# COMS-280 Final Project

## Implementation Plan

**Banking System in C++:**

This program demonstrates an advanced banking system using Object-Oriented Programming (OOP)

principles such as inheritance, polymorphism, templates, exception handling, and linked lists.

* Abstract base class for generic banking accounts.
* Derived classes for savings and checking accounts.
* Exception handling for overdrafts and insufficient funds.
* Transaction management using a linked list.
* Generic template-based account manager for handling different account types.
* Usage of smart pointers (`unique\_ptr`) for efficient memory management.

**Transaction History:**

* Transactions will be stored using a linked list (`std::list<Transaction>`).
* The `TransactionHistory` class will manage transactions, allowing new ones to be added and retrieved.
* Each transaction will have a type (e.g., "Deposit", "Withdrawal") and an amount.
* A function `displayHistory()` will allow users to view past transactions.

**Account Management:**

* Accounts will be created through an `AccountManager` template class.
* Users can open `SavingsAccount` or `CheckingAccount`.
* Each account type will support deposits, withdrawals, and balance checks.
* Accounts will be stored using `std::vector<unique\_ptr<BankAccount>>` to enable dynamic memory allocation**.**

**User Authentication:**

* Basic authentication can be added using a `User` class that stores usernames and passwords.
* Authentication will require users to provide correct credentials before accessing their accounts.
* Future implementations could integrate encryption for security.

**Operator Overloading:**

* The `+=` operator will be overloaded for deposits, allowing `account += amount` syntax.
* The `-=` operator will be overloaded for withdrawals, allowing `account -= amount` syntax.
* The `<<` operator will be overloaded for displaying account details in an easy-to-read format.

**Abstract Base Class: BankAccount:**

* Defines a blueprint for various types of bank accounts.
* Implements core properties such as owner name and balance.
* Provides pure virtual functions for deposit, withdrawal, and displaying account details.