# COMS-280 Final Project

## Reflection #2

The biggest challenge that I have had with phase 2 is the unit testing and how to include it into my code. I have tried several ways to incorporate it into my code but I have yet to figure it out. It does seem that we are doing nearly the same thing that we did last week. It just seems like it is adding a lot of work to the project without any real-world value. I can see application of phase 1, but in phase 2 could have been cut in half, not everything in phase 2 is really necessary. I feel that steps 1 and 2 was covered in phase 1. The implementation of inheritance and polymorphism in this project aligns well with the core objectives of creating a flexible, extensible, and maintainable banking system. The inheritance hierarchy ensures that both SavingsAccount and CheckingAccount share common functionalities from the BankAccount base class while still allowing specialized behaviors. This promotes code reusability by eliminating redundancy, as shared properties like owner, balance, and transaction history management are handled in the base class, ensuring consistency across all account types. Polymorphism plays a crucial role in enabling dynamic behavior through the use of virtual functions, allowing deposit and withdrawal operations to be performed correctly depending on the account type. The use of a common interface for all accounts allows the AccountManager to store and manage different account types seamlessly without knowing their specific implementation details. Additionally, runtime polymorphism ensures that method calls, such as display(), execute the correct version based on the actual object type, providing a more intuitive user experience. The system's extensibility is further enhanced by allowing future additions of new account types without modifying the existing code structure. Overall, the strategic use of inheritance and polymorphism makes the system more scalable, modular, and adaptable, fulfilling the project's goal of a robust and efficient banking application.