**Problem Statement:**

**Title**: Object-Oriented Implementation of a Banking System

**Overview**:  
A banking system needs to be modeled using Object-Oriented Programming (OOP) principles. The system should include basic functionalities such as creating accounts, depositing funds, withdrawing funds, checking balances, and transferring money between accounts. Additionally, the system should distinguish between different account types (e.g., Savings and Current accounts) and enforce specific rules for each type.

**Requirements**:

1. **Entities**:
   * A Bank entity to manage multiple accounts and transactions.
   * An Account superclass with common attributes and methods shared by all account types.
   * Specialized account types:
     + SavingsAccount: Includes a minimum balance requirement.
     + CurrentAccount: No minimum balance but allows for overdraft up to a limit.
2. **Functionalities**:
   * **Account Management**:
     + Create a new account with a unique account number.
     + View account details.
   * **Transactions**:
     + Deposit money into an account.
     + Withdraw money with constraints:
       - Savings accounts cannot go below the minimum balance.
       - Current accounts allow overdrafts up to a specific limit.
     + Transfer funds between accounts.
   * **Bank Operations**:
     + View a summary of all accounts.
     + Display transaction histories for individual accounts.
3. **Design Constraints**:
   * Use appropriate OOP concepts:
     + Classes and objects to represent entities.
     + Inheritance to differentiate between account types.
     + Polymorphism for shared and overridden methods (e.g., withdrawal rules).
     + Encapsulation for secure access to account attributes.
   * Ensure data consistency and error handling for invalid operations (e.g., insufficient funds, negative deposits).

**Implementation Goals**:

The system will be implemented in Python. The solution should include a class diagram representing the relationships between entities, followed by a Python implementation of the design. The implementation must be modular and user-friendly, allowing future enhancements such as integrating with a GUI or a database.

Class Diagram:

