Python Coding Challenge - Loan Management System

Problem Statement:

Create SQL Schema from the customer and loan class, use the class attributes for table column names.

1. **Define a `Customer` class with the following confidential attributes:**
    a. Customer ID
    b. Name
    c. Email Address
    d. Phone Number
    e. Address
    f. creditScore
2. **Define a base class `Loan` with the following attributes:**
    a. loanId
    b. customer (reference of customer class)
    c. principalAmount
    d. interestRate
    e. loanTerm (Loan Tenure in months)
    f. loanType (CarLoan, HomeLoan)
    g. loanStatus (Pending, Approved)

```sql
CREATE DATABASE IF NOT EXISTS loan_management;
USE loan_management;
DROP DATABASE loan_management;

-- Table: Customers
CREATE TABLE IF NOT EXISTS customers (
    customer_id VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15),
    address VARCHAR(255),
    credit_score INT
);
-- Table: Loans
CREATE TABLE IF NOT EXISTS loans (
    loan_id VARCHAR(10) PRIMARY KEY,
    customer_id VARCHAR(10),
    principal DOUBLE,
    interest_rate DOUBLE,
    loan_term INT,
    loan_type VARCHAR(50),
    loan_status VARCHAR(20),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

```
25
26      -- Sample Customers
27 •    INSERT INTO customers VALUES
28      ('C101', 'Pooja', 'pooja@mail.com', '9876543210', 'Bangalore', 720),
29      ('C102', 'Ravi', 'ravi@gmail.com', '9812345678', 'Chennai', 640);
30
31      -- Sample Loans
32 •    INSERT INTO loans VALUES
33      ('L101', 'C101', 500000, 8.5, 60, 'HomeLoan', 'Pending'),
34      ('L102', 'C102', 300000, 9.0, 36, 'CarLoan', 'Pending');
35
```

```
38 •  SELECT * FROM customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

loan_id	customer_id	principal	interest_rate	loan_term	loan_type	loan_status
L101	C101	500000	8.5	60	HomeLoan	Pending
L102	C102	300000	9	36	CarLoan	Rejected
l104	c101	600000	5	24	CarLoan	Pending
l105	c101	700000	5	24	CarLoan	Pending
l1l3	c1c3	30000	5	12	car loan	Rejected

```
36 •  SELECT * FROM loans;
37
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

loan_id	customer_id	principal	interest_rate	loan_term	loan_type	loan_status
L101	C101	500000	8.5	60	HomeLoan	Pending
L102	C102	300000	9	36	CarLoan	Rejected
l104	c101	600000	5	24	CarLoan	Pending
l105	c101	700000	5	24	CarLoan	Pending
l1l3	c1c3	30000	5	12	car loan	Rejected

3. Create two subclasses: `HomeLoan` and `CarLoan`. These subclasses should inherit from the Loan class and add attributes specific to their loan types. For example:

a. HomeLoan should have a propertyAddress (String) and propertyValue (int) attribute.

b. CarLoan should have a carModel (String) and carValue (int) attribute.

PROGRAM CODE (ENTITY FOLDER):

#entity>home_loan.py

```
from entity.loan import Loan

class HomeLoan(Loan):
    def __init__(self, loan_id, customer_id, principal, interest_rate, loan_term,
                 property_address, property_value):
        super().__init__(loan_id, customer_id, principal, interest_rate, loan_term,
                 "HomeLoan")
        self.property_address = property_address
        self.property_value = property_value
    def __str__(self):
        base = super().__str__()
        return f"{base}\nProperty Addr  : {self.property_address}\nProperty Value :
                 ₹{self.property_value}"
```

#entity>car_loan.py

```
from entity.loan import Loan

class CarLoan(Loan):

        def __init__(self, loan_id, customer_id, principal, interest_rate, loan_term,
                car_model, car_value):

    super().__init__(loan_id, customer_id, principal, interest_rate, loan_term,
                "CarLoan")

    self.car_model = car_model

    self.car_value = car_value


        def __str__(self):

                base = super().__str__()

                return f"{base}\nCar Model     : {self.car_model}\nCar Value     :
                ₹{self.car_value}"
```

#entity>customer.py

```
class Customer:

  def __init__(self, customer_id, name, email, phone, address, credit_score):

            self.customer_id = customer_id

            self.name = name

            self.email = email

            self.phone = phone

            self.address = address

            self.credit_score = credit_score

  def __str__(self):

            return (f"Customer ID    : {self.customer_id}\n"

                f"Name          : {self.name}\n"

                f"Email         : {self.email}\n"

                f"Phone         : {self.phone}\n"

                f"Address       : {self.address}\n"

                f"Credit Score   : {self.credit_score}")
```

#entity>home_loan.py

```
from entity.loan import Loan

class HomeLoan(Loan):

        def __init__(self, loan_id, customer_id, principal, interest_rate, loan_term,
                        property_address, property_value):

        super().__init__(loan_id, customer_id, principal, interest_rate, loan_term,
                        "HomeLoan")

        self.property_address = property_address

         self.property_value = property_value

   def __str__(self):

         base = super().__str__()

         return f"{base}\nProperty Addr  : {self.property_address}\nProperty Value :
₹{self.property_value}"
```

4. Implement the following for all classes. a. Write default constructors and overload the constructor with parameters, generate getter and setter, (print all information of attribute) methods for the attributes.

PROGRAM CODE (SERVICE FOLDER)

#service\loan_service.py

```
from abc import ABC, abstractmethod

class LoanService(ABC):

   @abstractmethod

   def apply_loan(self, loan): pass

   @abstractmethod

  def calculate_interest(self, loan_id=None, principal=None, rate=None, term=None):
pass

   @abstractmethod

   def loan_status(self, loan_id): pass

   @abstractmethod

    def calculate_emi(self, loan_id=None, principal=None, rate=None, term=None):
pass

   @abstractmethod

   def loan_repayment(self, loan_id, amount): pas
```

```
@abstractmethod
def get_all_loans(self): pass
@abstractmethod
def get_loan_by_id(self, loan_id): pass
```

5. Define ILoanRepository interface/abstract class with following methods to interact with database.

a. applyLoan(loan Loan): pass appropriate parameters for creating loan. Initially loan status is pending and stored in database. before storing in database get confirmation from the user as Yes/No b. calculateInterest(loanId): This method should calculate and return the interest amount for the loan. Loan should be retrieved from database and calculate the interest amount if loan not found generate InvalidLoanException.

i. Overload the same method with required parameters to calculate the loan interest amount. It is used to calculate the loan interest while creating loan.

ii.Interest = (Principal Amount * Interest Rate * Loan Tenure) / 12

c. loanStatus(loanId): This method should display a message indicating that the loan is approved or rejected based on credit score, if credit score above 650 loan approved else rejected and should update in database.

d. calculateEMI(loanId): This method will calculate the emi amount for a month to repayment. Loan should be retrieved from database and calculate the interest amount, if loan not found generate InvalidLoanException.

i. Overload the same method with required parameters to calculate the loan EMI amount. It is used to calculate the loan EMI while creating loan.

ii. EMI = [P * R * (1+R)^N] / [(1+R)^N-1]

1. EMI: The Equated Monthly Installment.
2. P: Principal Amount (Loan Amount).
3. R: Monthly Interest Rate (Annual Interest Rate / 12 / 100).
4. N: Loan Tenure in months.

e. loanRepayment(loanId, amount): calculate the noOfEmi can be paid from the amount if the amount is less than single emi reject the payment or pay the emi in whole number and update the variable.

f. getAllLoan(): get all loan as list and print the details.

g. getLoanById(loanId): get loan and print the details, if loan not found generate InvalidLoanException.

6. Define ILoanRepositoryImpl class and implement the ILoanRepository interface and provide implementation of all methods.

PROGRAM CODE (SERVICE FOLDER)

```
# service\loan_service_impl.py

from service.loan_service import LoanService

from util.db_conn_util import get_db_connection

from exception.loan_exceptions import InvalidLoanException


class LoanServiceImpl(LoanService):


   def apply_loan(self, loan):

      con = get_db_connection()

      cursor = con.cursor()


      # Check if customer exists

          cursor.execute("SELECT * FROM customers WHERE customer_id = %s", (loan.customer_id,))

      if not cursor.fetchone():

           print(f"❌ Cannot apply loan. Customer ID '{loan.customer_id}' does not exist.")

          con.close()

          return


      #  Check if loan ID already exists

      cursor.execute("SELECT * FROM loans WHERE loan_id = %s", (loan.loan_id,))

      if cursor.fetchone():

          print(f"❌ Loan ID '{loan.loan_id}' already exists.")

          con.close()
```

```
            return

        # Validate inputs
        if loan.principal <= 0 or loan.interest_rate <= 0 or loan.loan_term <= 0:
            print("❌ Invalid loan details. Principal, interest rate, and loan term must be greater than 0.")
            con.close()
            return

        if loan.loan_type not in ["HomeLoan", "CarLoan"]:
            print("❌ Invalid loan type. Only 'HomeLoan' or 'CarLoan' are allowed.")
            con.close()
            return

        # Confirm application
        confirm = input("Do you want to apply for this loan? (Yes/No): ").lower()
        if confirm != 'yes':
            print("❌ Loan application cancelled.")
            con.close()
            return

        # Insert loan
        cursor.execute("""
            INSERT INTO loans (loan_id, customer_id, principal, interest_rate, loan_term, loan_type, loan_status)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
        """, (loan.loan_id, loan.customer_id, loan.principal, loan.interest_rate,
              loan.loan_term, loan.loan_type, loan.loan_status))
        con.commit()
        con.close()
```

```
        print("✅ Loan applied successfully.")

    def calculate_interest(self, loan_id=None, principal=None, rate=None, term=None):
        if loan_id:
            con = get_db_connection()
            cursor = con.cursor()
            cursor.execute("SELECT principal, interest_rate, loan_term FROM loans WHERE loan_id = %s", (loan_id,))
            result = cursor.fetchone()
            con.close()
            if not result:
                raise InvalidLoanException("Loan ID not found.")
            principal, rate, term = result

        interest = (principal * rate * term) / 1200
        print(f"💡 Interest Amount: ₹{interest:.2f}")
        return round(interest, 2)

    def loan_status(self, loan_id):
        con = get_db_connection()
        cursor = con.cursor()
        cursor.execute("""
            SELECT c.credit_score FROM loans l
            JOIN customers c ON l.customer_id = c.customer_id
            WHERE l.loan_id = %s
        """, (loan_id,))
        row = cursor.fetchone()

        if not row:
            raise InvalidLoanException("Loan not found.")
```

```
        credit_score = row[0]
        status = 'Approved' if credit_score > 650 else 'Rejected'
        cursor.execute("UPDATE loans SET loan_status = %s WHERE loan_id = %s", (status, loan_id))
        con.commit()
        con.close()
        print(f" ✅ Loan {loan_id} status updated to: {status}")

    def calculate_emi(self, loan_id=None, principal=None, rate=None, term=None):
        if loan_id:
            con = get_db_connection()
            cursor = con.cursor()
            cursor.execute("SELECT principal, interest_rate, loan_term FROM loans WHERE loan_id = %s", (loan_id,))
            result = cursor.fetchone()
            con.close()
            if not result:
                raise InvalidLoanException("Loan not found.")
            principal, rate, term = result

        r = rate / 12 / 100
        emi = principal * r * ((1 + r) ** term) / (((1 + r) ** term) - 1)
        print(f" 💸 EMI Amount: ₹{emi:.2f}")
        return round(emi, 2)

    def loan_repayment(self, loan_id, amount):
        emi = self.calculate_emi(loan_id)
        if amount < emi:
            print(" ❌ Amount is less than one EMI. Cannot proceed.")
```

```
            return
        num_emis = int(amount // emi)
        remaining = amount % emi
                print(f"✅ {num_emis} EMI(s) paid. Remaining balance returned: ₹{remaining:.2f}")


    def get_all_loans(self):
        con = get_db_connection()
        cursor = con.cursor()
        cursor.execute("""
            SELECT l.*, c.name, c.email FROM loans l
            JOIN customers c ON l.customer_id = c.customer_id
        """)
        results = cursor.fetchall()
        con.close()
        return results


    def get_loan_by_id(self, loan_id):
        con = get_db_connection()
        cursor = con.cursor()
        cursor.execute("""
            SELECT l.*, c.name, c.email FROM loans l
            JOIN customers c ON l.customer_id = c.customer_id
            WHERE l.loan_id = %s
        """, (loan_id,))
        result = cursor.fetchone()
        con.close()
        if not result:
            raise InvalidLoanException("Loan ID not found.")
        return result
```

7. Create DBUtil class and add the following method.

a. static getDBConn():Connection Establish a connection to the database and return Connection reference

PROGRAM CODE (UTIL FOLDER)

```
#util\db_conn_util.py
import mysql.connector
from util.db_property_util import get_db_properties


def get_db_connection():
    props = get_db_properties()
    con = mysql.connector.connect(
        host=props['host'],
        user=props['user'],
        password=props['password'],
        database=props['database']
    )
    return con
```

 # util/db_connection.py

```
def get_db_properties():
    return {
        "host": "localhost",
        "user": "root",
        "password": "Poojashree",
        "database": "loan_management"
    }
```

PROGRAM CODE (EXCEPTION FOLDER)

```
class InvalidLoanException(Exception)
        def __init__(self, message):
            super().__init__(message)
```

8. Create LoanManagement main class and perform following operation:

a. main method to simulate the loan management system. Allow the user to interact with the system by entering choice from menu such as "applyLoan", "getAllLoan", "getLoan", "loanRepayment", "exit."

PROGRAM CODE (MAIN FOLDER):

#main\loan_management_main.py

```
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from util.db_conn_util import get_db_connection
from service.loan_service_impl import LoanServiceImpl
from entity.loan import Loan
from entity.customer import Customer

service = LoanServiceImpl()

def print_loan_details(loan):
    print("\n Loan Details")
    print(f"Loan ID       : {loan[0]}")
    print(f"Customer ID   : {loan[1]}")
    print(f"Principal     : ₹{loan[2]}")
    print(f"Interest Rate  : {loan[3]}%")
```

```
    print(f"Loan Term      : {loan[4]} months")
    print(f"Loan Type      : {loan[5]}")
    print(f"Loan Status    : {loan[6]}")
    print(f"Customer Name  : {loan[7]}")
    print(f"Customer Email : {loan[8]}")

def login():
    print("🔐 Login")
    username = input("Enter username: ")
    password = input("Enter password: ")

    if username == 'admin' and password == 'adminpass':
        return 'admin'
    elif username == 'user' and password == 'userpass':
        return 'user'
    else:
        print("❌ Invalid credentials")
        return None

def admin_menu():
    while True:
        print("\n===== Admin Menu =====")
        print("1. Add Customer")
        print("2. Apply for Loan")
        print("3. Calculate Interest")
        print("4. Calculate EMI")
        print("5. Update Loan Status")
        print("6. Loan Repayment")
        print("7. View All Loans")
        print("8. Get Loan by ID")
```

```
        print("0. Logout")

        ch = input("Enter your choice: ")

        if ch == '1':
            c = Customer(
                input("Customer ID: "),
                input("Name: "),
                input("Email: "),
                input("Phone: "),
                input("Address: "),
                int(input("Credit Score: "))
            )


            con = get_db_connection()

            cur = con.cursor()
                cur.execute("SELECT * FROM customers WHERE customer_id = %s",
(c.customer_id,))
            if cur.fetchone():
                print(f"❌ Customer ID '{c.customer_id}' already exists.")
            else:
                cur.execute("INSERT INTO customers VALUES (%s, %s, %s, %s, %s, %s)",
                        (c.customer_id, c.name, c.email, c.phone, c.address, c.credit_score))
                con.commit()
                print("✅ Customer added successfully.")

            con.close()
        elif ch == '2':
```

```
        l = Loan(
            input("Loan ID: "),
            input("Customer ID: "),
            float(input("Principal Amount: ")),
            float(input("Interest Rate: ")),
            int(input("Loan Term (months): ")),
            input("Loan Type: ")
        )
        service.apply_loan(l)

    elif ch == '3':
        lid = input("Loan ID: ")
        service.calculate_interest(loan_id=lid)

    elif ch == '4':
        lid = input("Loan ID: ")
        service.calculate_emi(loan_id=lid)

    elif ch == '5':
        lid = input("Loan ID: ")
        service.loan_status(lid)

    elif ch == '6':
        lid = input("Loan ID: ")
        amt = float(input("Amount to repay: ₹"))
        service.loan_repayment(lid, amt)

    elif ch == '7':
        loans = service.get_all_loans()
        for loan in loans:
```

```
            print_loan_details(loan)

        elif ch == '8':
            lid = input("Loan ID: ")
            try:
                loan = service.get_loan_by_id(lid)
                print_loan_details(loan)
            except Exception as e:
                print("❌", e)

        elif ch == '0':
            print("👋 Logging out.")
            break

        else:
            print("❌ Invalid choice.")

def user_menu():
    while True:
        print("\n===== User Menu =====")
        print("1. Apply for Loan")
        print("2. View My Loans")
        print("3. Calculate EMI")
        print("4. Loan Repayment")
        print("0. Logout")

        ch = input("Enter your choice: ")

        if ch == '1':
```

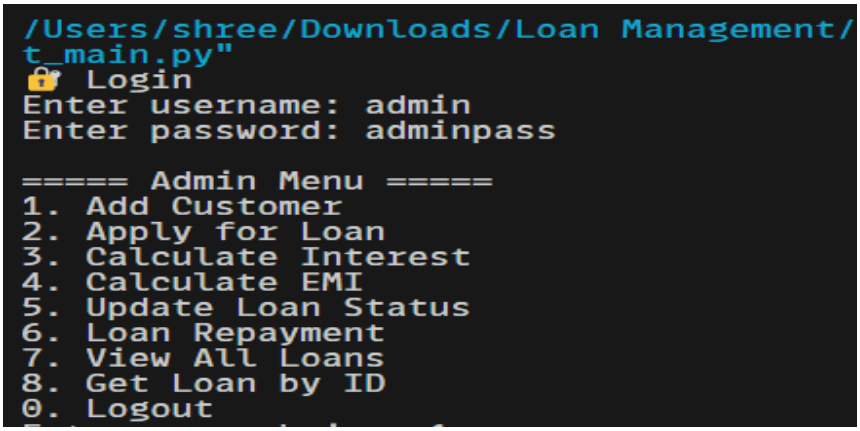
```
        l = Loan(
            input("Loan ID: "),
            input("Customer ID: "),
            float(input("Principal Amount: ")),
            float(input("Interest Rate: ")),
            int(input("Loan Term (months): ")),
            input("Loan Type: ")
        )
        service.apply_loan(l)

    elif ch == '2':
        cid = input("Enter your Customer ID: ")
        loans = service.get_all_loans()
        found = False
        for loan in loans:
            if loan[1] == cid:
                print_loan_details(loan)
                found = True
        if not found:
            print("❌ No loans found for this customer.")

    elif ch == '3':
        lid = input("Loan ID: ")
        service.calculate_emi(loan_id=lid)

    elif ch == '4':
        lid = input("Loan ID: ")
        amt = float(input("Enter repayment amount: ₹"))
        service.loan_repayment(lid, amt)
```

```
        elif ch == '0':
            print("👋 Logging out.")
            break

        else:
            print("❌ Invalid choice.")

# Entry point
role = login()
if role == 'admin':
    admin_menu()
elif role == 'user':
    user_menu()
```

LOAN MANAGEMENT SYSTEM CONSOLE OUTPUT :

Username	Password	Role
admin	Adminpass	Admin
user	Userpass	User

ADMIN LOGIN



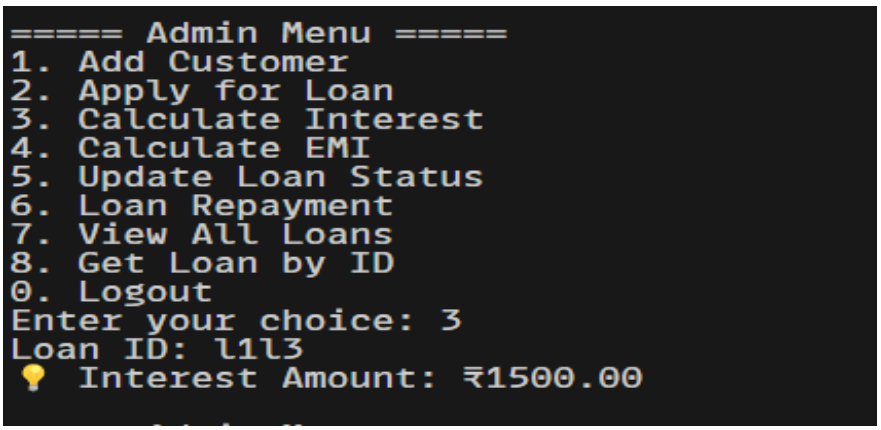
1. **ADD CUSTOMER :**

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 1
Customer ID: c1c3
Name: siva
Email: siva@gmail.com
Phone: 1423765809
Address: 1/308,annur
Credit Score: 7
✅ Customer added successfully.
```

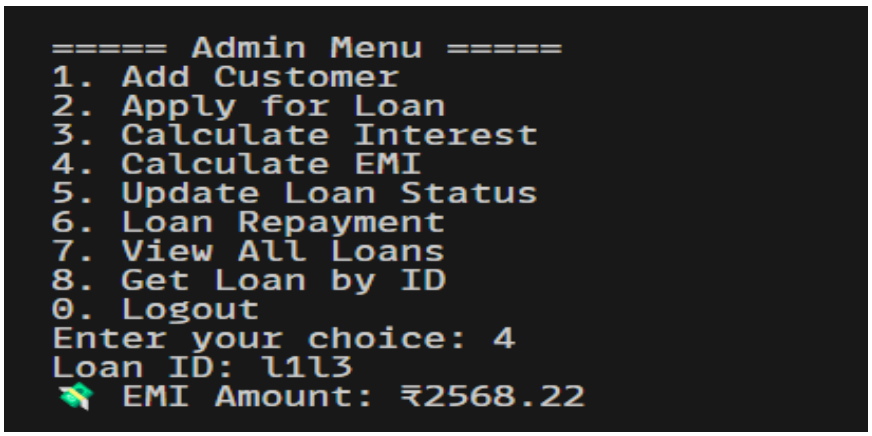
2. **APPLY FOR LOAN:**

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 2
Loan ID: l1l3
Customer ID: c1c3
Principal Amount: 30000
Interest Rate: 5
Loan Term (months): 12
Loan Type: car loan
Do you want to apply for this loan? (Yes/No): yes
✅ Loan applied successfully.
```

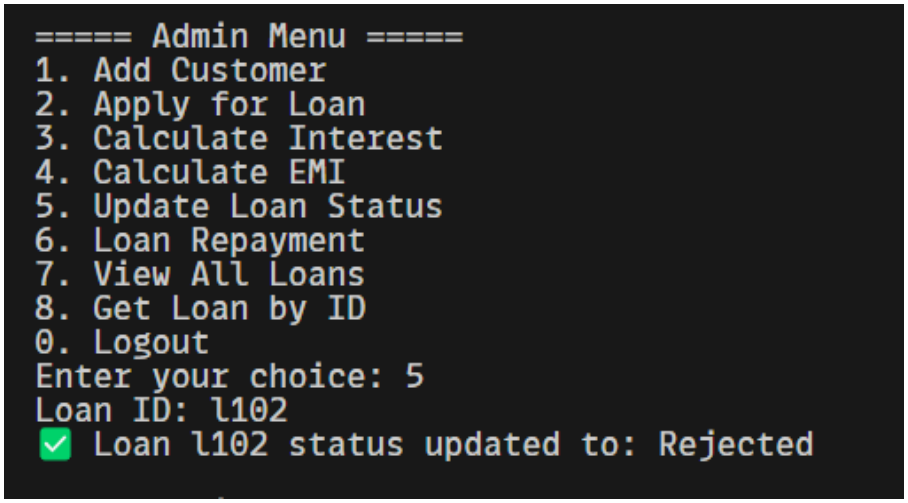
3. **CALCULATE INTEREST:**



4. **CALCULATE EMI :**



5. **UPDATE LOAN STATUS:**



6. LOAN REPAYMENT:

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 6
Loan ID: l1l3
Amount to repay: ₹30000
💸 EMI Amount: ₹2568.22
✅ 11 EMI(s) paid. Remaining balance returned: ₹1749.58

===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 6
Loan ID: l1l3
Amount to repay: ₹100
💸 EMI Amount: ₹2568.22
❌ Amount is less than one EMI. Cannot proceed.
```

7. VIEW ALL LOANS :

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 7

📄 Loan Details
Loan ID          : L101
Customer ID      : C101
Principal        : ₹500000.0
Interest Rate    : 8.5%
Loan Term        : 60 months
Loan Type        : HomeLoan
Loan Status      : Pending
Customer Name    : Pooja
Customer Email   : pooja@mail.com

📄 Loan Details
Loan ID          : L102
Customer ID      : C102
Principal        : ₹300000.0
Interest Rate    : 9.0%
Loan Term        : 36 months
Loan Type        : CarLoan
Loan Status      : Rejected
Customer Name    : Ravi
Customer Email   : ravi@gmail.com

📄 Loan Details
Loan ID          : l1l3
Customer ID      : c1c3
Principal        : ₹30000.0
Interest Rate    : 5.0%
Loan Term        : 12 months
Loan Type        : car loan
Loan Status      : Rejected
Customer Name    : siva
Customer Email   : siva@gmail.com
```

8. **GET LOAN BY ID :**

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 8
Loan ID: l102

📄 Loan Details
Loan ID          : L102
Customer ID      : C102
Principal        : ₹300000.0
Interest Rate    : 9.0%
Loan Term        : 36 months
Loan Type        : CarLoan
Loan Status      : Rejected
Customer Name    : Ravi
Customer Email   : ravi@gmail.com
```

0. **LOGOUT :**

```
===== Admin Menu =====
1. Add Customer
2. Apply for Loan
3. Calculate Interest
4. Calculate EMI
5. Update Loan Status
6. Loan Repayment
7. View All Loans
8. Get Loan by ID
0. Logout
Enter your choice: 0
👋 Logging out.
PS C:\Users\shree\Downloads\Loan Management>
```

USER LOGIN :

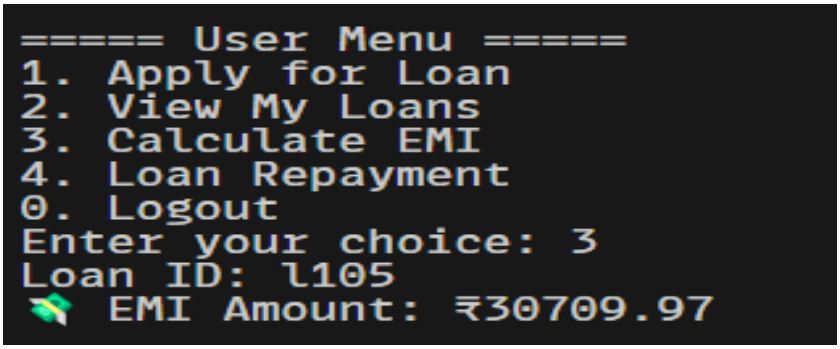
1. **APPLY FOR LOAN :**

```
🔐 Login
Enter username: user
Enter password: userpass

===== User Menu =====
1. Apply for Loan
2. View My Loans
3. Calculate EMI
4. Loan Repayment
0. Logout
Enter your choice: 1
Loan ID: l105
Customer ID: c101
Principal Amount: 700000
Interest Rate: 5
Loan Term (months): 24
Loan Type: CarLoan
Do you want to apply for this loan? (Yes/No): Yes
✅ Loan applied successfully.
```

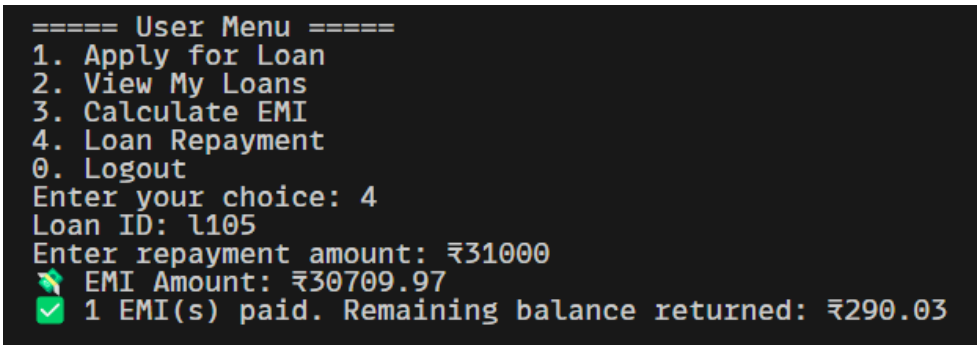
2. **VIEW MY LOANS:**

```
===== User Menu =====
1. Apply for Loan
2. View My Loans
3. Calculate EMI
4. Loan Repayment
0. Logout
Enter your choice: 2
Enter your Customer ID: c101

📄 Loan Details
Loan ID          : l104
Customer ID      : c101
Principal        : ₹600000.0
Interest Rate    : 5.0%
Loan Term        : 24 months
Loan Type        : CarLoan
Loan Status      : Pending
Customer Name    : Pooja
Customer Email   : pooja@mail.com

📄 Loan Details
Loan ID          : l105
Customer ID      : c101
Principal        : ₹700000.0
Interest Rate    : 5.0%
Loan Term        : 24 months
Loan Type        : CarLoan
Loan Status      : Pending
Customer Name    : Pooja
Customer Email   : pooja@mail.com
```

3. **CALCULATE EMI :**



4. **LOAN REPAYMENT :**



0. **LOGOUT :**