SE Experiment-3

**(Batch-A/A1)**

# -Team Details-

Shanouf Ansari – UID (2021300004) Allen Andrew – UID (2021300006)

T.E. Computer Engineering – A

# Aim:

To draw Class diagram for a banking software.

# Implementation:

Mentioned below is the chronological order that was followed to design a Class diagram for our banking software. Initially, problem description was re-visited to identify nouns and noun phrases. These were then reviewed to get rid of any ambiguity and classes were identified. Relations between different classes and how they interact with each other were noted to get an idea about the overall structure of the class diagram. Finally, the class diagram was constructed keeping the above- mentioned points in mind.

Problem Description-

A banking software needs to be designed for a particular **bank**. This software should be capable of authorising different types of **users** and allowing them to perform their respective **functions**. A **guest user** must be able to view the software through an **interface**. He must also be able to get a glimpse of **loans** provided by the bank such as **home loans**, **education loans** and **vehicle loans**. A guest user must be able to register himself. **Registered users** must be able to log into their **accounts** and create **bank accounts**. A **bank supervisor** would be responsible for approving account creation requests and loan appointment requests. An **account holder** must be able to manage their accounts in terms of creation, activation, deactivation and reactivation. An account holder must also be able to generate **mini statements** for their respective accounts and apply for various loans. Currently, the bank offers two types of accounts – **Current** and **Savings**.

Following is a breakdown of various parts that were considered for designing the class diagram-

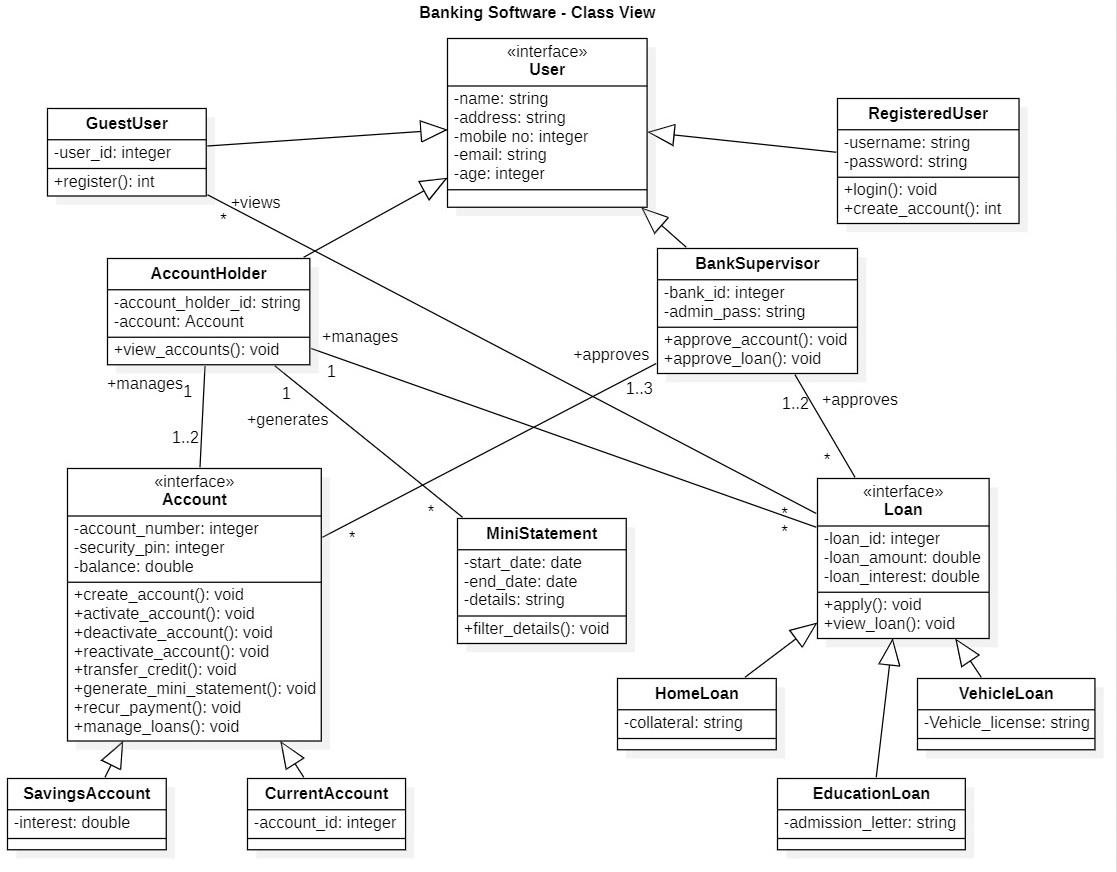
1. Noun/Noun Phrases.

Bank, User, Function, Guest User, Registered User, Account Holder, Supervisor, Account, Loan, Home Loan, Education Loan, Vehicle Loan, Interface, Mini Statement, Savings Account, Current Account.

1. Classes.

User, GuestUser, RegisteredUser, AccountHolder, Supervisor, Account, SavingsAccount, CurrentAccount, MiniStatement, Loan, HomeLoan, EducationLoan, VehicleLoan.

1. Verb Phrases.
   1. Guest User registers on the software.
   2. Guest User views Loans.
   3. Registered User logins.
   4. Registered User creates an Account.
   5. Account Holder views Accounts.
   6. Account Holder activates, deactivates or reactivates an Account.
   7. Account Holder transfers funds and makes recurring payments.
   8. Account Holder generates Mini Statement.
   9. Account Holder manages Loans.
   10. Supervisor approves Accounts and Loans.
2. Relations.
   1. Guest User, Registered User, Account Holder and Supervisor are different users of the software and hence are generalized to User.
   2. Education Loan, Home Loan and Vehicle Loan are generalized to Loan.
   3. Savings Account and Current Account are generalized to Account.
   4. An Account Holder may have one or two Accounts.
   5. An Account Holder must have at-least one bank Account.
   6. An Account is required to be approved by one to three Supervisor.
   7. An Account Holder may manage zero or more Loans.
   8. A Loan is to be approved by one to two Supervisor.
   9. An Account Holder may generate zero or more Mini Statements.
   10. A Guest User may view zero or more Loans.
3. Class Diagram.



Following are the code snippets that were written corresponding to each class and interface demonstrated in the above diagram-

User

public interface User{ String name; String address; *int* mobile\_no; String email;

*int* age;

}

Guest User

public class GuestUser implements User{ private *int* user\_id;

public *int* register(){}

}

Registered User

public class RegisteredUser implements User{ private String username;

private String password;

public *void* login(){}

public *int* create\_account(){}

}

Account Holder

public class AccountHolder implements User{ private String account\_holder\_id; private Account account;

public *void* view\_accounts(){}

}

Supervisor

public class BankSupervisor implements User{ private *int* bank\_id;

private String admin\_pass;

public *void* approve\_account(){} public *void* approve\_loan(){}

}

Account

public interface Account{ *int* account\_number; *int* security\_pin; *double* balance;

public *void* create\_account(); public *void* activate\_account(); public *void* deactivate\_account(); public *void* reactivate\_account(); public *void* transfer\_credit();

public *void* generate\_mini\_statement(); public *void* recur\_payment();

public *void* manage\_loans();

}

Savings Account

public class SavingsAccount implements Account{

private *double* interest;

public *void* create\_account(){} public *void* activate\_account(){} public *void* deactivate\_account(){} public *void* reactivate\_account(){} public *void* transfer\_credit(){}

public *void* generate\_mini\_statement(){} public *void* recur\_payment(){}

public *void* manage\_loans(){}

}

Current Account

public class CurrentAccount implements Account{ private *int* account\_id;

public *void* create\_account(){} public *void* activate\_account(){} public *void* deactivate\_account(){} public *void* reactivate\_account(){} public *void* transfer\_credit(){}

public *void* generate\_mini\_statement(){} public *void* recur\_payment(){}

public *void* manage\_loans(){}

}

Loan

public interface Loan{

*int* loan\_id;

*double* loan\_amount;

*double* loan\_interest;

public *void* apply(); public *void* view\_loan();

}

Home Loan

public class HomeLoan implements Loan{ private String collateral;

public *void* apply(){} public *void* view\_loan(){}

}

Education Loan

public class EducationLoan implements Loan{ private String admission\_letter;

public *void* apply(){} public *void* view\_loan(){}

}

Vehicle Loan

public class VehicleLoan implements Loan{ private String vehicle\_license;

public *void* apply(){} public *void* view\_loan(){}

}

Mini Statement

public class MiniStatement{ private Date start\_date; private Date end\_date; private String details;

public *void* filter\_details(){}

}

# Conclusion:

By performing this experiment, we were able to understand various class diagram related concepts in designing a software. We were also able to construct a class diagram for our proposed banking software and write class declarations in Java.