Department of Electrical,
Computer, & Biomedical Engineering
Faculty of Engineering
& Architectural Science

Course Title:	Object Oriented Eng Analysis
Course Number:	COE528
Semester/Year (e.g.F2016)	W2024
Instructor:	Boujemaa Guermazi
Assignment/Lab Number:	Project/Lab 5
Assignment/Lab Title:	Bank
Submission Date:	03/24/2024
Due Date:	03/24/2024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Su	Jason	501158090	1	J.S.

Use Case Diagram

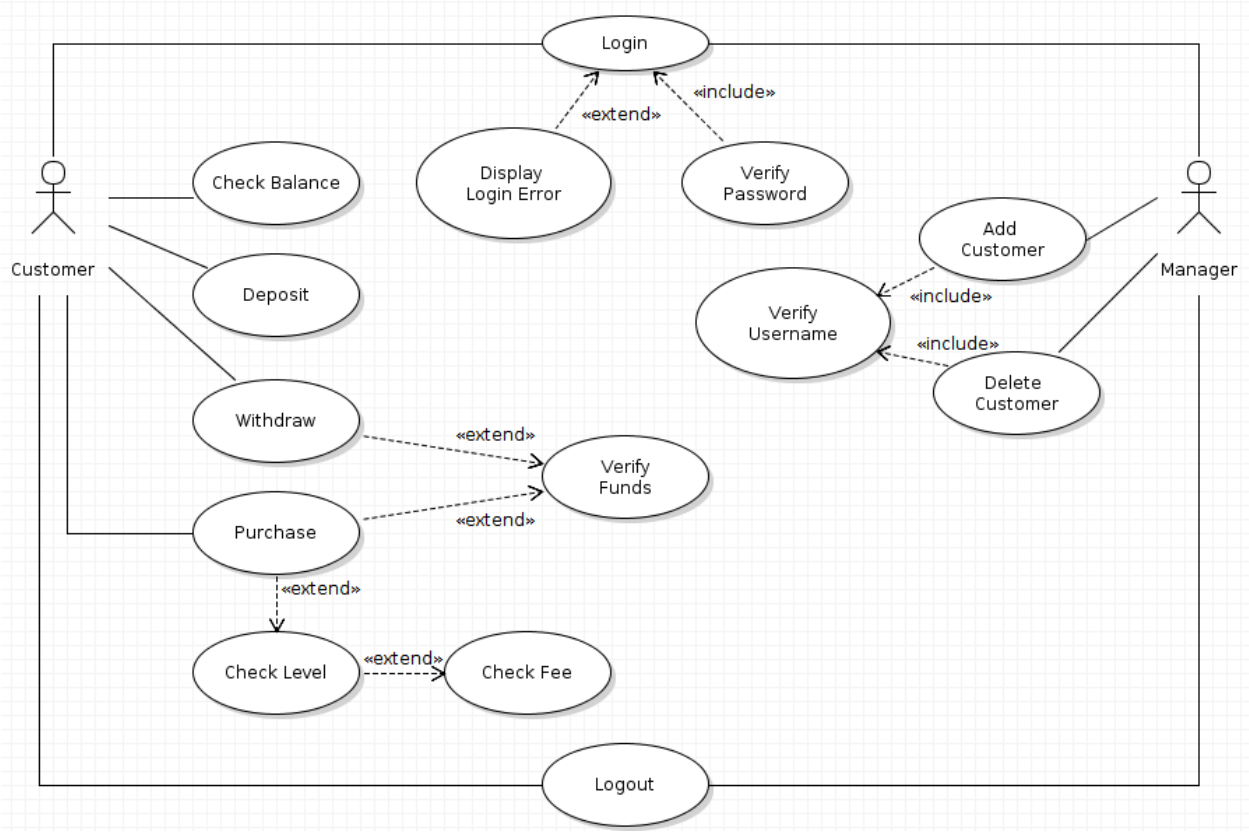

Figure 1

The use case diagram depicted in **Figure 1** shows the use case relationships for the Bank application. The system consists of two actors; Customer and Manager. Managers have access to Add and Delete Customer functions which include a username verify case to check if the specified username file exists. Customers have the ability to check balance, deposit, withdraw, and purchase. In order to perform a withdrawal or purchase transaction, the customer's funds must first be verified. A purchase must also check the Customer's account level to get the purchase fee. Both Customer and Manager have access to login and logout. Logins require a password verification and will display a login error message if invalid.

Figure 1 A use case of the bank system and its relations.

Use Case Name: withdraw
Participating Actor: Customer
Flow of Events: Customer attempts to perform a withdrawal. Account funds are then verified. If sufficient funds are available, withdrawal is performed.
Entry Condition: Customer attempts to withdraw funds.
Exit Condition: Withdrawal is completed or insufficient funds are detected.

Quality Requirements: When a Customer attempts to withdraw more money than that in their account, withdrawal will fail and display an error message.

Class Diagram

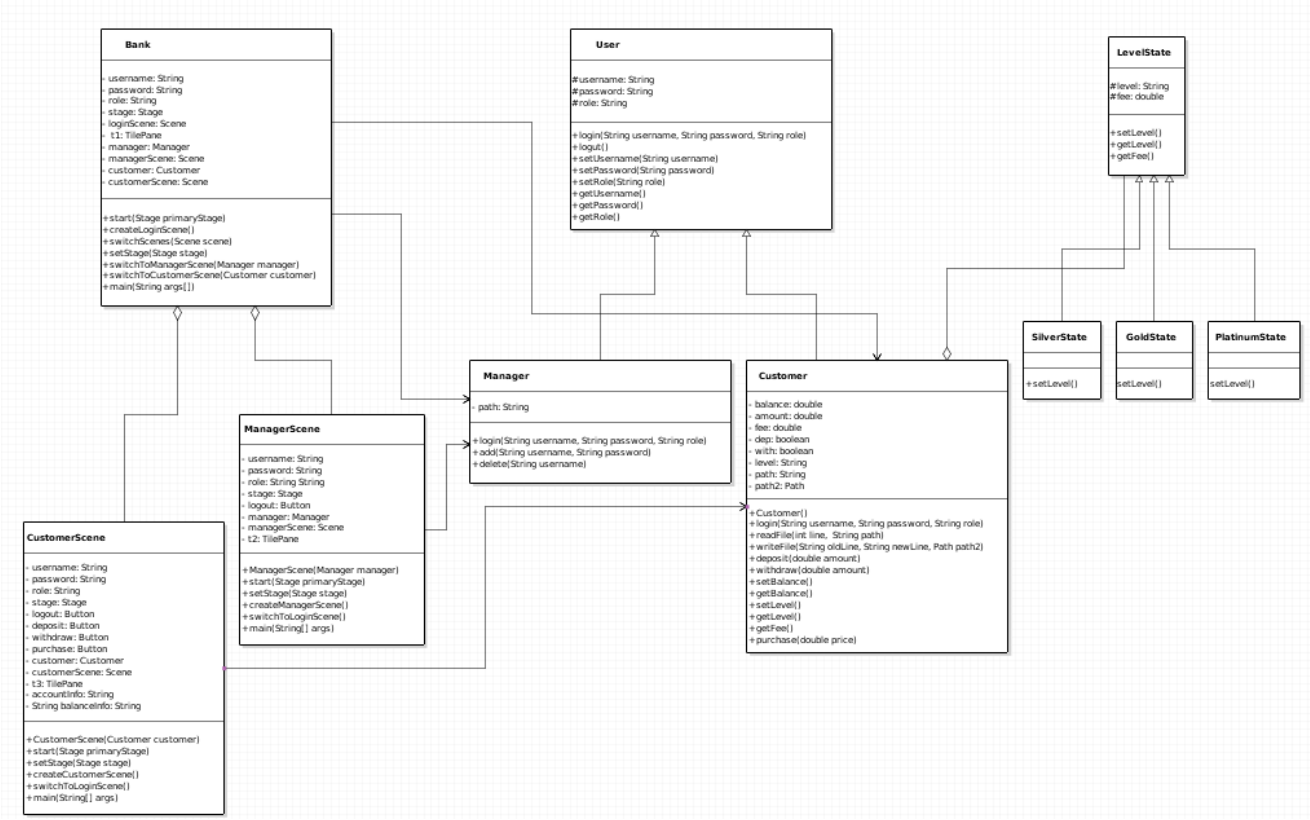

Figure 2

The class diagram depicted in **Figure 2** consists of three JavaFX classes, two abstract classes, and five regular classes. The JavaFX classes each represent a different scene in the GUI; Bank, CustomerScene, and ManagerScene represent the Login, Customer page, and Manager page respectively. Bank is associated with both Manager and Customer classes because objects of Manager and Customer must be created to display their respective information. The abstract class User represents the blueprint for Manager and Customer as it includes shared functions such as Login, Logout, and getter/setter methods. The User class is also the one used to address point 2, with an overview clause, abstraction function, and rep invariant included. The Manger Class contains add and delete functions and overrides the login method. The Customer class has methods to complete transactions, read and write to files, and check account level. The abstract class levelState represents the structure for its subclasses SilverState, GoldState, and PlatinumState. These classes form the State design pattern by changing state depending on Customer balance. When the balance is under 10000, the state will be set to silver. When balance is increased to value between 20000 and 10000, the state will change to gold. When balance exceeds 20000, the state will then be changed to platinum.