HEXAWARE CODING CHALLENGE

Done By: Ambati Sesha Sai Sahithya

TOPIC: 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

CODE:

```
class Customer:
    def __init__(self, customer_id=None, name=None, email=None, phone=None, address=None, credit_score=None):
        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}, Name: {self.name}, Email: {self.email}, Phone: {self.phone}, Address: {self.address}, Credit Score: {self.credit_score})"

    # Getters and Setters
    def get_customer_id(self):
        return self.customer_id

    def set_customer_id(self, customer_id):
        self.customer_id = customer_id

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_email(self):
        return self.email

    def set_email(self, email):
        self.email = email

    def get_phone(self):
        return self.phone

    def set_phone(self, phone):
        self.phone = phone

    def get_address(self):
        return self.address

    def set_address(self, address):
        self.address = address
    def get_credit_score(self):
        return self.credit_score
    def set_credit_score(self, credit_score):
        self.credit_score = credit_score
```

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)

CODE:

```
class Loan:
    def __init__(self, loan_id=None, customer=None,
principal_amount=None, interest_rate=None, loan_term=None,
loan_type=None, loan_status="Pending"):
        self._loan_id = loan_id
        self._customer = customer  # Instance of the Customer class
        self._principal_amount = principal_amount
        self._interest_rate = interest_rate
        self._loan_term = loan_term
        self._loan_type = loan_type
        self._loan_status = loan_status
```

```
 def __str__(self):
       return (f"Loan(ID: {self.loan_id}, Customer ID:
{self.get_customer_id()}, "
              f"Type: {self.loan_type}, Status: {self.loan_status}, "
              f"Amount: {self.principal_amount}, Interest Rate:
{self.interest_rate}, "
              f"Term: {self.loan_term})")
    # Get customer ID from the customer object
    def get_customer_id(self):
        return self.customer.customer_id if self.customer else None
    # loan_id property
    @property
    def loan_id(self):
        return self._loan_id
    @loan_id.setter
    def loan_id(self, value):
        self._loan_id = value
    # customer property
    @property
    def customer(self):
        return self._customer
    @customer.setter
    def customer(self, value):
        self._customer = value
    # principal_amount property
    @property
    def principal_amount(self):
        return self._principal_amount
```

```
@principal_amount.setter
def principal_amount(self, value):
    self._principal_amount = value
# interest_rate property
@property
def interest_rate(self):
    return self._interest_rate
@interest_rate.setter
def interest_rate(self, value):
    self._interest_rate = value
# loan_term property
@property
def loan_term(self):
    return self._loan_term
@loan_term.setter
def loan_term(self, value):
    self._loan_term = value
# loan_type property
@property
def loan_type(self):
    return self._loan_type
@loan_type.setter
def loan_type(self, value):
    self._loan_type = value
# loan_status property
@property
def loan_status(self):
    return self._loan_status
```

```
        @loan_status.setter
       def loan_status(self, value):
         self._loan_status = value
```

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.

HOME LOAN CODE:

```
import sys

sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN
MANAGEMENT SYSTEM")

from entity.Loan import Loan

class HomeLoan(Loan):

    def __init__(self, loan_id=None, customer=None,
principal_amount=None, interest_rate=None, loan_term=None,
property_address=None, property_value=None,
loan_status="Pending"):

        super().__init__(loan_id, customer, principal_amount,
interest_rate, loan_term, "HomeLoan", loan_status)

        self.property_address = property_address

        self.property_value = property_value

    def __str__(self):

        return super().__str__() + f", Property Address:
```

```
{self.property_address}, Property Value: {self.property_value}"
```
# Getters and Setters

```
    def get_property_address(self):
        return self.property_address
    def set_property_address(self, property_address):
        self.property_address = property_address
    def get_property_value(self):
        return self.property_value
    def set_property_value(self, property_value):
        self.property_value = property_value
```

CAR LOAN CODE:

```
import sys
sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN
MANAGEMENT SYSTEM")
from entity.Loan import Loan
class CarLoan(Loan):
    def __init__(self, loan_id, customer, principal_amount,
interest_rate, loan_term, car_model, car_value,
loan_status="Pending"):
        super().__init__(loan_id, customer, principal_amount,
interest_rate, loan_term, "CarLoan", loan_status)
        self.car_model = car_model
        self.car_value = car_value
    def __str__(self):
        return super().__str__() + f", Car Model: {self.car_model}, Car
Value: {self.car_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.

CUSTOMER CLASS CODE:

```
class Customer:

    def __init__(self, customer_id=None, name=None, email=None,
phone=None, address=None, credit_score=None):

        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}, Name: {self.name},
Email: {self.email}, Phone: {self.phone}, Address: {self.address}, Credit
Score: {self.credit_score})"

    # Getters and Setters

    def get_customer_id(self):

        return self.customer_id

    def set_customer_id(self, customer_id):

        self.customer_id = customer_id

    def get_name(self):
```

```
    return self.name
def set_name(self, name):
    self.name = name
def get_email(self):
    return self.email
def set_email(self, email):
    self.email = email
def get_phone(self):
    return self.phone
def set_phone(self, phone):
    self.phone = phone
def get_address(self):
    return self.address
def set_address(self, address):
    self.address = address
def get_credit_score(self):
    return self.credit_score
def set_credit_score(self, credit_score):
    self.credit_score = credit_score
```

LOAN CLASS CODE:

```
class Loan:
    def __init__(self, loan_id=None, customer=None,
principal_amount=None, interest_rate=None, loan_term=None,
loan_type=None, loan_status="Pending"):
        self.loan_id = loan_id
        self.customer = customer
        self.principal_amount = principal_amount
        self.interest_rate = interest_rate
        self.loan_term = loan_term
        self.loan_type = loan_type
        self.loan_status = loan_status
    def __str__(self):
         return f"Loan(ID: {self.loan_id}, Type: {self.loan_type}, Status:
 {self.loan_status}, Amount: {self.principal_amount}, Interest Rate:
{self.interest_rate}, Term: {self.loan_term})"
    # Getters and Setters
    def get_loan_id(self):
      return self.loan_id
    def set_loan_id(self, loan_id):
        self.loan_id = loan_id
    def get_customer(self):
        return self.customer
    def set_customer(self, customer):
        self.customer = customer
 def get_principal_amount(self):
    return self.principal_amount
 def set_principal_amount(self, principal_amount):
    self.principal_amount = principal_amount
 def get_interest_rate(self):
    return self.interest_rate

def set_interest_rate(self, interest_rate):
    self.interest_rate = interest_rate
 def get_loan_term(self):
    return self.loan_term
 def set_loan_term(self, loan_term):
    self.loan_term = loan_term
 def get_loan_type(self):
    return self.loan_type
 def set_loan_type(self, loan_type):
    self.loan_type = loan_type
 def get_loan_status(self):
    return self.loan_status
 def set_loan_status(self, loan_status):
    self.loan_status = loan_status
```

CAR LOAN CODE:

```
import sys
sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN MANAGEMENT SYSTEM")
from entity.Loan import Loan
class CarLoan(Loan):
    def __init__(self, loan_id, customer, principal_amount, interest_rate, loan_term, car_model, car_value, loan_status="Pending"):
        super().__init__(loan_id, customer, principal_amount, interest_rate, loan_term, "CarLoan", loan_status)
        self.car_model = car_model
        self.car_value = car_value
    def __str__(self):
        return super().__str__() + f", Car Model: {self.car_model}, Car Value: {self.car_value}"
```

HOME LOAN CODE:

```
import sys
sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN MANAGEMENT SYSTEM")
from entity.Loan import Loan
class HomeLoan(Loan):
    def __init__(self, loan_id=None, customer=None, principal_amount=None, interest_rate=None, loan_term=None,
property_address=None, property_value=None,
loan_status="Pending"):
        super().__init__(loan_id, customer, principal_amount,
interest_rate, loan_term, "HomeLoan", loan_status)
        self.property_address = property_address
        self.property_value = property_value
    def __str__(self):
        return super().__str__() + f", Property Address:
{self.property_address}, Property Value: {self.property_value}"



# Getters and Setters
    def get_property_address(self):
        return self.property_address

    def set_property_address(self, property_address):
        self.property_address = property_address

    def get_property_value(self):
        return self.property_value

    def set_property_value(self, property_value):
        self.property_value = property_value
```

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 loans 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

CODE:

```
from abc import ABC, abstractmethod
class ILoanRepository(ABC):
    @abstractmethod
    def apply_loan(self, loan):
        """Applies for a loan, stores it in the database."""
        pass
    @abstractmethod
    def calculate_interest(self, loan_id):
        """Calculates interest for a loan."""
        pass
    @abstractmethod
    def loan_status(self, loan_id):
        """Checks and updates the loan status based on credit score."""
        pass
    @abstractmethod
    def calculate_emi(self, loan_id):
        """Calculates EMI for the loan."""
        pass
    @abstractmethod
    def loan_repayment(self, loan_id, amount):
        """Processes loan repayment and updates the balance."""
        pass
    @abstractmethod
    def get_all_loans(self):
        """Retrieves all loans."""
        pass
    @abstractmethod
    def get_loan_by_id(self, loan_id):
        """Retrieves a loan by ID."""
        pass
```

CODE:

```
class InvalidLoanException(Exception):
    def __init__(self, message="Invalid loan operation"):
        self.message = message
        super().__init__(self.message)
```

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

LoanRepositoryImpl.py CODE:

```
import sys

sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN
MANAGEMENT SYSTEM")

from dao.ILoanRepository import ILoanRepository

from exception.InvalidLoanException import InvalidLoanException

from entity.Customer import Customer

def apply_loan(self):

    loan_type = input("Enter loan type (HomeLoan/CarLoan): ")

    customer_id = int(input("Enter Customer ID: "))


    # Retrieve actual customer data from your repository or data
structure

    customer = self.loan_repo.get_customer_by_id(customer_id)  #
Fetch customer data


    if not customer:

        print("Customer not found!")

        return


    principal_amount = float(input("Enter principal amount: "))

    interest_rate = float(input("Enter interest rate: "))

    loan_term = int(input("Enter loan term (months): "))


    if loan_type.lower() == "homeloan":

        property_address = input("Enter property address: ")
        property_value = float(input("Enter property value: "))
        loan = HomeLoan(0, customer, principal_amount, interest_rate,
loan_term, property_address, property_value)
    elif loan_type.lower() == "carloan":
        car_model = input("Enter car model: ")
        car_value = float(input("Enter car value: "))
        loan = CarLoan(0, customer, principal_amount, interest_rate,
loan_term, car_model, car_value)
    else:
        print("Invalid loan type.")
        return

    self.loan_repo.apply_loan(loan)
class LoanRepositoryImpl(ILoanRepository):
    def __init__(self):
        self.loans = {}  # Simulating a database with a dictionary
        self.customers = {
            1: Customer(1, 'Rahul Sharma', 'rahul.sharma@example.com',
'9876543210', '1st Main, Bangalore', 720),
            2: Customer(2, 'Anita Desai', 'anita.desai@example.com',
'8765432109', '2nd Cross, Mumbai', 680),
            3: Customer(3, 'Vikram Singh', 'vikram.singh@example.com',
'7654321098', '3rd Street, Delhi', 750),
            4: Customer(4, 'Priya Iyer', 'priya.iyer@example.com',
'6543210987', '4th Lane, Chennai', 800),
            5: Customer(5, 'Ravi Patel', 'ravi.patel@example.com',
'5432109876', '5th Road, Ahmedabad', 660),
            6: Customer(6, 'Sneha Nair', 'sneha.nair@example.com',
'4321098765', '6th Avenue, Pune', 690),
            7: Customer(7, 'Karan Mehta', 'karan.mehta@example.com',
'3210987654', '7th Circle, Kolkata', 740),
            8: Customer(8, 'Simran Kaur', 'simran.kaur@example.com',
'2109876543', '8th Block, Hyderabad', 710),
            9: Customer(9, 'Aditya Roy', 'aditya.roy@example.com',
'1098765432', '9th Path, Jaipur', 760),
            10: Customer(10, 'Pooja Gupta', 'pooja.gupta@example.com',
'0987654321', '10th Lane, Chandigarh', 780)
        }
        self.next_loan_id = 1  # Initialize loan ID counter

    def get_customer_by_id(self, customer_id):
        return self.customers.get(customer_id)

    def apply_loan(self, loan):
        confirmation = input("Do you want to apply for this loan?
(Yes/No): ")
        if confirmation.lower() == 'yes':
            loan.loan_id = self.next_loan_id  # Assign the current loan ID
            loan.loan_status = "Pending"
            self.loans[loan.loan_id] = loan
            print(f"Loan application submitted with ID: {loan.loan_id}")
            self.next_loan_id += 1  # Increment for the next loan ID
        else:
            print("Loan application canceled.")

    def get_customer_by_id(self, customer_id):
        # Replace this with actual customer retrieval logic
        return self.customers.get(customer_id)
    def calculate_interest(self, loan_id):
        loan = self.get_loan_by_id(loan_id)
        if loan:
            interest = (loan.principal_amount * loan.interest_rate *
loan.loan_term) / 12
            return interest
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")

    def check_and_update_loan_status(self, loan_id):
        loan = self.get_loan_by_id(loan_id)
        if loan:
            if loan.customer.credit_score > 650:
                loan.loan_status = "Approved"
            else:
                loan.loan_status = "Rejected"
            print(f"Loan status for ID {loan_id}: {loan.loan_status}")
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")

    def get_loan_status(self, loan_id):
        loan = self.get_loan_by_id(loan_id)
        if loan:
            return loan.loan_status
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")
```

```
    def calculate_emi(self, loan_id):
        loan = self.get_loan_by_id(loan_id)
        if loan:
            P = loan.principal_amount
            R = loan.interest_rate / 12 / 100
            N = loan.loan_term
            emi = (P * R * (1 + R)**N) / ((1 + R)**N - 1)
            return emi
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")

    def loan_repayment(self, loan_id, amount):
        loan = self.get_loan_by_id(loan_id)
        if loan:
            emi = self.calculate_emi(loan_id)
            if amount < emi:
                print("Repayment rejected. Amount is less than the EMI.")
            else:
                print(f"Repayment of {amount} accepted for loan ID
{loan_id}.")
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")

    def get_all_loans(self):
        return list(self.loans.values())
    def get_loan_by_id(self, loan_id):
        loan = self.loans.get(loan_id, None)
        if loan:
            return loan
        else:
            raise InvalidLoanException(f"Loan with ID {loan_id} not
found.")
```

Explanation of the Code:

1. **Class Initialization**: The LoanManagement class initializes the LoanRepositoryImpl.
2. **Menu Display**: The display_menu method shows the available options to the user.
3. **Loan Application**: The apply_loan method gathers loan details and applies for the loan based on the user’s input.
4. **Get All Loans**: The get_all_loans method retrieves and prints all loans from the repository.
5. **Get Loan by ID**: The get_loan_by_id method fetches and prints a specific loan's details.
6. **Loan Repayment**: The loan_repayment method processes repayments for a given loan.
7. **Main Loop**: The run method continuously displays the menu and executes the corresponding methods based on user choices.

7. Create DBUtil class and add the following method.

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

CODE:

```
import pyodbc


class DBUtil:
    @staticmethod
    def getDBConn():
        try:
            # Define the connection string
            conn = pyodbc.connect(
                'Driver={SQL Server};'
                'Server=sahithya;'
                'Database=LoanManagementSystem;'
                'Trusted_Connection=yes;'
            )
            print("Connected Successfully to the database.")
            return conn  # Return the connection object
        except pyodbc.Error as e:
            print("Connection failed with error:", e)
            return None  # Return None if connection fails
```

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."

CODE:

MainModule.py:

```
import sys

sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN
MANAGEMENT SYSTEM")


from dao.LoanRepositoryImpl import LoanRepositoryImpl

from entity.Customer import Customer

from entity.HomeLoan import HomeLoan

from entity.CarLoan import CarLoan

from exception.InvalidLoanException import InvalidLoanException


class LoanManagement:

  def __init__(self):

    self.loan_repo = LoanRepositoryImpl()


  def display_menu(self):

    print("\n--- Loan Management System ---")

    print("1. Apply for Loan")

    print("2. Get All Loans")

    print("3. Get Loan by ID")

    print("4. Loan Repayment")
```

```
        print("5. Calculate EMI")
        print("6. Calculate Interest")
        print("7. Check Loan Status")  # New option for loan status
        print("8. Exit")

    def apply_loan(self):
        loan_type = input("Enter loan type (HomeLoan/CarLoan): ")
        customer_id = int(input("Enter Customer ID: "))
        customer = self.loan_repo.get_customer_by_id(customer_id)
        if not customer:
            print("Customer not found!")
            return
        principal_amount = float(input("Enter principal amount: "))
        interest_rate = float(input("Enter interest rate: "))
        loan_term = int(input("Enter loan term (months): "))
        if loan_type.lower() == "homeloan":
            property_address = input("Enter property address: ")
            property_value = float(input("Enter property value: "))
            loan = HomeLoan(0, customer, principal_amount,
interest_rate, loan_term, property_address, property_value)
        elif loan_type.lower() == "carloan":
            car_model = input("Enter car model: ")
            car_value = float(input("Enter car value: "))
            loan = CarLoan(0, customer, principal_amount, interest_rate,
loan_term, car_model, car_value)
        else:
            print("Invalid loan type.")
            return
        self.loan_repo.apply_loan(loan)

    def get_all_loans(self):
        loans = self.loan_repo.get_all_loans()
        for loan in loans:
            print(loan)

    def get_loan_by_id(self):
        loan_id = int(input("Enter loan ID: "))
        try:
            loan = self.loan_repo.get_loan_by_id(loan_id)
            print(loan)  # Print the loan details
        except InvalidLoanException as e:
            print(e)

    def loan_repayment(self):
        loan_id = int(input("Enter Loan ID: "))
        amount = float(input("Enter repayment amount: "))
        try:
            self.loan_repo.loan_repayment(loan_id, amount)
            print("Repayment processed successfully.")
        except InvalidLoanException as e:
            print(e)

    def calculate_emi(self):
        loan_id = int(input("Enter Loan ID: "))
        try:
            emi = self.loan_repo.calculate_emi(loan_id)
            print(f"EMI for Loan ID {loan_id} is: {emi:.2f}")
        except InvalidLoanException as e:
            print(e)

    def calculate_interest(self):
        loan_id = int(input("Enter Loan ID: "))
        try:
            interest = self.loan_repo.calculate_interest(loan_id)
            print(f"Interest for Loan ID {loan_id} is: {interest:.2f}")
        except InvalidLoanException as e:
            print(e)

    def check_loan_status(self):
        loan_id = int(input("Enter Loan ID: "))
        try:
            loan = self.loan_repo.get_loan_by_id(loan_id)
            print(f"Loan Status for Loan ID {loan_id} is: {loan.loan_status}")
        except InvalidLoanException as e:
            print(e)

    def run(self):
        while True:
            self.display_menu()
            choice = input("Enter your choice: ")

            if choice == '1':
                self.apply_loan()
            elif choice == '2':
                self.get_all_loans()
            elif choice == '3':
                self.get_loan_by_id()
            elif choice == '4':
                self.loan_repayment()
            elif choice == '5':
                self.calculate_emi()
            elif choice == '6':
                self.calculate_interest()
            elif choice == '7':  # Check loan status
                self.check_loan_status()
            elif choice == '8':
                print("Exiting the Loan Management System.")
                break
            else:
                print("Invalid choice. Please try again.")

# Main function to start the system
def main():
    loan_management_system = LoanManagement()
    loan_management_system.run()
```

```
 if __name__ == "__main__":
     main()
```

Main.py CODE:

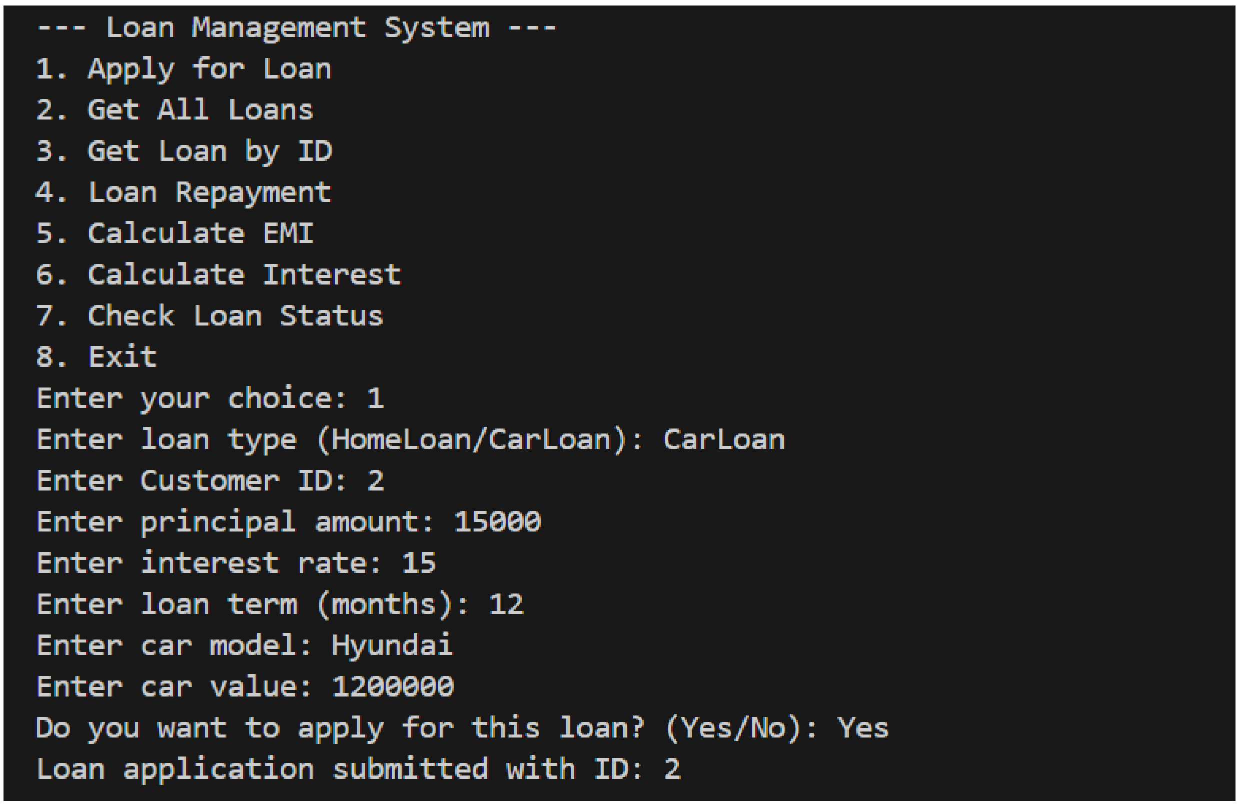
```
import sys
sys.path.append(r"C:\Users\SAHITHYA\OneDrive\Desktop\LOAN
MANAGEMENT SYSTEM")
from dao.LoanRepositoryImpl import LoanRepositoryImpl
from entity.Customer import Customer
from entity.HomeLoan import HomeLoan
from entity.CarLoan import CarLoan
from exception.InvalidLoanException import InvalidLoanException
def menu():
    print("--- Loan Management System ---")
    print("1. Apply for Loan")
    print("2. Get All Loans")
    print("3. Get Loan by ID")
    print("4. Loan Repayment")
    print("5. Calculate EMI")
    print("6. Calculate Interest")
    print("7. Check Loan Status")
    print("8. Exit")
def calculate_emi(loan_repository):
    loan_id = int(input("Enter Loan ID: "))
    try:
        emi = loan_repository.calculate_emi(loan_id)  # Call calculate_emi
method from repository
        print(f"EMI for Loan ID {loan_id} is: {emi:.2f}")  # Display the EMI
```

```
amount
    except InvalidLoanException as e:
        print(e)
class LoanManagement:
    def __init__(self):
        self.loan_repo = LoanRepositoryImpl()
    def apply_loan(self):
        loan_type = input("Enter loan type (HomeLoan/CarLoan): ")
        customer_id = int(input("Enter Customer ID: "))
        customer = Customer(customer_id, "Sample Name",
"email@example.com", "1234567890", "Sample Address", 700)  #
Mock customer data
        principal_amount = float(input("Enter principal amount: "))
        interest_rate = float(input("Enter interest rate: "))
        loan_term = int(input("Enter loan term (months): "))
        if loan_type.lower() == "homeloan":
            property_address = input("Enter property address: ")
            property_value = float(input("Enter property value: "))
            loan = HomeLoan(0, customer, principal_amount, interest_rate,
loan_term, property_address, property_value)
        elif loan_type.lower() == "carloan":
            car_model = input("Enter car model: ")
            car_value = float(input("Enter car value: "))
            loan = CarLoan(0, customer, principal_amount, interest_rate,
loan_term, car_model, car_value)
        else:
            print("Invalid loan type.")
            return
        self.loan_repo.apply_loan(loan)
    def get_all_loans(self):
        loans = self.loan_repo.get_all_loans()
        for loan in loans:
            print(loan)
    def get_loan_by_id(self):
        loan_id = int(input("Enter loan ID: "))
        try:
            loan = self.loan_repo.get_loan_by_id(loan_id)
            print(loan)  # Print the loan details
        except InvalidLoanException as e:
            print(e)


    def loan_repayment(self):
        loan_id = int(input("Enter Loan ID: "))
        amount = float(input("Enter repayment amount: "))
        try:
            self.loan_repo.loan_repayment(loan_id, amount)
            print("Repayment processed successfully.")
        except InvalidLoanException as e:
            print(e)
    def calculate_interest(self):
        loan_id = int(input("Enter Loan ID: "))
        try:
            interest = self.loan_repo.calculate_interest(loan_id)
            print(f"Interest for Loan ID {loan_id} is: {interest:.2f}")
        except InvalidLoanException as e:
            print(e)
    def check_loan_status(self):
        loan_id = int(input("Enter Loan ID: "))
        try:
            loan = self.loan_repo.get_loan_by_id(loan_id)
            print(f"Loan Status for Loan ID {loan_id} is: {loan.loan_status}")
        except InvalidLoanException as e:
            print(e)
    def run(self):
        while True:
            menu()
            choice = input("Enter your choice: ")
            if choice == '1':
                self.apply_loan()
            elif choice == '2':
                self.get_all_loans()
            elif choice == '3':
                self.get_loan_by_id()
            elif choice == '4':
                self.loan_repayment()
            elif choice == '5':
                calculate_emi(self.loan_repo)  # Calculate EMI
            elif choice == '6':
                self.calculate_interest()  # Calculate Interest
            elif choice == '7':
                self.check_loan_status()  # Check Loan Status
            elif choice == '8':
                print("Exiting the Loan Management System.")
                break
            else:
                print("Invalid choice. Please try again.")
def main():
    loan_management_system = LoanManagement()
    loan_management_system.run()

if __name__ == "__main__":
    main()
```

OUTPUT:

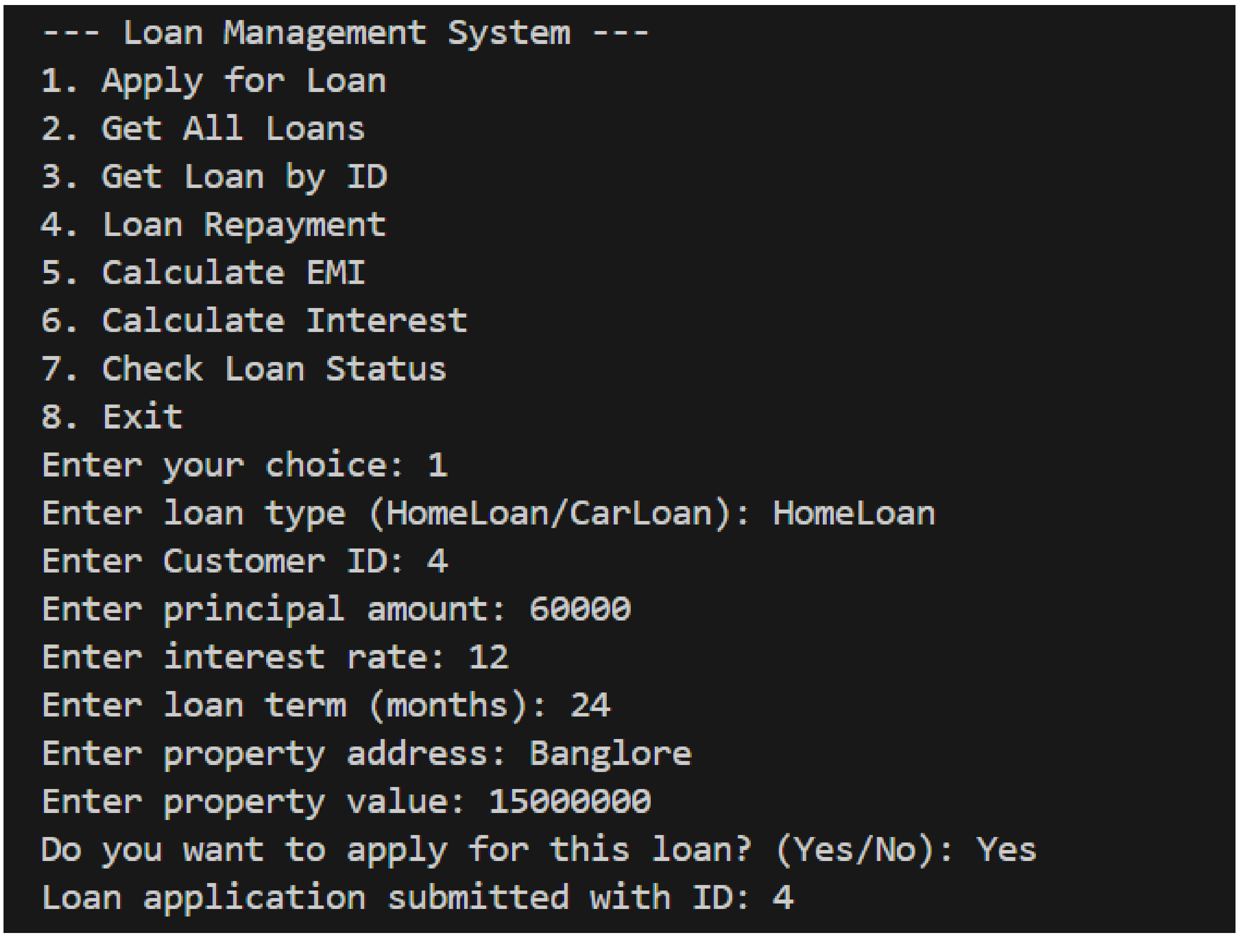
1. APPLYING LOAN:

CarLoan:



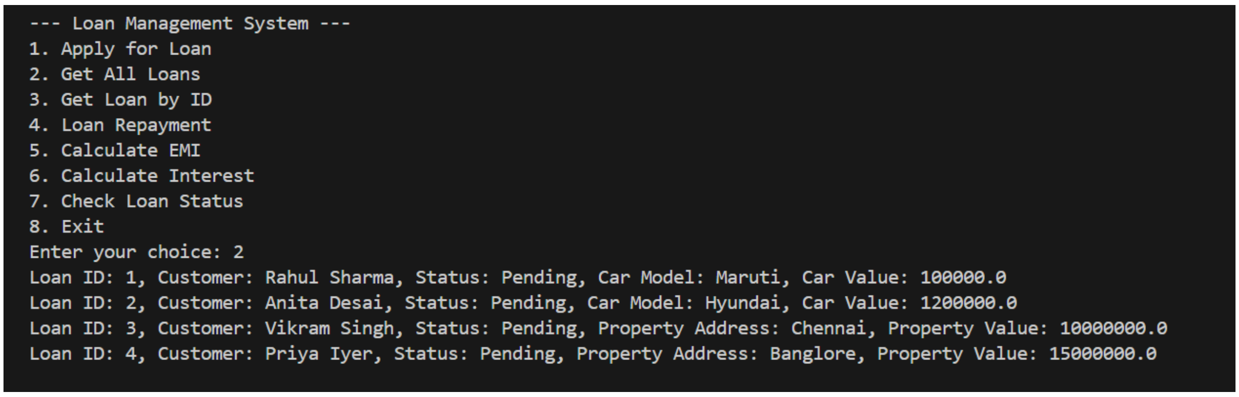
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 1
Enter loan type (HomeLoan/CarLoan): CarLoan
Enter Customer ID: 2
Enter principal amount: 15000
Enter interest rate: 15
Enter loan term (months): 12
Enter car model: Hyundai
Enter car value: 1200000
Do you want to apply for this loan? (Yes/No): Yes
Loan application submitted with ID: 2
```

HomeLoan:



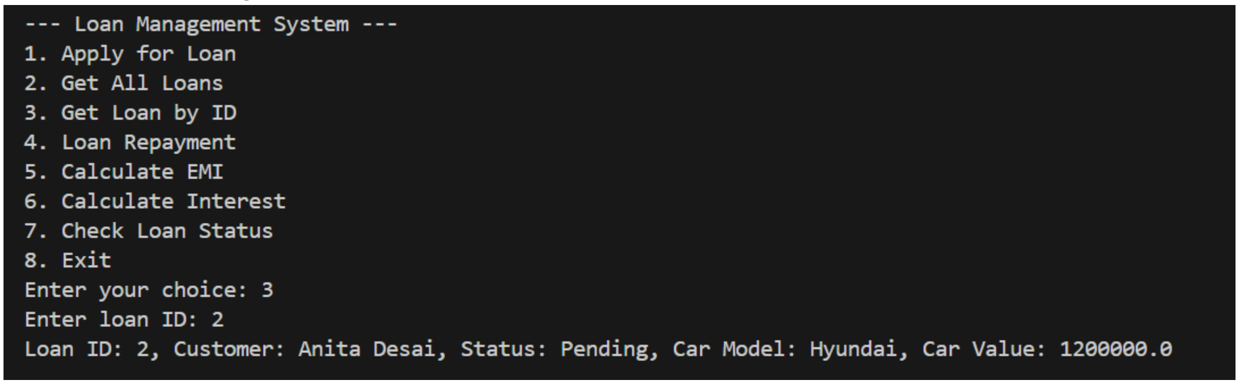
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 1
Enter loan type (HomeLoan/CarLoan): HomeLoan
Enter Customer ID: 4
Enter principal amount: 60000
Enter interest rate: 12
Enter loan term (months): 24
Enter property address: Banglore
Enter property value: 15000000
Do you want to apply for this loan? (Yes/No): Yes
Loan application submitted with ID: 4
```

2. Get all Loans:



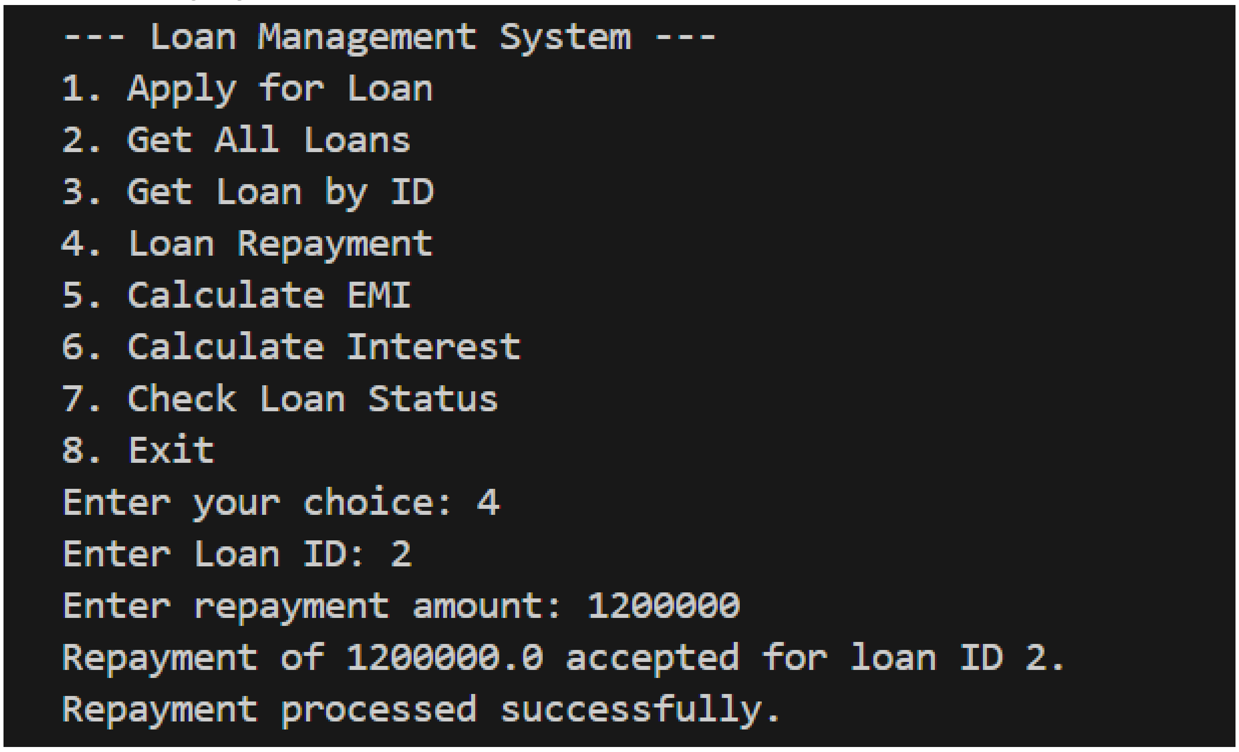
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 2
Loan ID: 1, Customer: Rahul Sharma, Status: Pending, Car Model: Maruti, Car Value: 100000.0
Loan ID: 2, Customer: Anita Desai, Status: Pending, Car Model: Hyundai, Car Value: 1200000.0
Loan ID: 3, Customer: Vikram Singh, Status: Pending, Property Address: Chennai, Property Value: 10000000.0
Loan ID: 4, Customer: Priya Iyer, Status: Pending, Property Address: Banglore, Property Value: 15000000.0
```

3. Get Loan By ID:



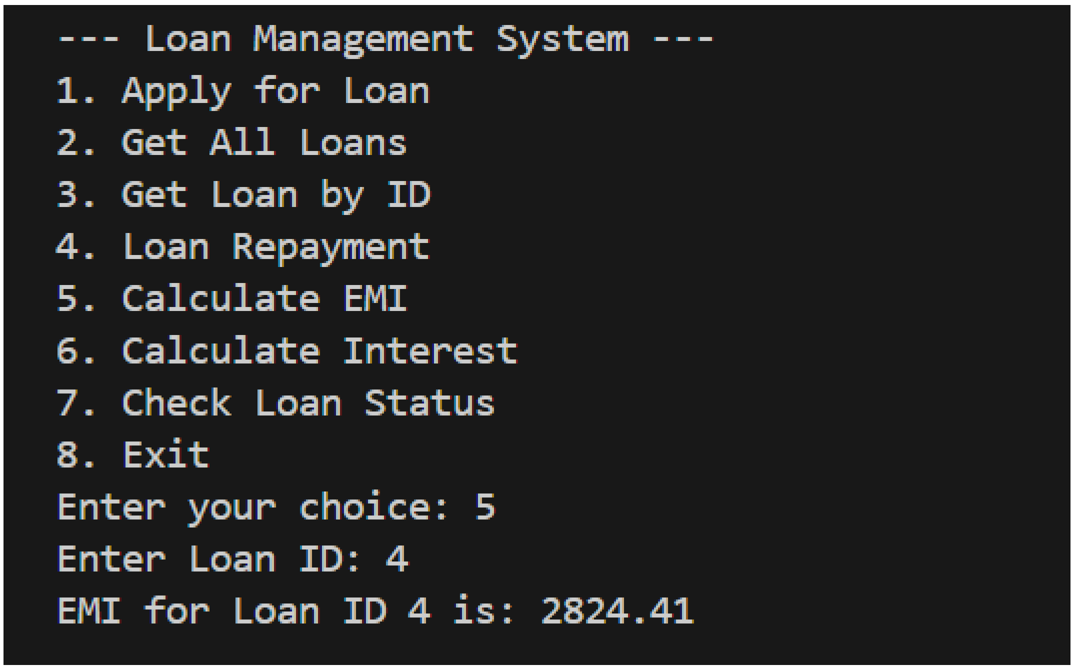
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 3
Enter loan ID: 2
Loan ID: 2, Customer: Anita Desai, Status: Pending, Car Model: Hyundai, Car Value: 1200000.0
```

4. Loan Repayment:



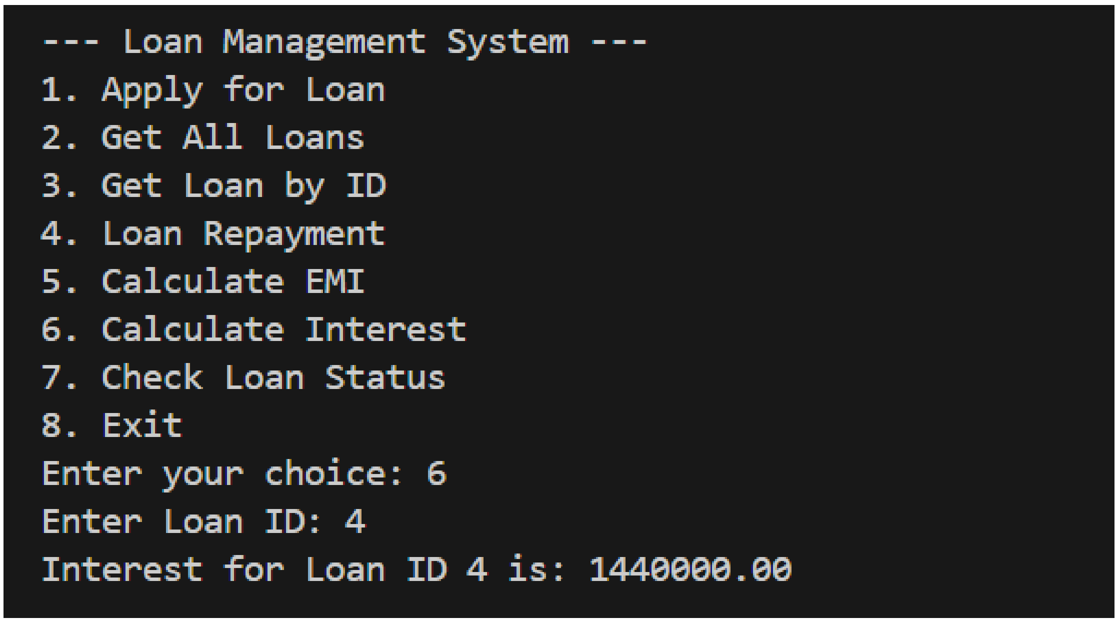
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 4
Enter Loan ID: 2
Enter repayment amount: 1200000
Repayment of 1200000.0 accepted for loan ID 2.
Repayment processed successfully.
```

5.Calculate EMI:



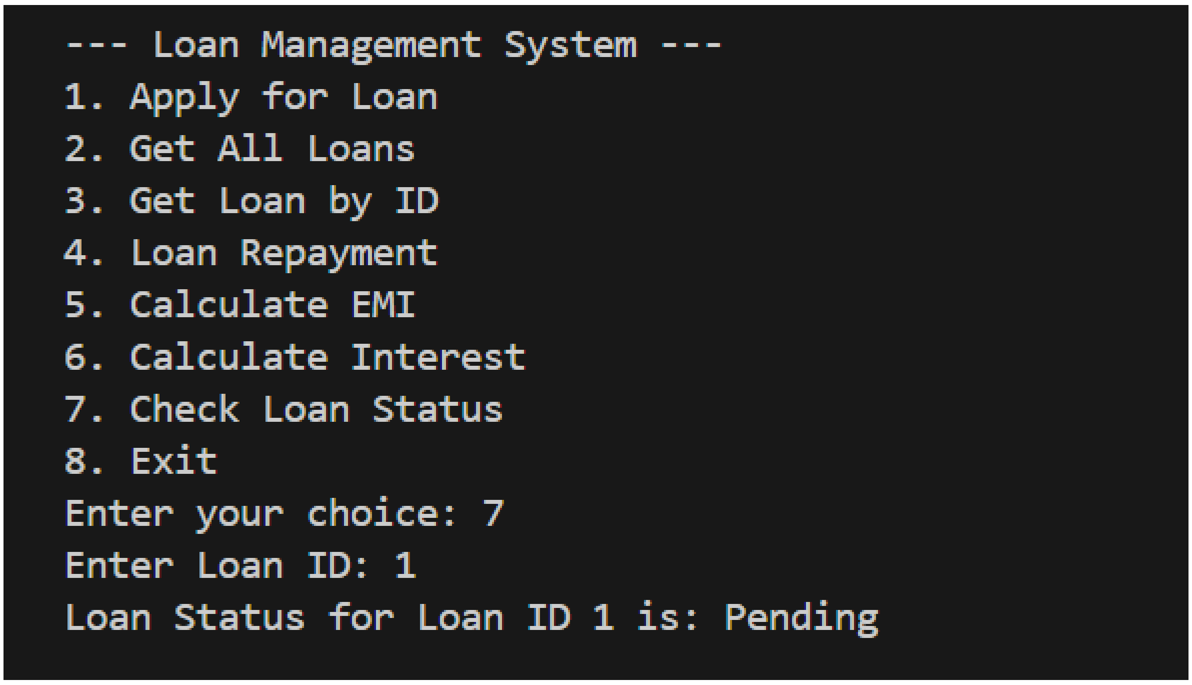
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 5
Enter Loan ID: 4
EMI for Loan ID 4 is: 2824.41
```

6.Calculate Interest:



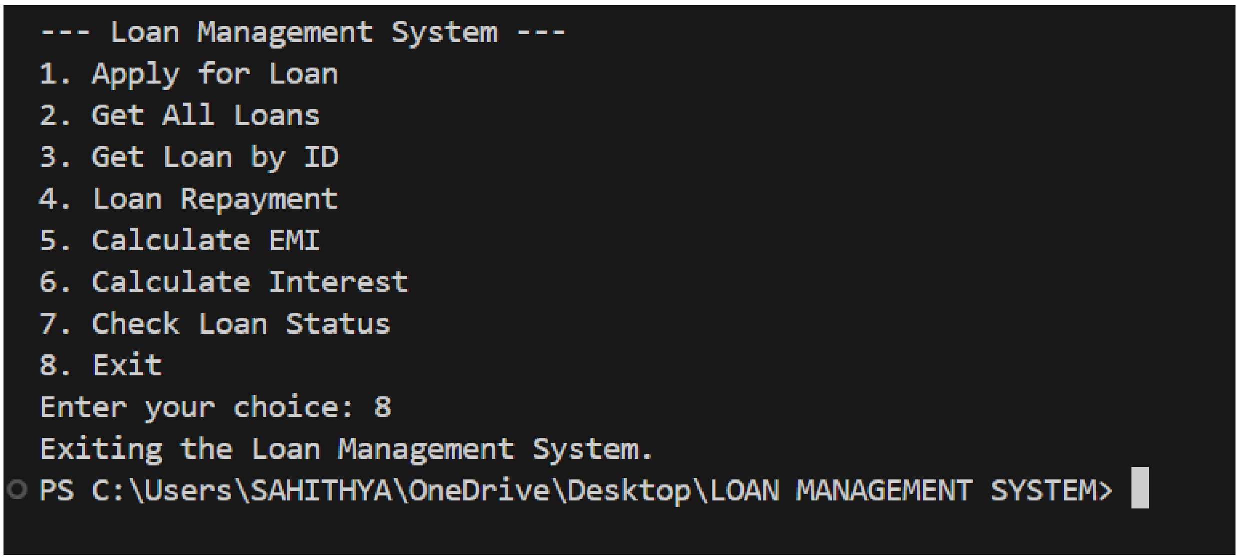
```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 6
Enter Loan ID: 4
Interest for Loan ID 4 is: 1440000.00
```

7.Check Loan Status:



```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 7
Enter Loan ID: 1
Loan Status for Loan ID 1 is: Pending
```

8. Exit:



```
--- Loan Management System ---
1. Apply for Loan
2. Get All Loans
3. Get Loan by ID
4. Loan Repayment
5. Calculate EMI
6. Calculate Interest
7. Check Loan Status
8. Exit
Enter your choice: 8
Exiting the Loan Management System.
PS C:\Users\SAHITHYA\OneDrive\Desktop\LOAN MANAGEMENT SYSTEM>
```