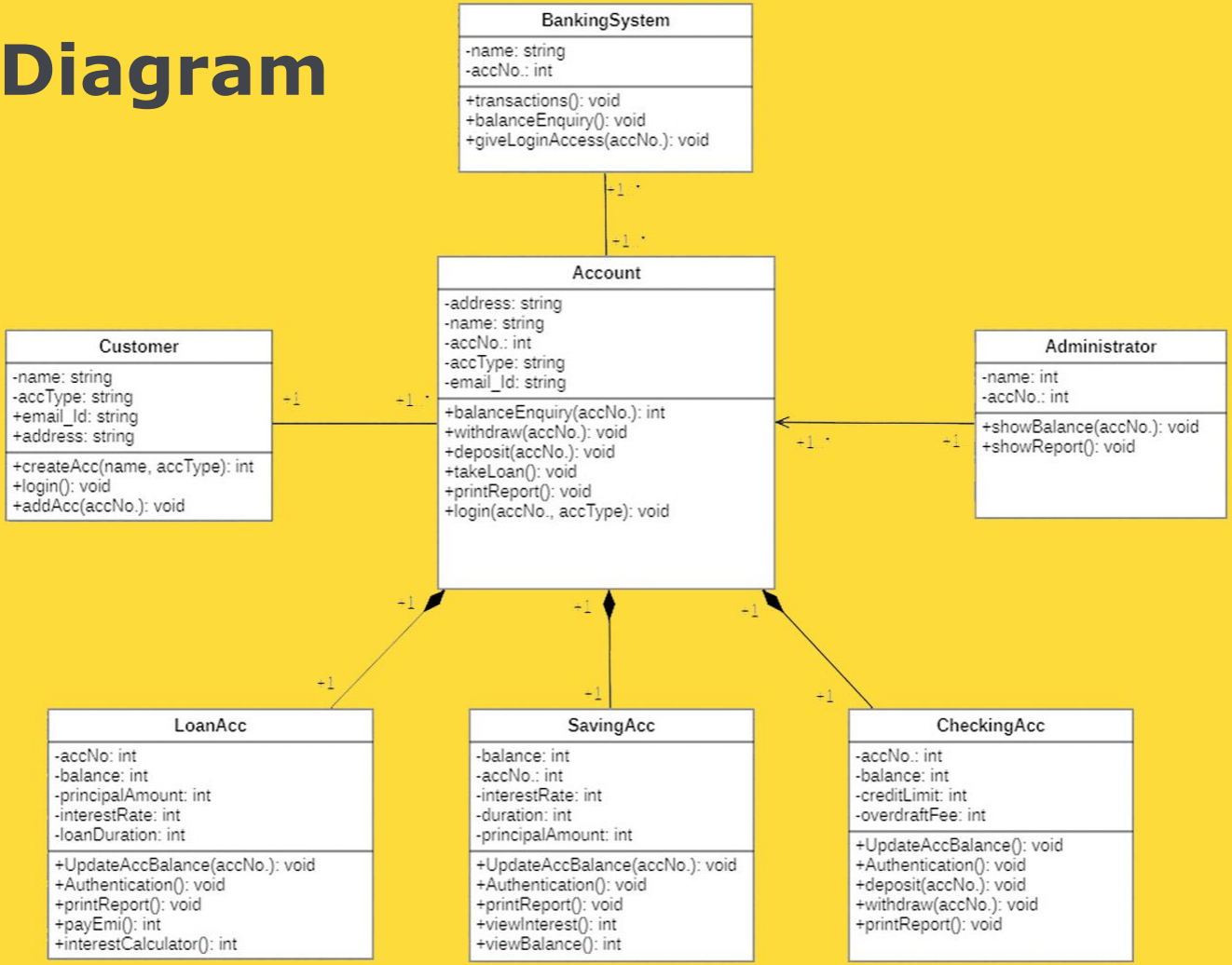
Bank Management System

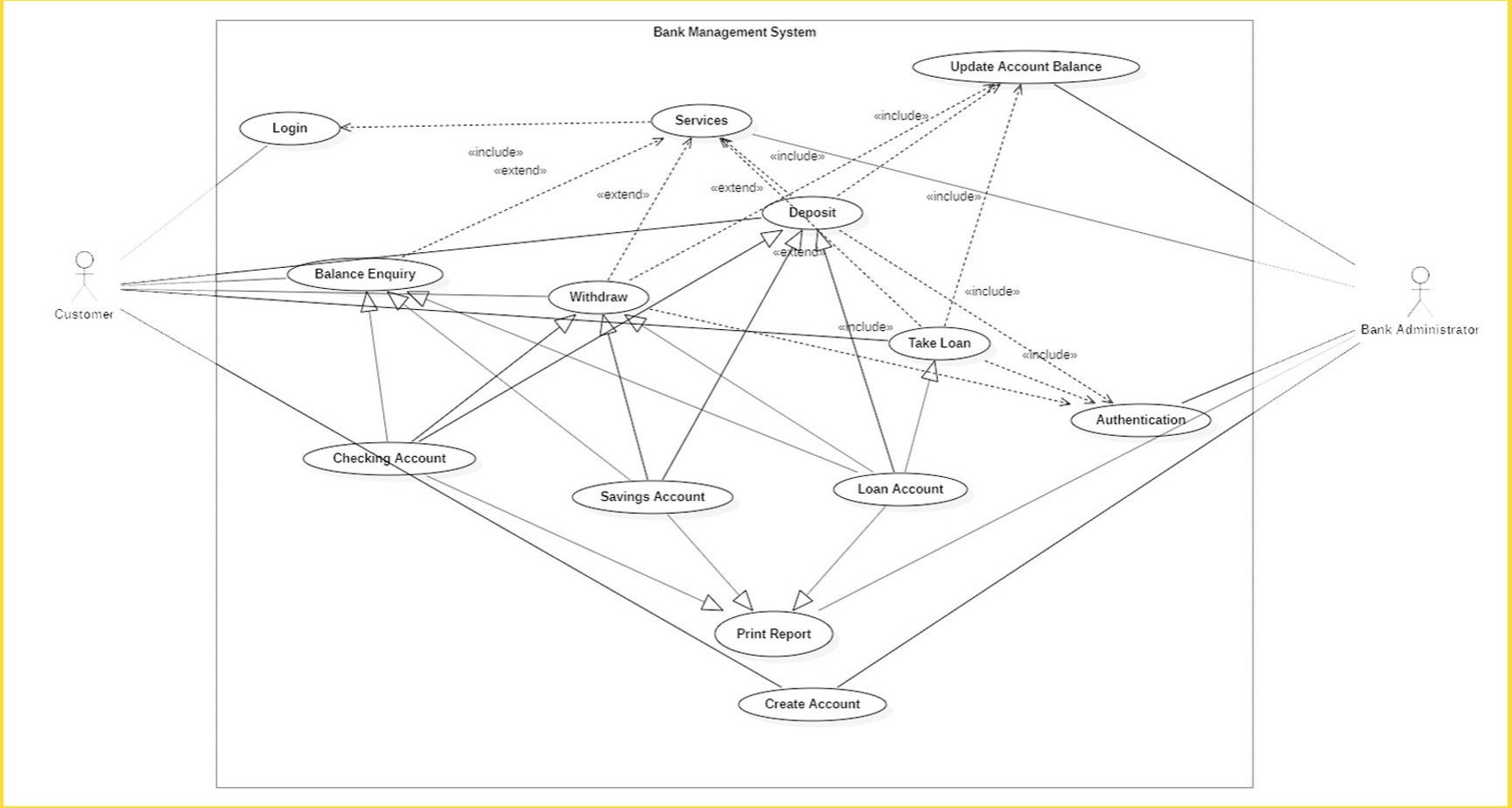
Mehul Garg - IIT2019041
Smitesh Hadape - IIT2019090
Tanish Patel - IIT2019092

UML Diagrams

Class Diagram



Use Case Diagram



CRC Diagram

Customer	
• login • createAccount • addAccount	• Account

Edit card #1 × ↓

Account	
• balanceEnquiry • withdraw • deposit • takeLoan	• Banking system • Administrator

Edit card #2 × ↑ ↓

checkingAcc	Account
• updateAccBalance • Authentication	• Account • Administrator • BankingSystem

Edit card #3 × ↑ ↓

savingsAcc	Account
• updateAccBalance • Authentication	• Account • Administrator • BankingSystem

Edit card #4 × ↑ ↓

loanAcc	Account
• UpdateAccBalance • Authentication • takeLoan	• Account • Administrator • BankingSystem

Edit card #5 × ↑ ↓

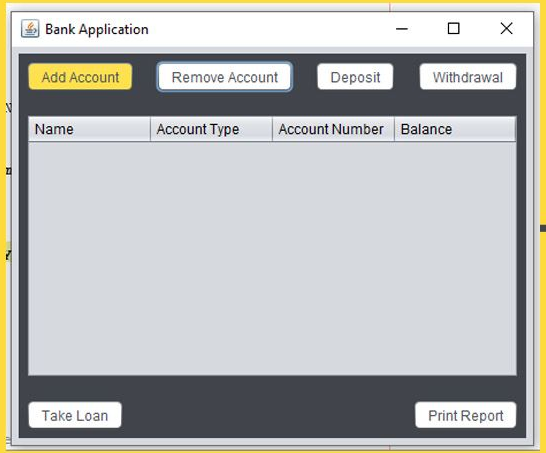
Banking System	
• transactions • balanceEnquiry	

Edit card #6 × ↑ ↓

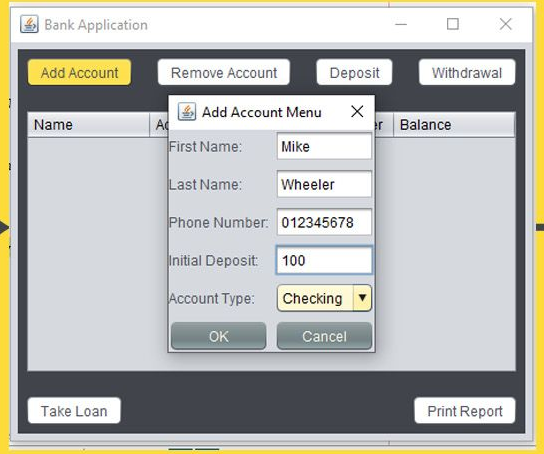
Administrator	
• balanceEnquiry • printReport	

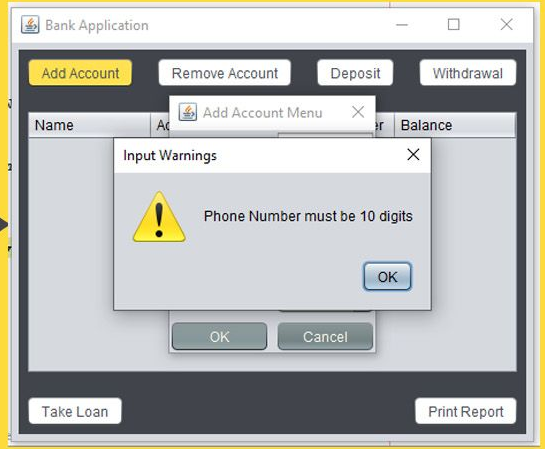
Edit card #7 × ↑

Project Walkthrough

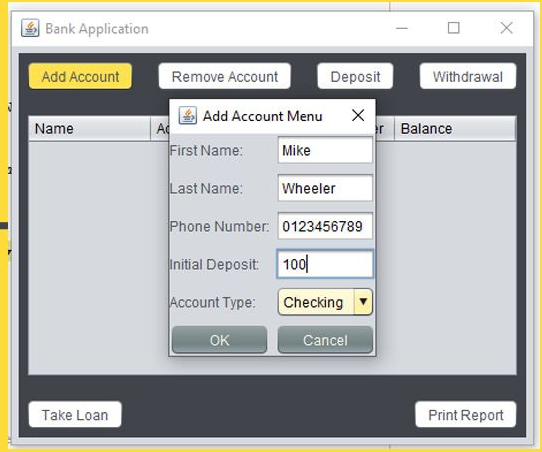


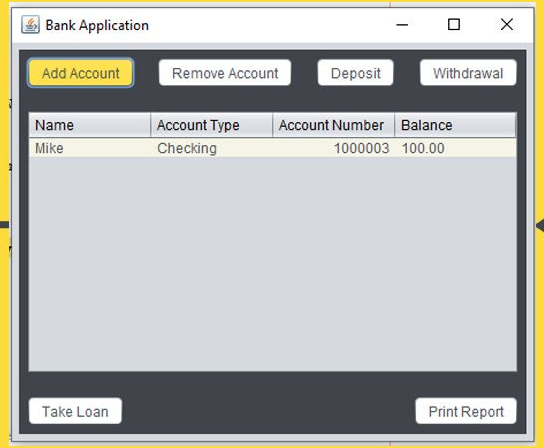
Create Account



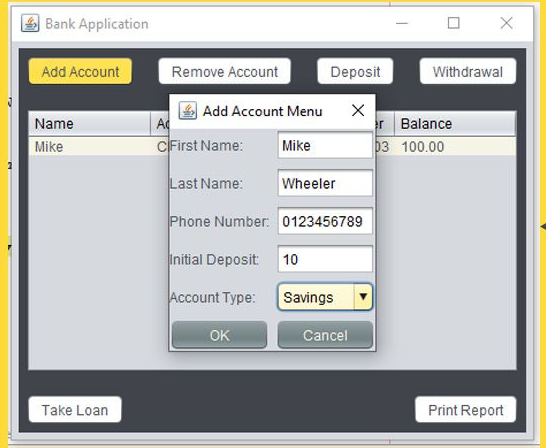


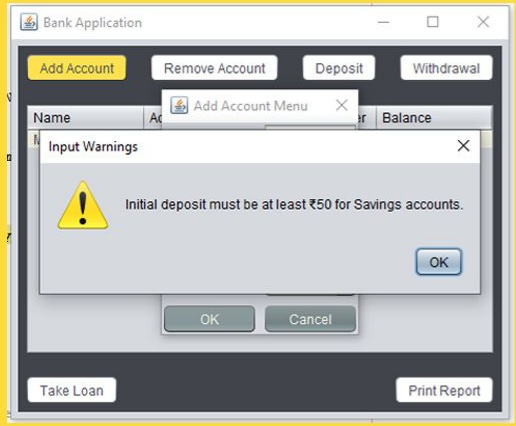
Error



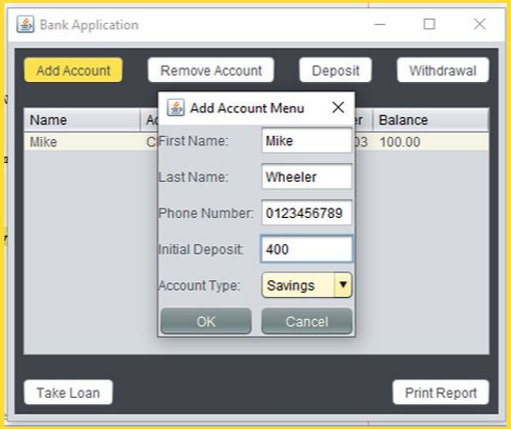


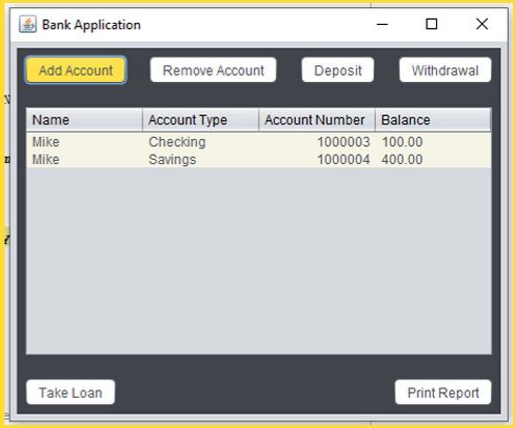
Add more Accounts



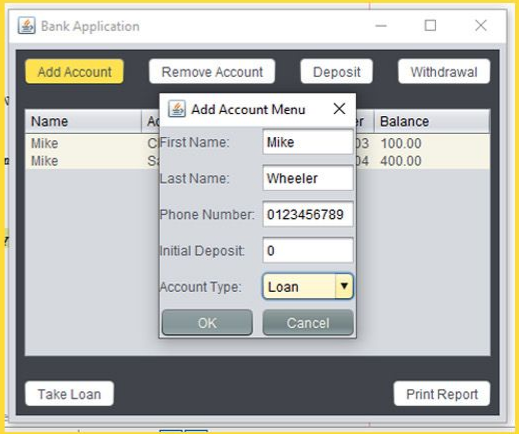


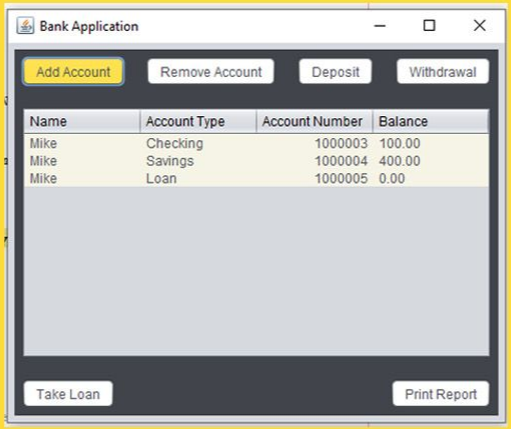
Minimum Deposit Amount



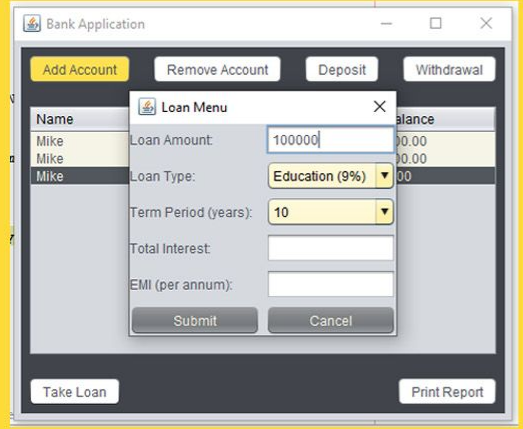


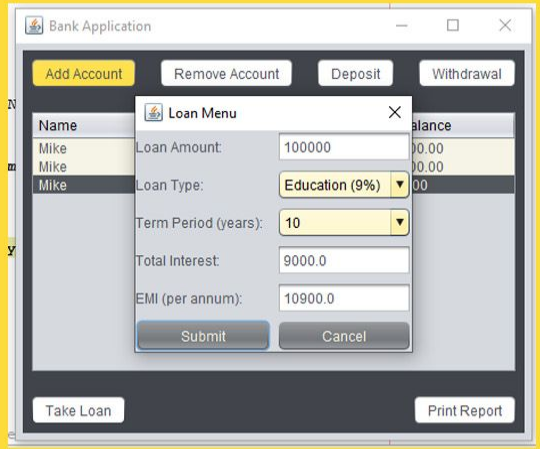
Make Loan Account

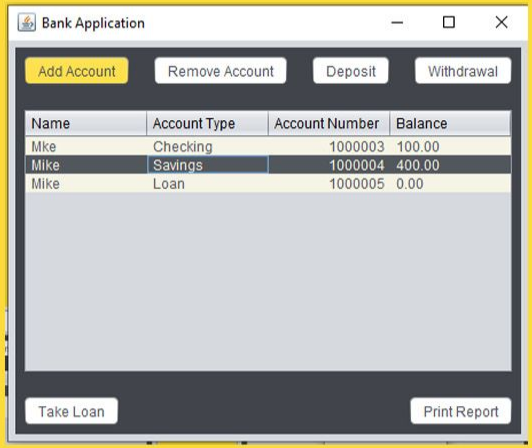


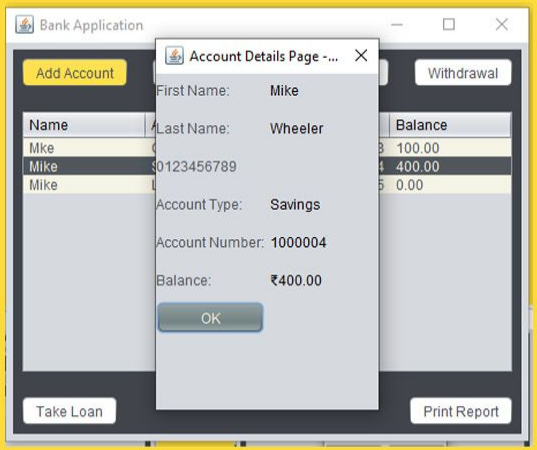


Take Loan

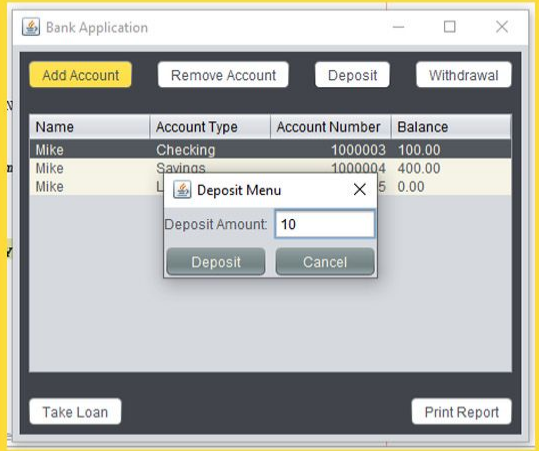




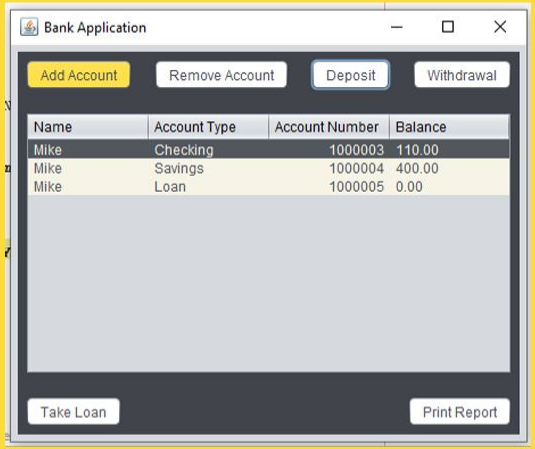


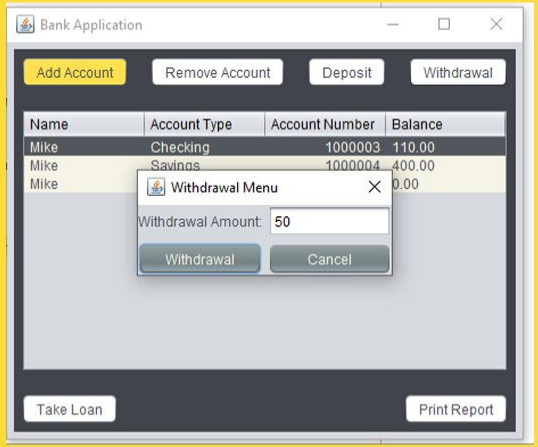


View Details

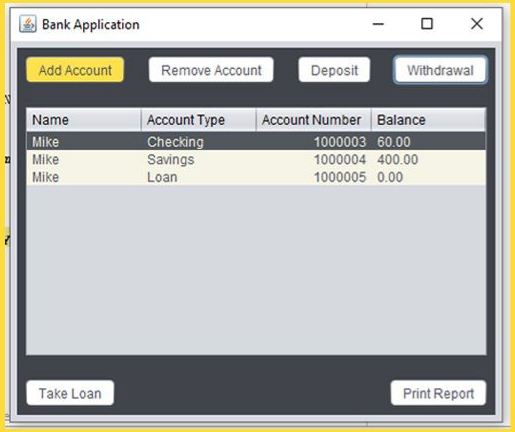


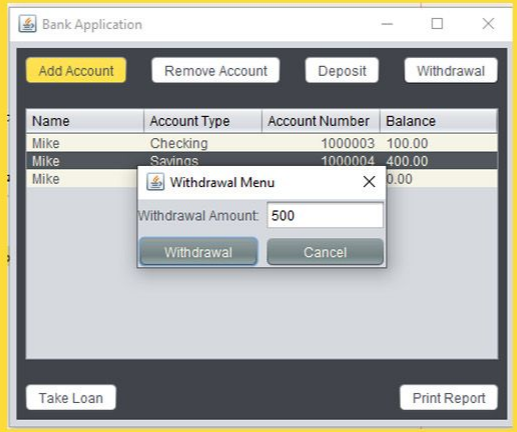
Deposit



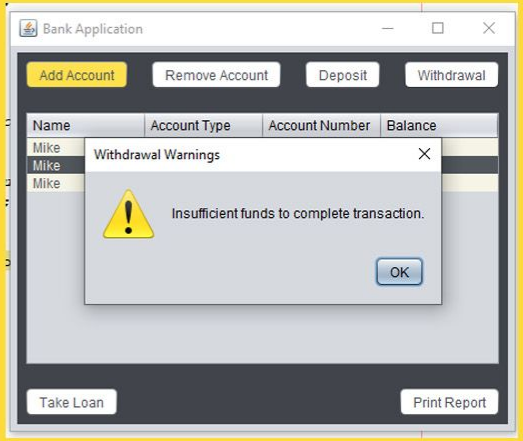


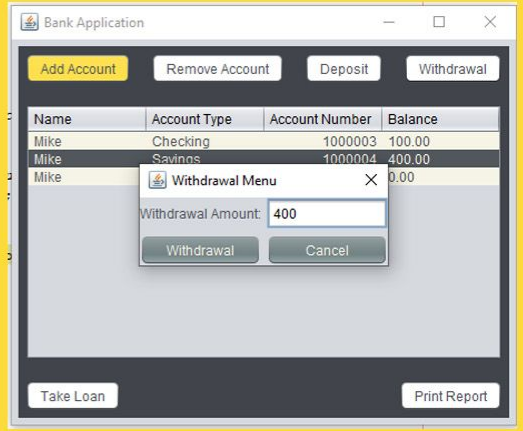
Withdrawal

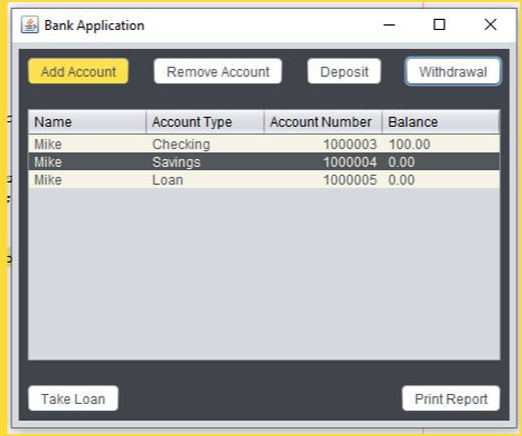


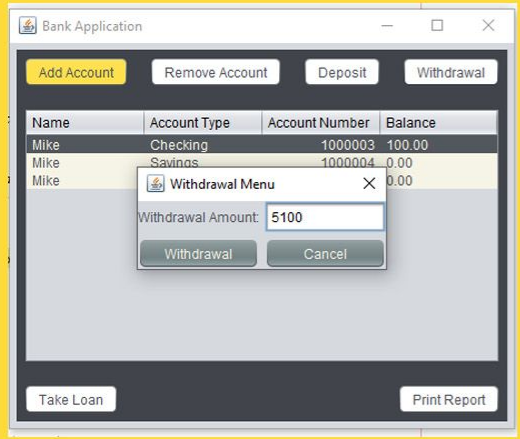


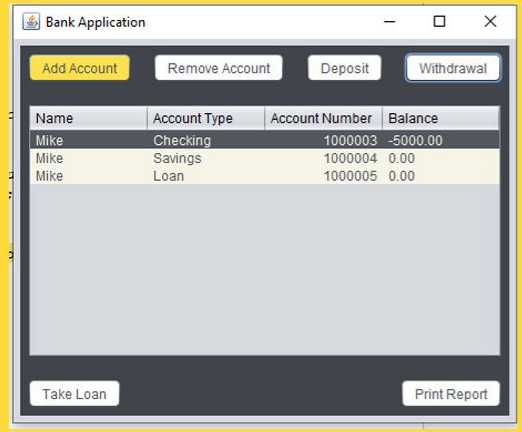
Withdrawal exceeds



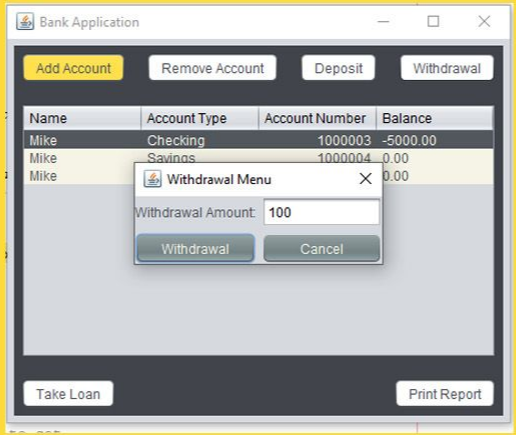


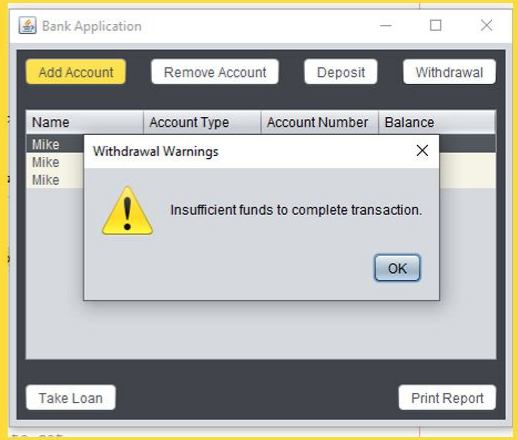


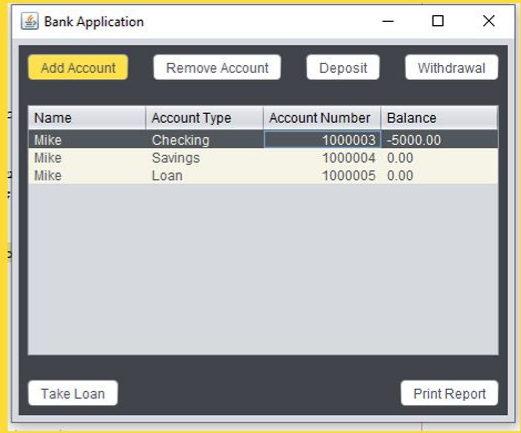




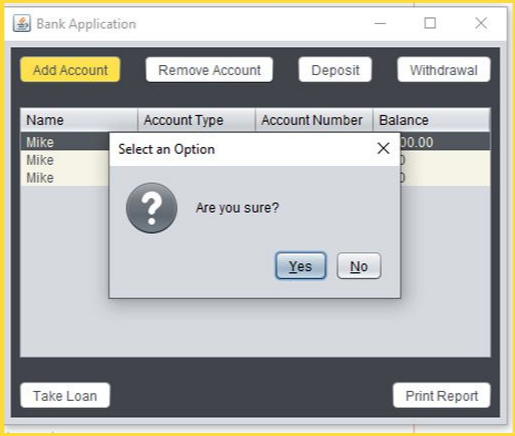
Minimum credit limit

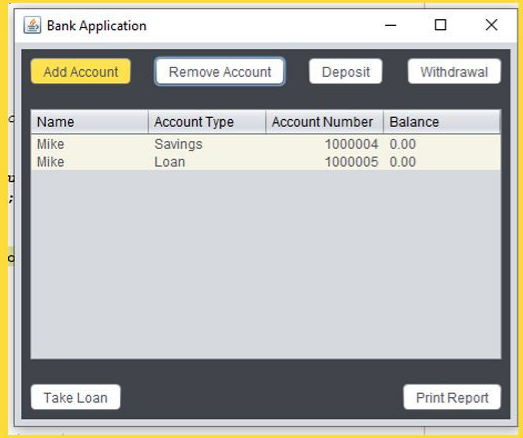




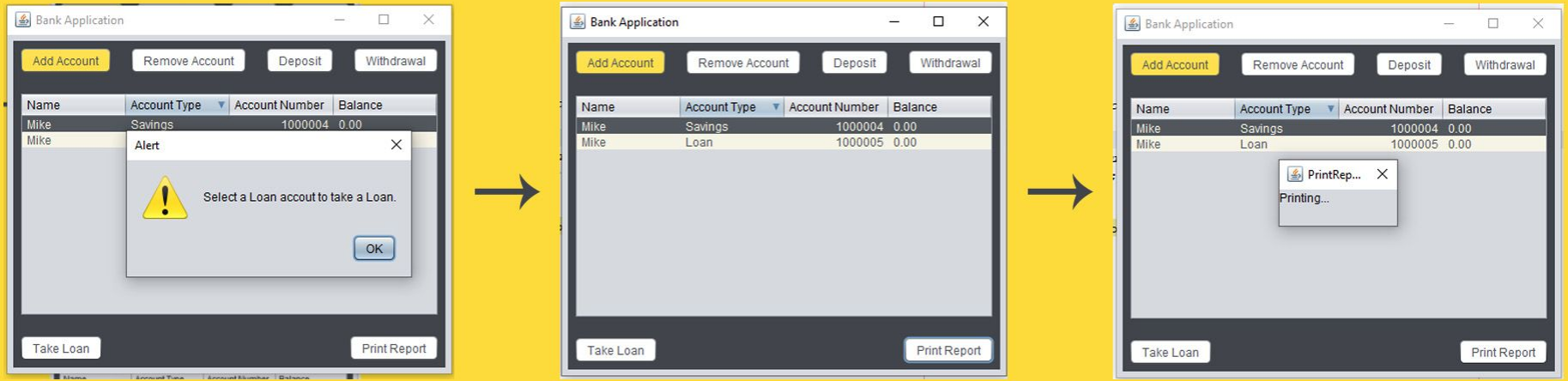


Remove Account





Print Report



Project Code

Code for Account.java

```
if(amount  > balance){

        throw new InsufficientFundsException();

         }

        else

        {

         balance=balance-amount;

        }
```

```
public void deposit(double amount) throws
InvalidAmountException{

        if(amount <= 0){

            throw new InvalidAmountException();

        }

        balance += amount; }

public double Calcroi(double LAmt,double roi,int
timep){

        LAmt+=(LAmt*roi)/100;

        LAmt=LAmt/timep;

        return LAmt; }
```

Code for Account.java(Part 2)

```
public double checkInterest(double LAmt,double
roi,int timep){

    LAmt=(LAmt*roi)/100;

    return LAmt;

    }
```

```
public void deposit(double amount) throws
InvalidAmountException{

        if(amount <= 0){

            throw new InvalidAmountException();

        }

        balance += amount;

    }
```

Code for Bank.java

```
private ArrayList<Customer> customers = new
ArrayList<Customer>();

void addCustomer(Customer customer) {
if(accountExists(customer.getAccount().getAccountNu
mber())){
customer.getAccount().setAccountNumber(findValidAc
countNumber());

    }

        customers.add(customer);

    }
```

```
private boolean accountExists(int accountNumber){

        for(Customer c : customers){

            if(c.getAccount().getAccountNumber() ==
accountNumber){

                return true;

            }

        }

        return false;

    }
```

Code for Bank.java

```
private int findValidAccountNumber(){

        int accountNumber;

        do {

            accountNumber =
Account.getNextAccountNumber();

        } while(accountExists(accountNumber));

        return accountNumber;

    }
```

```
Customer getCustomerByAccountNumber(int
accountNumber) {

        Customer customer = null;

        for(Customer c : customers){

            if(c.getAccount().getAccountNumber() ==
accountNumber){

                customer = c;

                break; }

        }

        return customer;}
```

Code for Checking.java

```
private static String accountType = "Checking";

    Checking(double initialDeposit){

        this.setBalance(initialDeposit);}

    @Override

    public String toString(){

        return "Account Type: " + accountType + " Account\n" +

                "Account Number: " + this.getAccountNumber() + "\n" +

                "Balance: " + this.getBalance() + "\n" +

                "Interest Rate: " + (this.getInterest() * 100) + "%\n"; }
```

Code for Customer.java

```
public String toString(){

        return "\nCustomer Information\n" +

                "First Name: " + getFirstName() + "\n" +

                "Last Name: " + getLastName() +  "\n" +

                "SSN: " + getSsn() +  "\n" +

                account;}

    public String basicInfo(){

        return " Account Number: " + account.getAccountNumber() + " - Name: " + getFirstName() + " " +
getLastName();

    }
```

Code for DepositMenu.java

```
private void
depositButtonActionPerformed(java.awt.event.ActionEvent evt) {

        StringBuilder warnings = new StringBuilder();

        if (amountField.getText().isEmpty()) {

            warnings.append("Deposit amount is required.\n");

        } else {

            double amount = 0;

 try { amount =
Bank.round(Double.parseDouble(amountField.getText()), 2);

       int result = JOptionPane.showConfirmDialog(this, "Deposit ₹"
+ String.format("%.2f", amount) + " to the account?");

                if (result == JOptionPane.OK_OPTION) {

                   try {

                       customer.getAccount().deposit(amount);

                       this.dispose();

                   } catch (InvalidAmountException ex) {

                       warnings.append("Deposit amount is
invalid.\n"); }

                  }

               } catch (NumberFormatException ex) {

                   warnings.append("Deposit must be a number.\n"); }
```

Code for Loan.java

```
Loan(double LoanAmt, double roi, int timep){
        this.setLoanAmount(LoanAmt);
        this.setInterest(roi);
        this.setTimePeriod(timep);}
    public String toString(){
        return "Account Type: " + accountType + " Account\n" +
                "Account Number: " + this.getAccountNumber() + "\n" +
                "Balance: " + this.getBalance() + "\n" +
                "Interest Rate: " + (this.getInterest() * 100) + "%\n";}
```

Code for LoanMenu.java

```
LType=LoanType.getSelectedItem().toString();

        if (LoanType.getSelectedItem().toString() ==
"Education (9%)") {

            roi=9.00;

            account=new Loan(LAmt, roi, timep); }

        else if(LoanType.getSelectedItem().toString() ==
"Home (7%)") {

            roi=7.00;

            account=new Loan(LAmt, roi, timep);}
```

```
else if(LoanType.getSelectedItem().toString() == "Gold
(9%)") {

            roi=9.00;

            account=new Loan(LAmt, roi, timep); }

        else if(LoanType.getSelectedItem().toString() ==
"Personal (10%)") {

            roi=10.00;

            account=new Loan(LAmt, roi, timep);

        }
```

Code for MainMenu.java

```
private void
accountTableMouseClicked(java.awt.event.MouseEvent evt)
{

        setAccountButtonsActive(true);

        if (evt.getClickCount() == 2) {

            int selectedRow = accountTable.getSelectedRow();

            Customer customer =
getSelectedCustomer(selectedRow);

            if (customer != null) {

                AccountDetailsPage page = new
AccountDetailsPage(this, true, customer);

                page.setVisible(true); } }}
```

```
private Customer getSelectedCustomer(int selectedRow) {

        Customer customer = null;

        if (selectedRow >= 0) {

            int accountNumber = (int)
accountTable.getValueAt(selectedRow, 2);

            customer =
bank.getCustomerByAccountNumber(accountNumber);}

 return customer; }

 private void addCustomerToTable(Customer customer) {

        model.addRow(new Object[]{});

        reloadCustomerRowData(model.getRowCount() - 1,
customer); }
```

Code for Savings.java

```
public String toString(){
        return "Account Type: " + accountType + " Account\n" +
                "Account Number: " + this.getAccountNumber() + "\n" +
                "Balance: " + this.getBalance() + "\n" +
                "Interest Rate: " + (this.getInterest() * 100) + "%\n";
    }
    @Override
    public String getAccountType() {
        return accountType; }
```