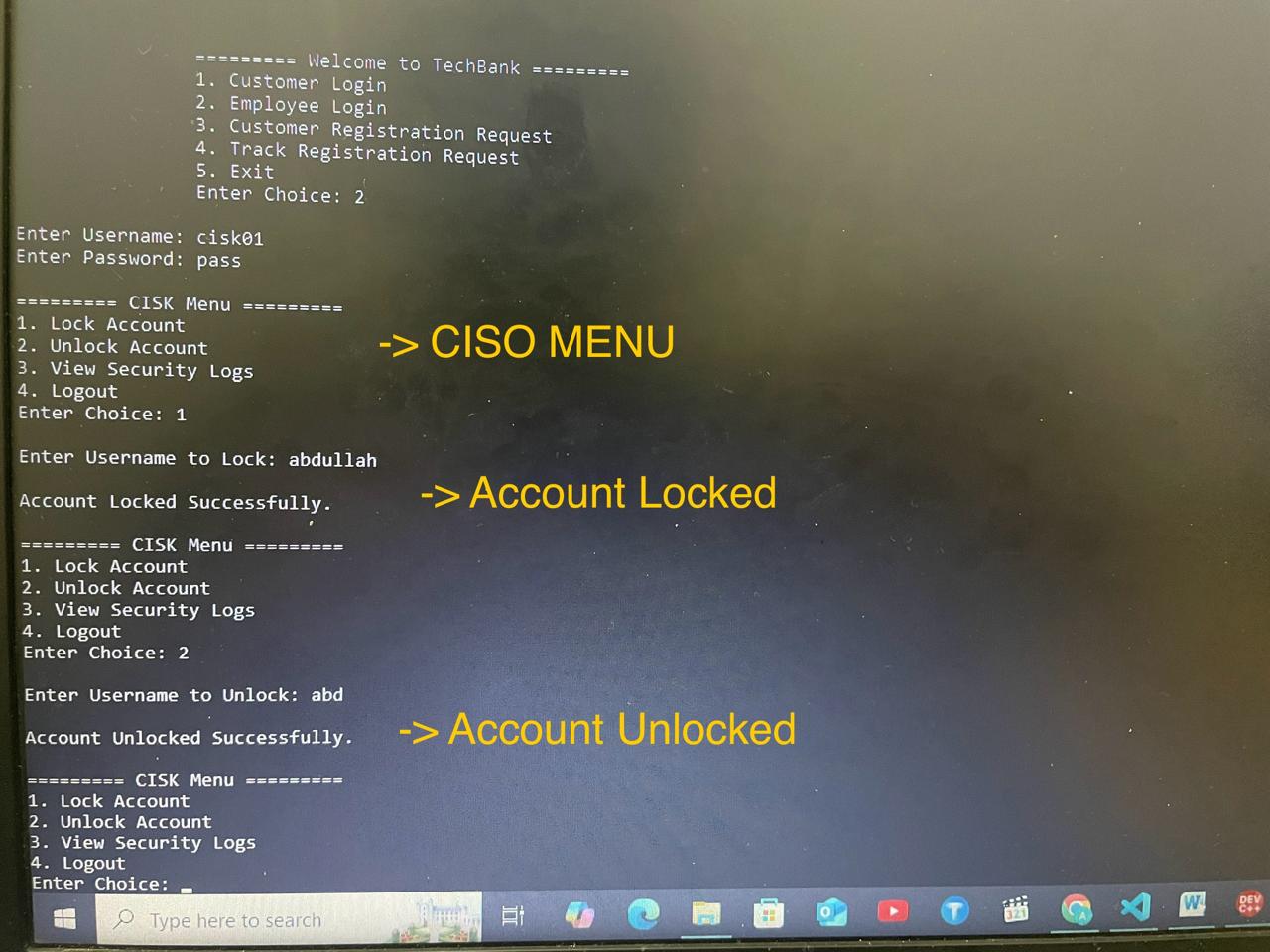
# 3. Application of OOP Concepts in C++

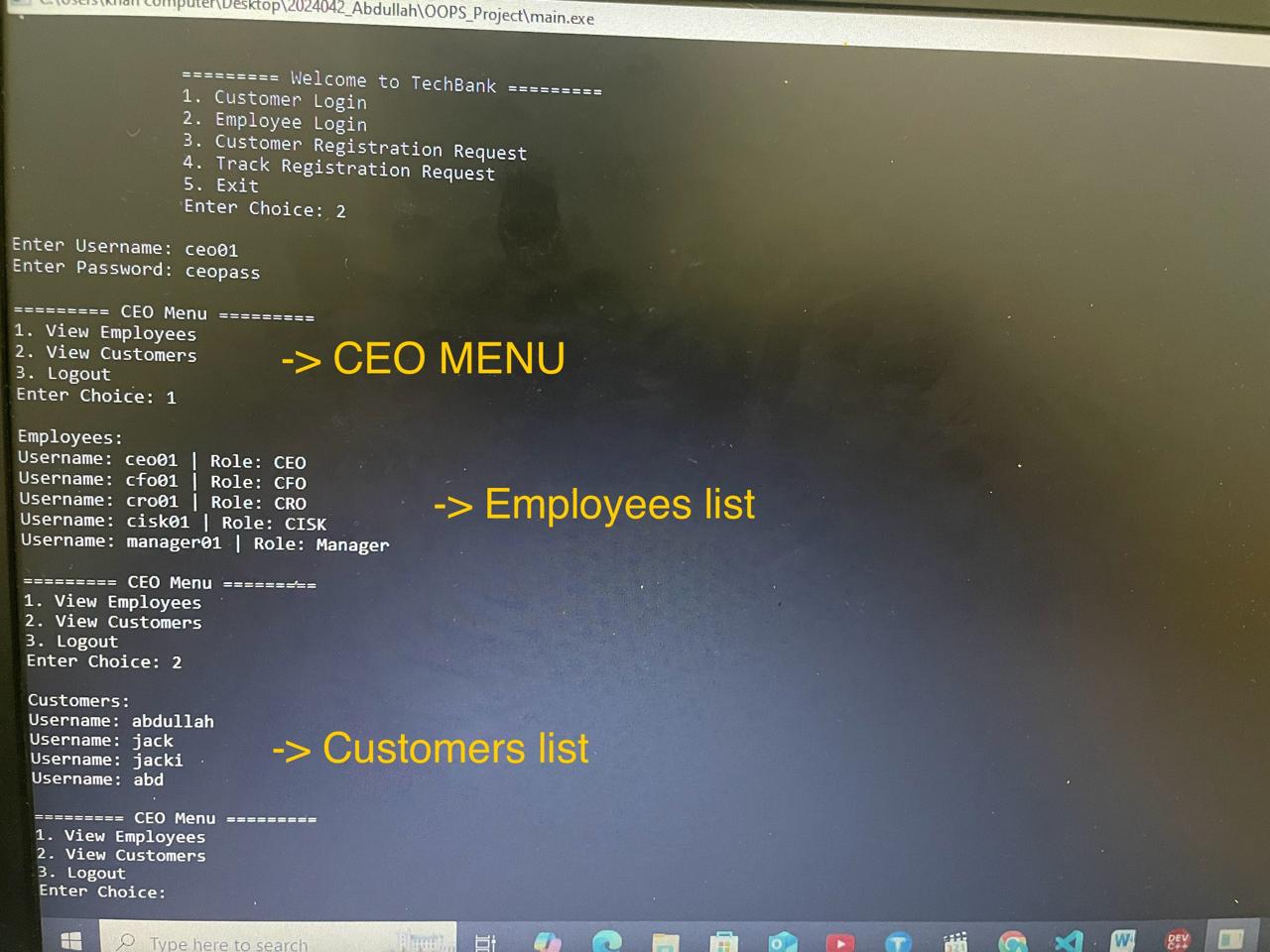
The BMS project makes use of several key OOP concepts:  
- \*\*Encapsulation\*\*: Data and functions are encapsulated within classes like `Customer`, `Manager`, etc.  
- \*\*Inheritance\*\*: Classes like `Customer` and `Employee` inherit from the base class `User`. And then again another class `Manager` is inherited from class `Employee`.  
- \*\*Polymorphism\*\*: While not heavily used, role-based behavior switching simulates polymorphism.  
- \*\*Abstraction\*\*: Role-based functions abstract away implementation details from the user interface.

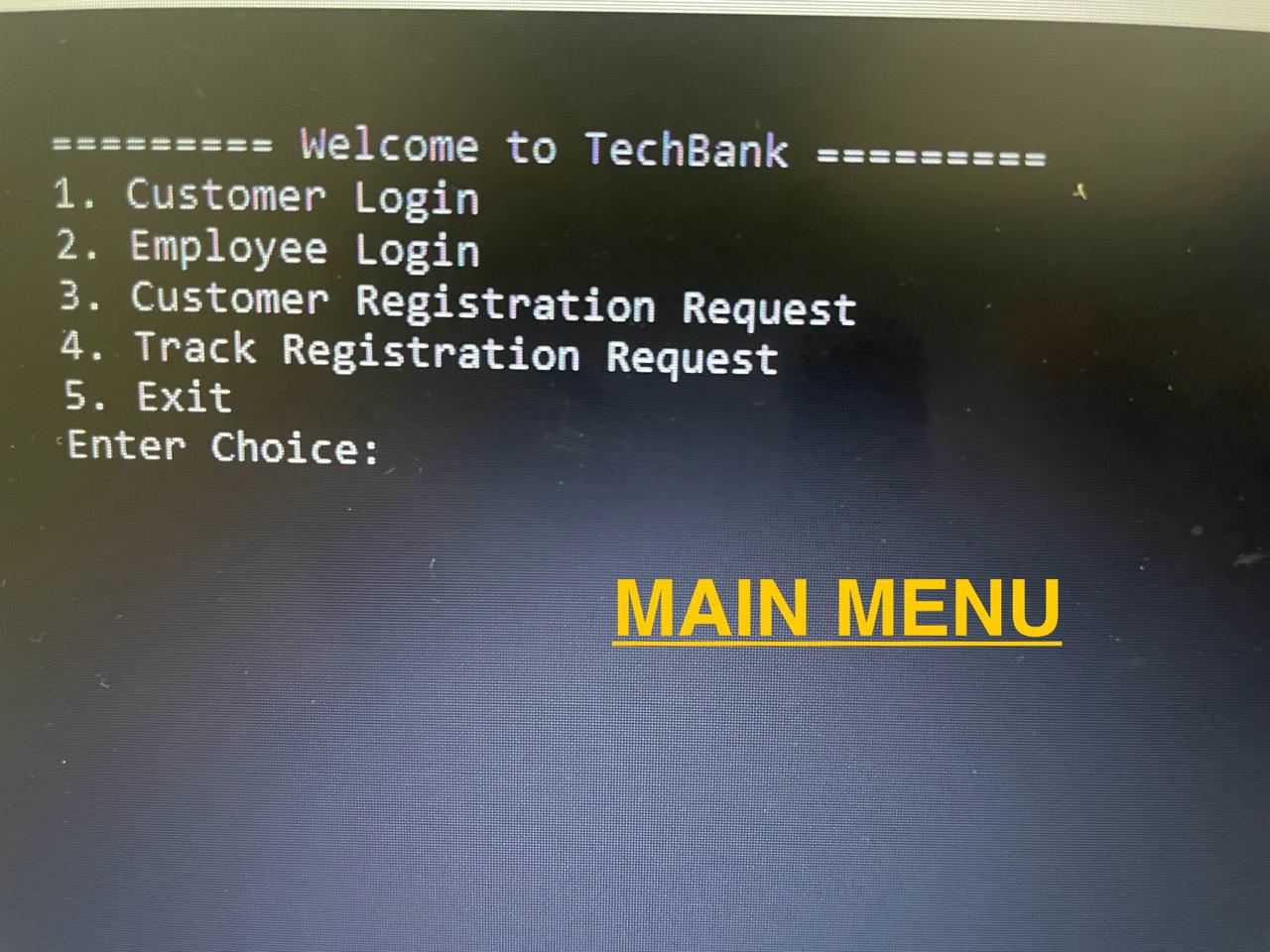
# 4. Implementation Details and Challenges

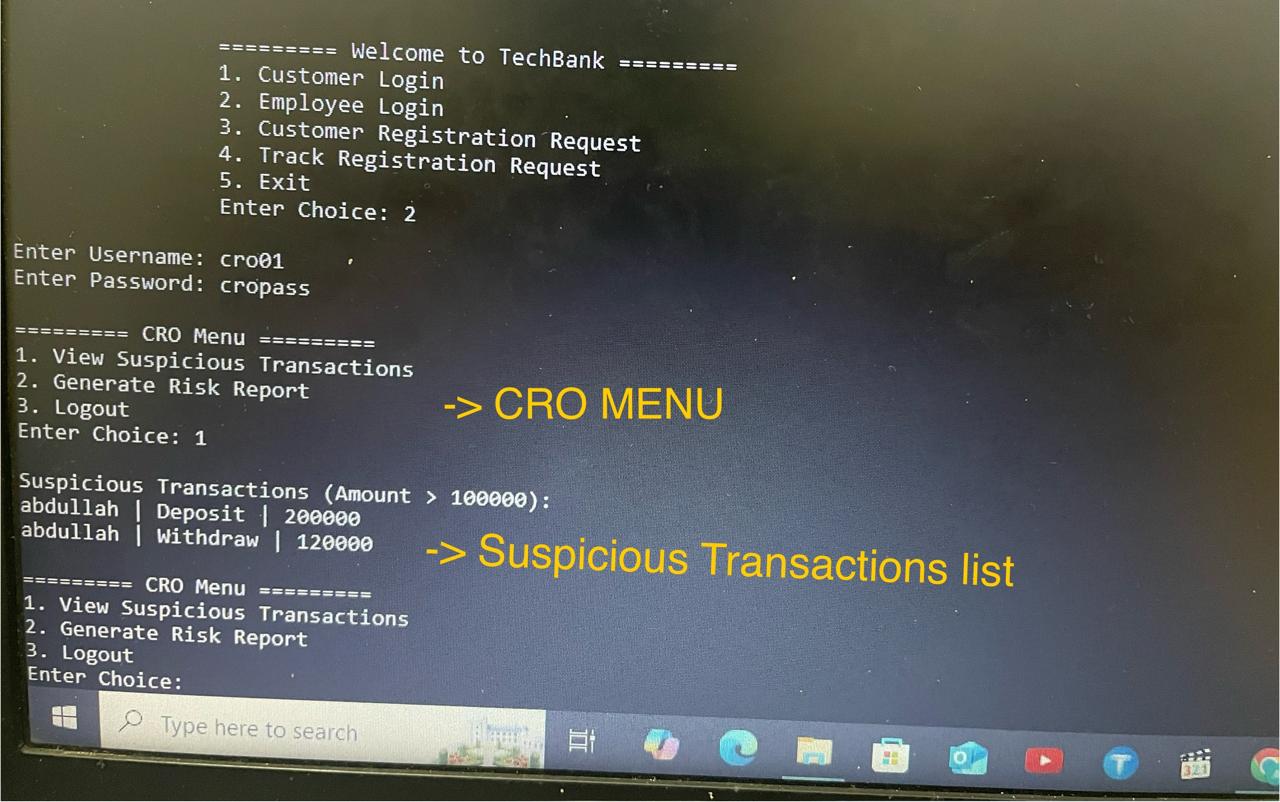
The system was implemented in C++ using standard input/output streams and file handling for data persistence.  
Key challenges included:  
- Managing user sessions securely using simple file encryption.  
- Ensuring correct role-based access to sensitive functions (e.g., loan approval, account lock).  
- Tracking and managing registration requests and transaction records.  
- Maintaining data integrity across multiple text files without using databases.

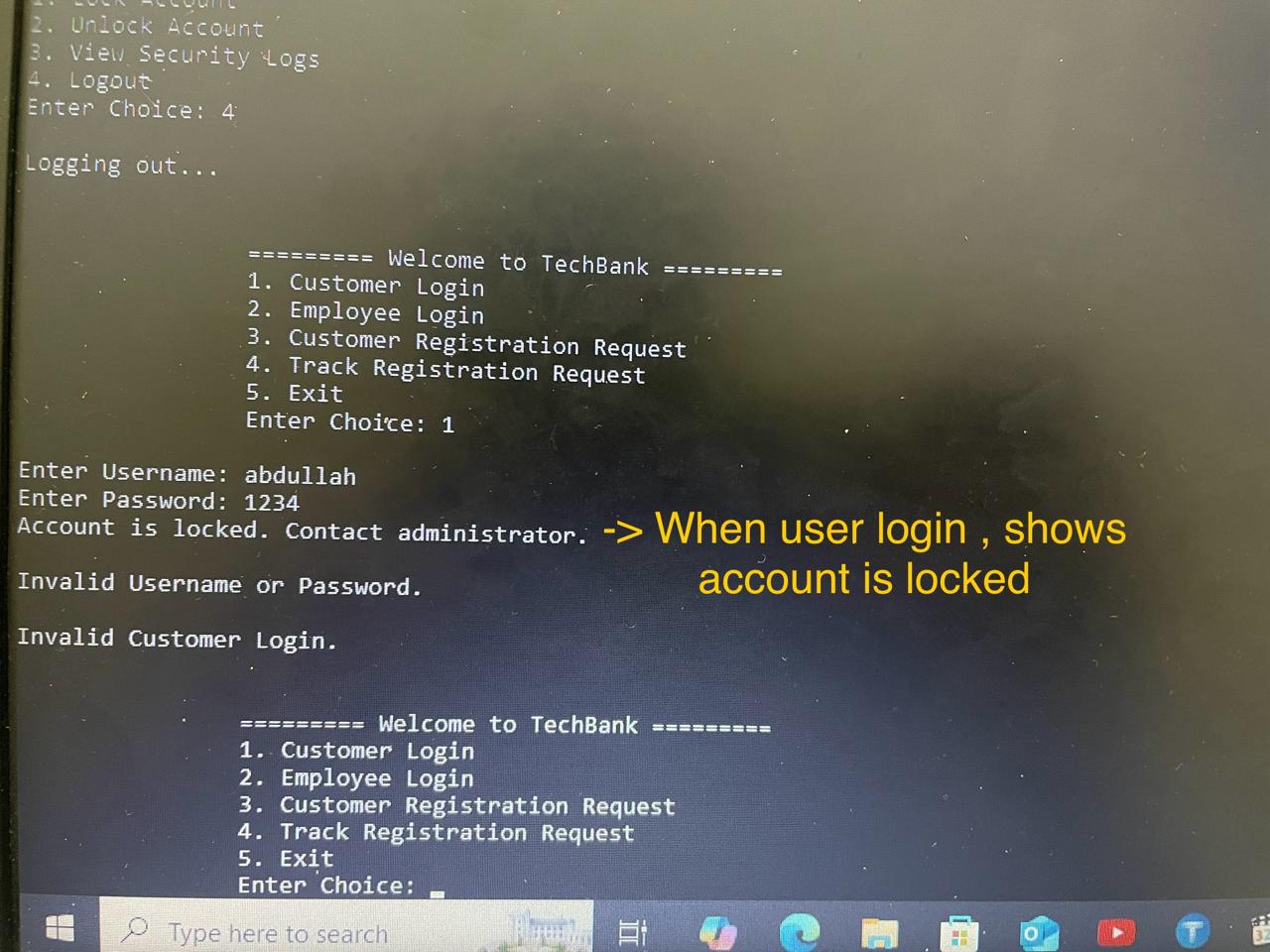
# 5. Screenshots or Outputs of the System

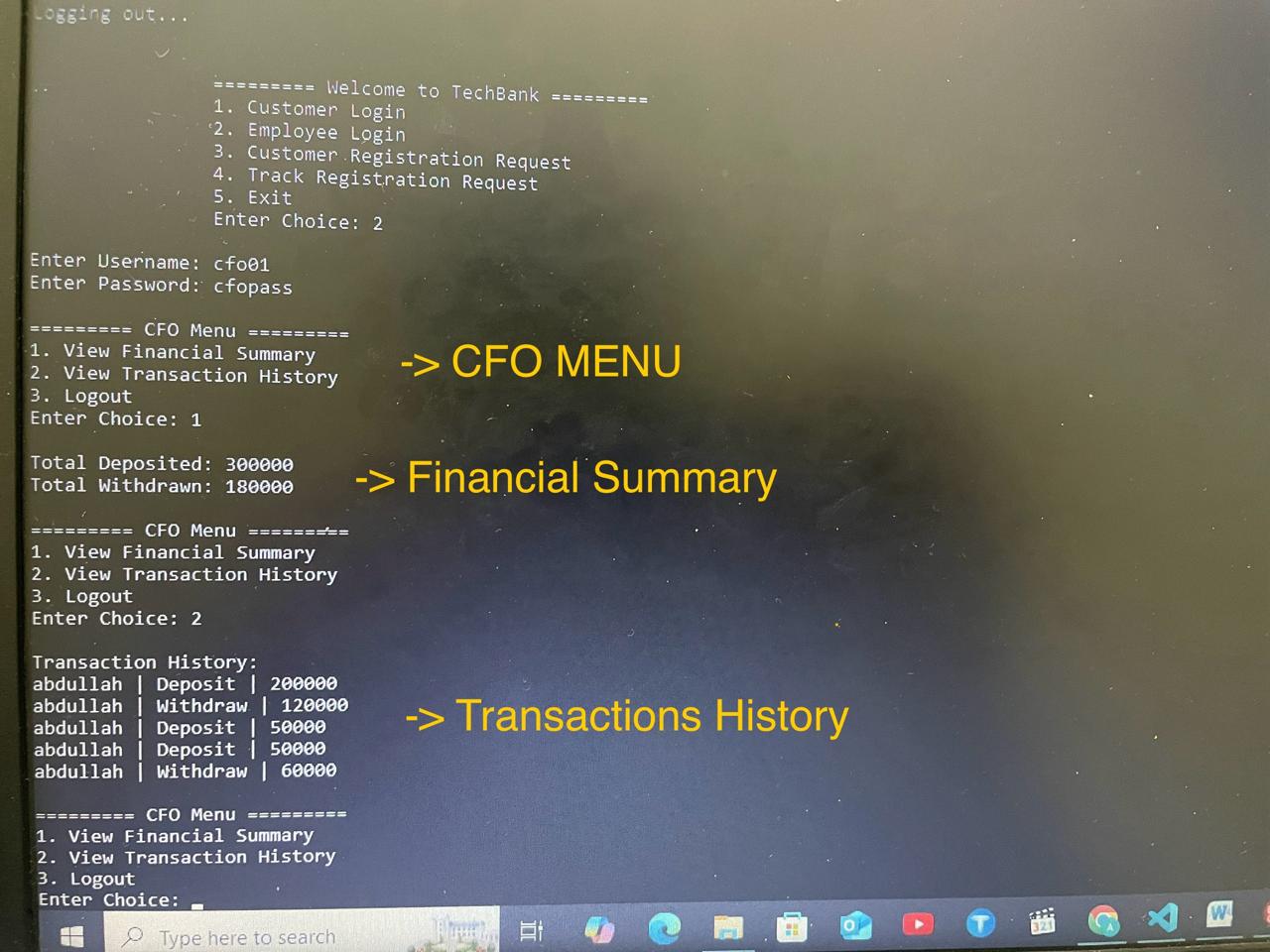


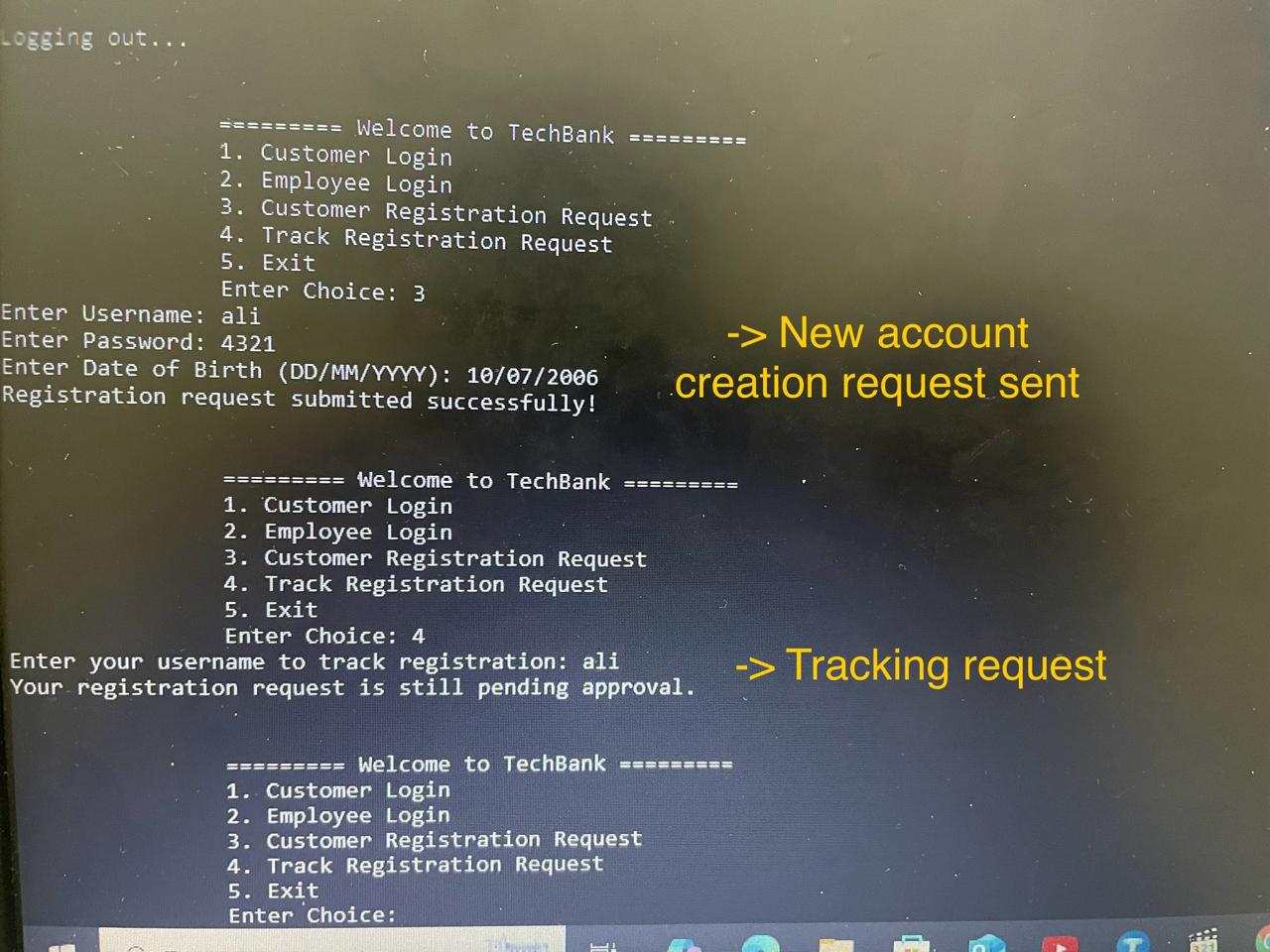


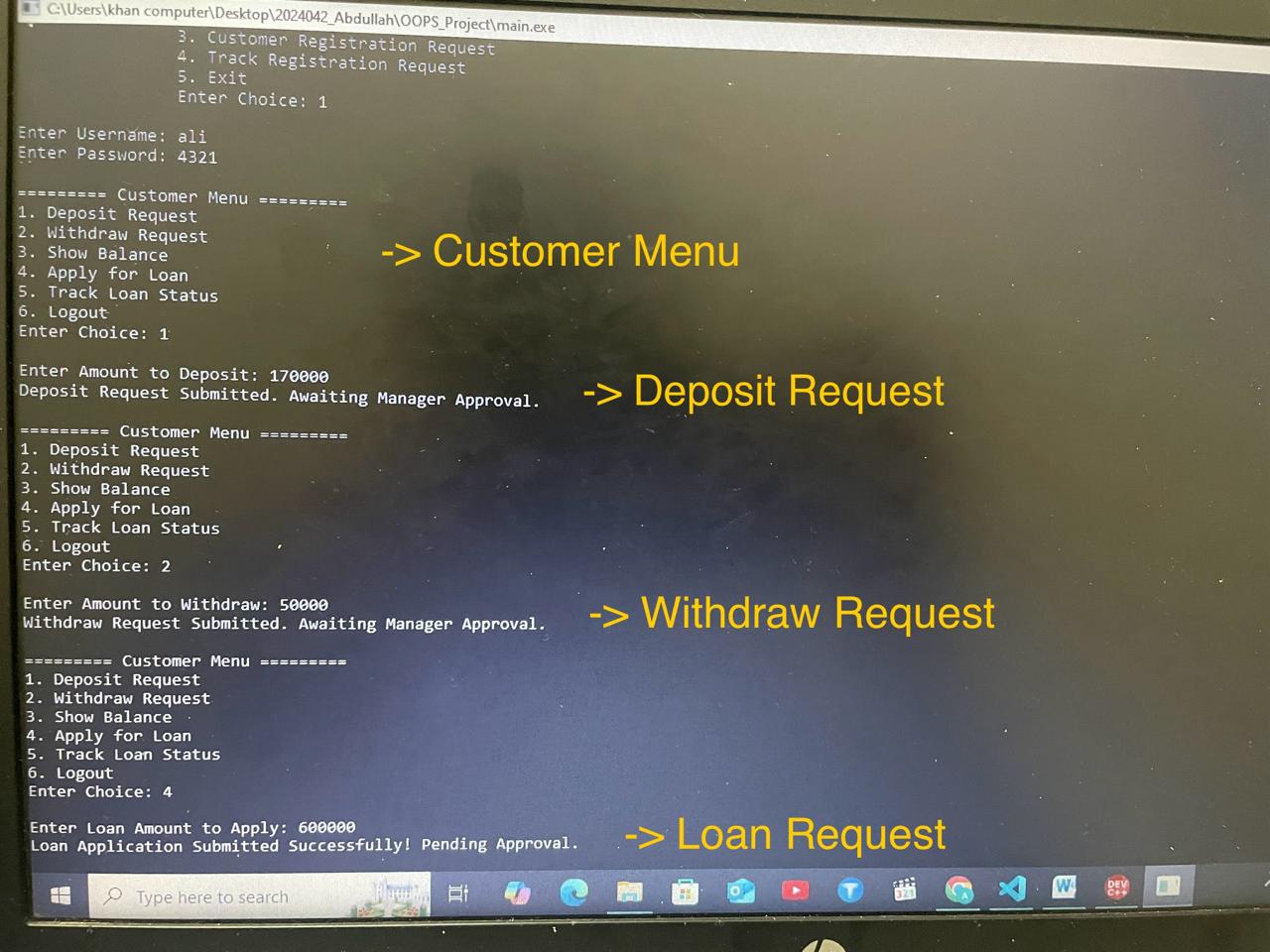


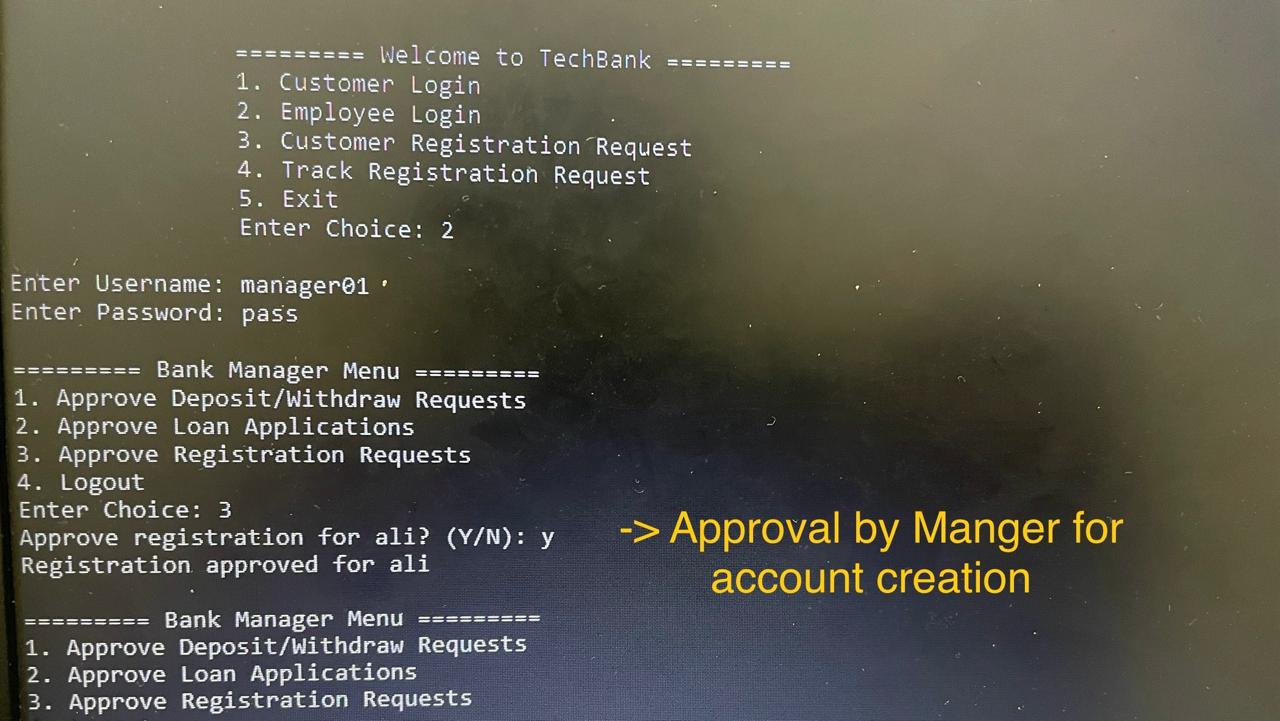


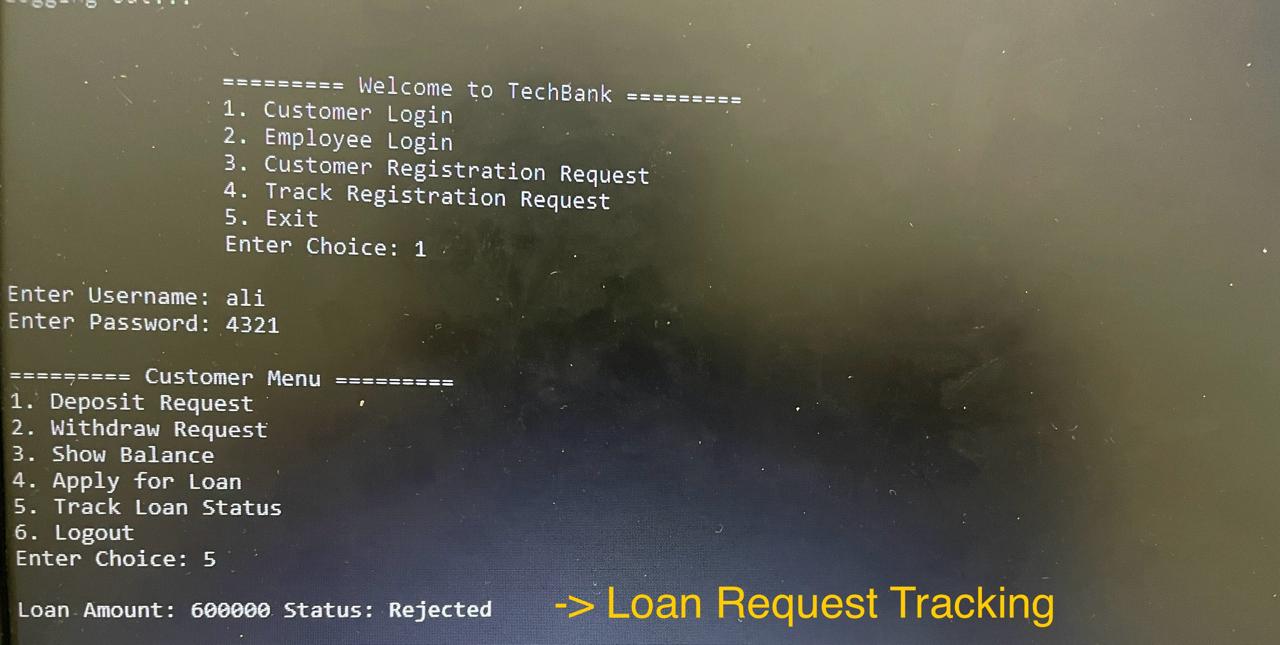


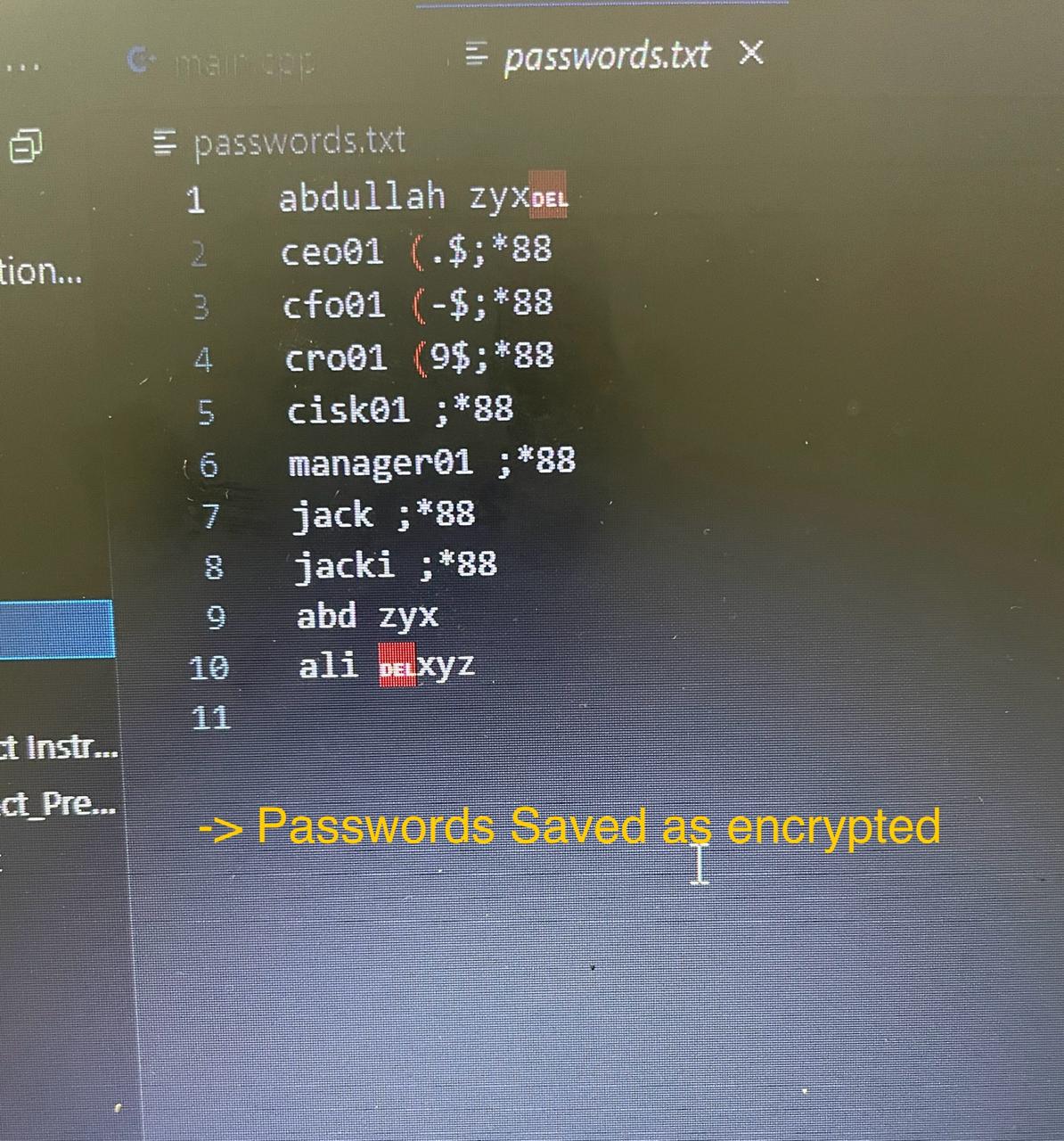


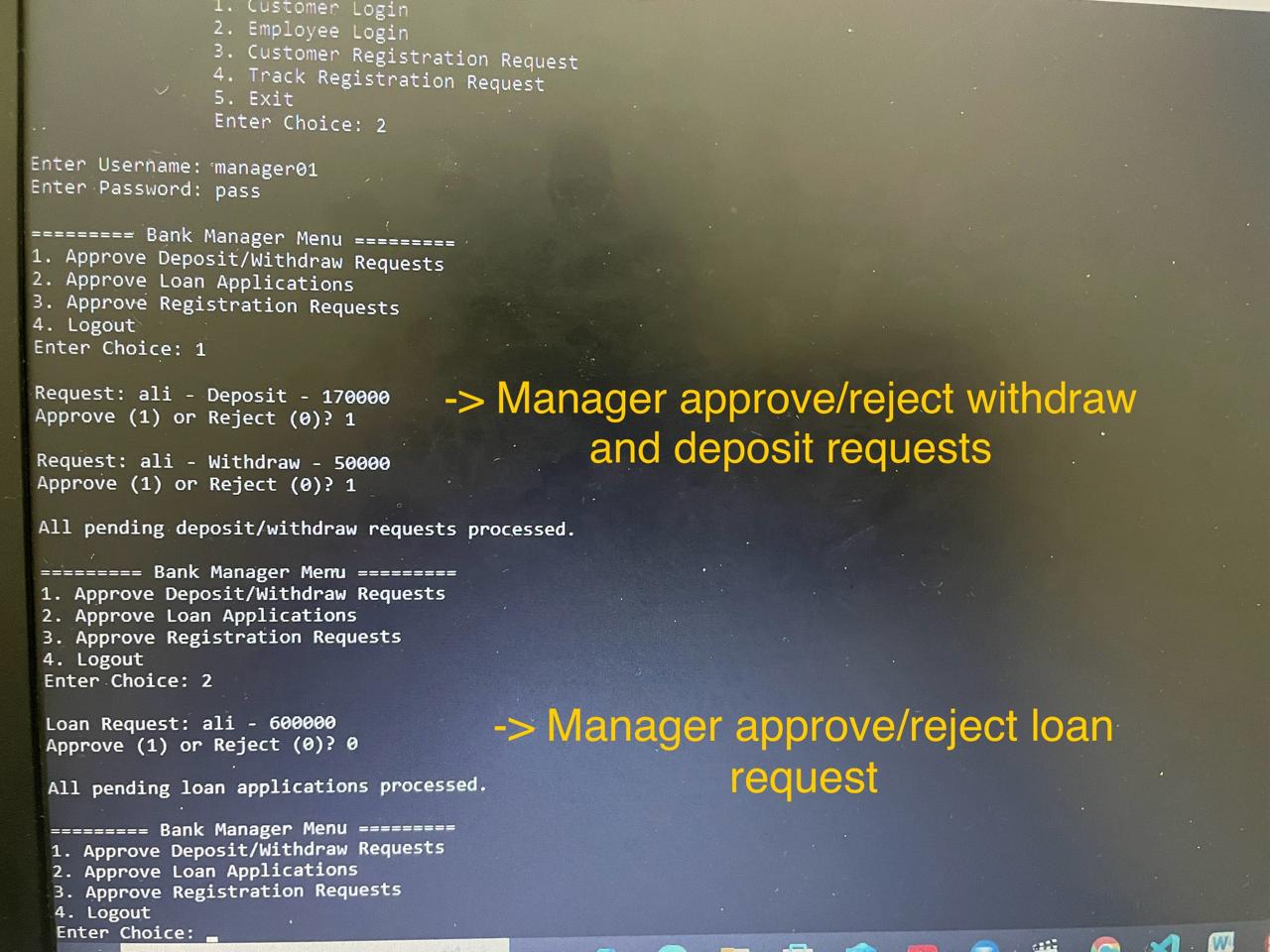












# 6. Conclusion and Future Improvements

The Bank Management System successfully demonstrates the implementation of a functional banking backend system using OOP concepts. It effectively handles customer and employee workflows and ensures modular code structure.  
Future improvements may include:  
- Migrating to a database system for better scalability and data integrity.  
- Adding a GUI or web-based frontend.  
- Implementing encryption using secure libraries (like hashing etc).  
- Adding automated test cases and error logging.