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.

[2] Classes.

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

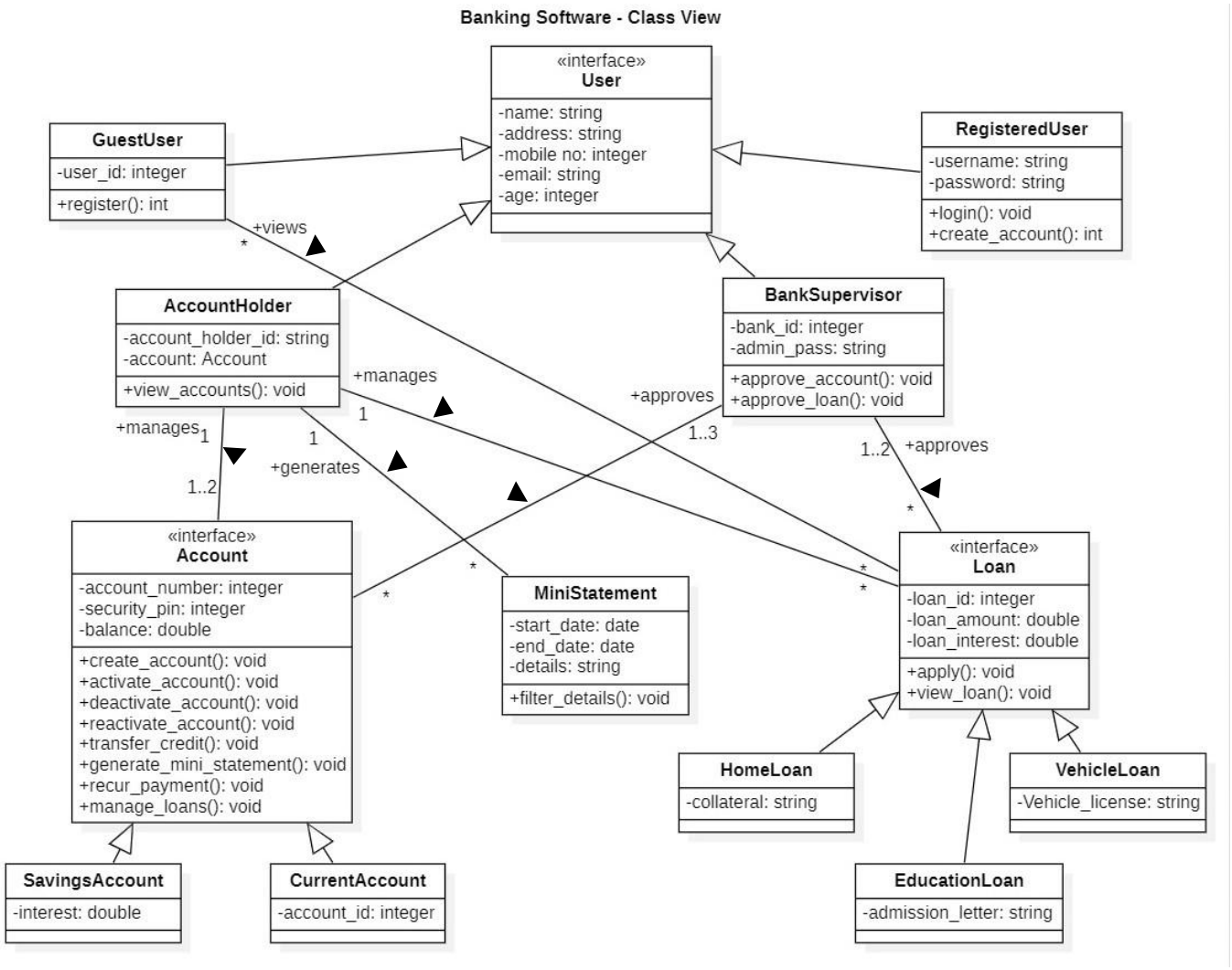
[3] Verb Phrases.

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.

[4] 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.

[5] 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.