CODING CHALLENGE

Loan Management System

Name: Seemalamudi Nikhil Sai

Create SQL Schema from the customer and loan class.

```
use Loan_management_system;
CREATE TABLE Customer (
    customer_id INT IDENTITY(1,1) PRIMARY KEY ,
    name VARCHAR(50),
    email VARCHAR(60),
    phone_number VARCHAR(15),
    address VARCHAR(150),
    credit_score INT
);
CREATE TABLE Loan (
    loan_id INT IDENTITY(101,1) PRIMARY KEY ,
    customer_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) on
DELETE CASCADE,
    principal_amount Decimal(10, 2),
    interest_rate INT,
    loan_term_months INT,
    loan_type VARCHAR(20),
    loan_status VARCHAR(20)
);


INSERT INTO Customer (name, email , phone_number, address, credit_score)
VALUES
     ('Nikhil', 'nikhil@gmail.com', '+91 8458393088', '123 Happy St, Hyderabad',
830),
     ('Rajashekar', 'raj@gmail.com', '+91 6974581165', '456 Park Avenue,
Bengaluru', 850),
    ('Jayakumar', 'jayak@gmail.com', '+91 7035024126', '789 Diamond Road,
Pondicherry', 900),
    ('Meghana', 'megana@gmail.com', '+91 9443342088', '11 Olive Garden,
Chennai', 850),
    ('Vishal', 'vishal@gmail.com', '+91 9975325173', '23 Sam Street, Chittoor',
750),
    ('Rahul', 'rahul@gmail.com', '+91 8769362439', '57 Park Street, Chennai', 880),
    ('Ajay', 'ajay@gmail.com', '+91 7095392498', '800 Hyung Avenue, Vijayawada',
790),
    ('Kumar', 'kumar@gmail.com', '+91 8723402109', '345 Platinum Lane, Delhi',
900),
    ('Mahesh', 'mahesh@gmail.com', '+91 7953488348', '6 V Avenue, Vizag', 650),
    ('Jayanth', 'jayanth@gmail.com', '+91 8799383759', '12 Ghost Street, Mumbai',
580);

INSERT INTO Loan (customer_id, principal_amount, interest_rate,
loan_term_months, loan_type, loan_status) VALUES
    (7, 500000, 8, 36, 'CarLoan', 'Approved'),
    (1, 1000000, 7, 60, 'HomeLoan', 'Pending'),
    (5, 300000, 9, 24, 'CarLoan', 'Approved'),
    (6, 800000, 6, 48, 'HomeLoan', 'Approved'),
    (4, 700000, 8, 36, 'CarLoan', 'Approved'),
    (3, 1200000, 7, 60, 'HomeLoan', 'Pending'),
    (2, 400000, 9, 24, 'CarLoan', 'Approved'),
    (9, 900000, 6, 48, 'HomeLoan', 'Pending'),
    (8, 600000, 8, 36, 'CarLoan', 'Pending'),
    (10, 1100000, 7, 60, 'HomeLoan', 'Pending');

alter table Loan add property_value int null,property_address varchar(255)
null,car_value int null,car_model varchar(255)null;
alter table Loan add no_emi int null;

select * from Customer;
select * from Loan
ALTER TABLE Loan ADD NoOfEMI INT DEFAULT 0;
```

1. **Define a `Customer` class with the following confidential attributes with all getters and setter:**
   a. Customer ID
   b. Name
   c. Email Address
   d. Phone Number
   e. Address
   f. creditScore

   Code:

```
class Customer:
    def __init__(self, customer_id, name, email, phone_number, address,
credit_score):
        self.customer_id = customer_id
        self.name = name
        self.email = email
        self.phone_number = phone_number
        self.address = address
        self.credit_score = credit_score

    def get_customer_id(self):
        return self.customer_id
    def get_name(self):
        return self.name
    def get_email(self):
        return self.email
    def get_phone_number(self):
        return self.phone_number
    def get_address(self):
        return self.address
    def get_credit_score(self):
        return self.credit_score

    def set_customer_id(self, customer_id):
        self.customer_id = customer_id
    def set_name(self, name):
        self.name = name
    def set_email(self, email):
        self.email = email
    def set_phone_number(self, phone_number):
            self.phone_number = phone_number
        def set_address(self, address):
            self.address = address
        def set_credit_score(self, credit_score):
            self.credit_score = credit_score

        def __str__(self):
            return (f"Customer ID: {self.customer_id}\nName: {self.name}\nEmail:
    {self.email}\n"
                    f"Phone Number: {self.phone_number}\nAddress:
    {self.address}\nCredit Score: {self.credit_score}")
```

2. **Define a base class `Loan` with the following attributes with all setters and getters:**
   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)

   <u>Code:</u>

```
class Loan:
    def __init__(self, loan_id, customer_id, principal_amount, interest_rate,
loan_term_months, loan_type, loan_status):
        self.loan_id = loan_id
        self.customer_id = customer_id
        self.principal_amount = principal_amount
        self.interest_rate = interest_rate
        self.loan_term_months = loan_term_months
        self.loan_type = loan_type
        self.loan_status = loan_status

    def get_loan_id(self):
        return self.loan_id
    def get_customer_id(self):
        return self.customer_id
    def get_principal_amount(self):
        return self.principal_amount
    def get_interest_rate(self):
        return self.interest_rate
    def get_loan_term_months(self):
        return self.loan_term_months
    def get_loan_type(self):
        return self.loan_type
    def get_loan_status(self):
        return self.loan_status

    def set_loan_id(self, loan_id):
        self.loan_id = loan_id
    def set_customer_id(self, customer_id):
        self.customer_id = customer_id
    def set_principal_amount(self, principal_amount):
        self.principal_amount = principal_amount
    def set_interest_rate(self, interest_rate):
        self.interest_rate = interest_rate
    def set_loan_term_months(self, loan_term_months):
        self.loan_term_months = loan_term_months
    def set_loan_type(self, loan_type):
        self.loan_type = loan_type
    def set_loan_status(self, loan_status):
        self.loan_status = loan_status
```

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**.

<u>Code:</u>

```
class HomeLoan(Loan):

    def __init__(self, loan_id, customer_id, principal_amount, interest_rate,
    loan_term_months, property_address, property_value):
        super().__init__(loan_id, customer_id, principal_amount, interest_rate,
    loan_term_months, "HomeLoan", "Pending")
        self.property_address = property_address
        self.property_value = property_value

    def get_property_address(self):
        return self.property_address
    def get_property_value(self):
        return self.property_value

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

class CarLoan(Loan):
    def __init__(self, loan_id, customer_id, principal_amount, interest_rate,
    loan_term_months, car_model, car_value):
        super().__init__(loan_id, customer_id, principal_amount, interest_rate,
    loan_term_months, "CarLoan", "Pending")
        self.car_model = car_model
        self.car_value = car_value
    def get_car_model(self):
        return self.car_model
    def get_car_value(self):
        return self.car_value
    def set_car_model(self, car_model):
        self.car_model = car_model
    def set_car_value(self, car_value):
        self.car_value = car_value
```

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

a. applyLoan(loan Loan)

b. calculateInterest(loanId)

c. loanStatus(loanId)

d. calculateEMI(loanId)

e. loanRepayment(loanId, amount):

f. getAllLoan():

g. getLoanById(loanId):

Code:

```
import sys
sys.path.append(r"C:\Users\nikhi\OneDrive\Pictures\Documents\loan_project_oops
")
from abc import ABC, abstractmethod


class ILoanRepository(ABC):
    @abstractmethod
    def apply_loan(self, loan):
        pass


    @abstractmethod
    def calculate_interest(self, loan_id):
        pass


    @abstractmethod
    def loan_status(self, loan_id):
        pass

    @abstractmethod
    def calculate_emi(self, loan_id):
        pass

    @abstractmethod
    def loan_repayment(self, loan_id, amount):
        pass

    @abstractmethod
    def get_all_loan(self):
        pass

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

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

Code:

```
import sys
sys.path.append(r"C:\Users\nikhi\OneDrive\Pictures\Documents\loan_project_oops")

from exception.invalidloanexception import InvalidLoanException
from util import dbutil
```

```
import pyodbc
from dao.loanrepository import ILoanRepository
class LoanRepositoryImpl(ILoanRepository):
    def __init__(self):
        self.conn = dbutil.DBUtil.get_db_conn()


    def apply_loan(self, loan):
        try:
            print("Do you want to apply for the loan? (Yes/No):")
            choice = input()
            if choice.lower() == "yes":
                cursor = self.conn.cursor()
                if loan.loan_type=='HomeLoan':
                    cursor.execute("INSERT INTO Loan ( customer_id, principal_amount,
    interest_rate, loan_term_months, loan_type,
    loan_status,property_address,property_value) VALUES ( ?, ?, ?, ?, ?, ?,?,?)",
                            ( loan.customer_id, loan.principal_amount, loan.interest_rate,
    loan.loan_term_months,
    loan.loan_status,loan.property_address,loan.property_value))
                self.conn.commit()
                print("HomeLoan applied successfully!")
                if loan.loan_type=='CarLoan':
                    cursor.execute("INSERT INTO Loan ( customer_id, principal_amount,
    interest_rate, loan_term_months, loan_type, loan_status,car_model,car_value)
    VALUES ( ?, ?, ?, ?, ?, ?,?,?)",
                            ( loan.customer_id, loan.principal_amount, loan.interest_rate,
    loan.loan_term_months, loan.loan_type,
    loan.loan_status,loan.car_model,loan.car_value))
                self.conn.commit()
                print("CarLoan applied successfully!")
            else:
                print("Loan application cancelled.")
        except pyodbc.Error as ex:
            print(f"Error applying for the loan: {ex}")


    def calculate_interest(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT principal_amount, interest_rate,
    loan_term_months FROM Loan WHERE loan_id=?", (loan_id,))
            row = cursor.fetchone()
            if row:
                principal_amount, interest_rate, loan_term_months = row
                interest = (principal_amount * interest_rate * loan_term_months) / 12
                return interest
            else:
                raise InvalidLoanException("Loan not found.")
        except pyodbc.Error as ex:
            print(f"Error calculating interest: {ex}")


    def loan_status(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT credit_score FROM Customer WHERE
    customer_id=(SELECT customer_id FROM Loan WHERE loan_id=?)",
    (loan_id,))
            row = cursor.fetchone()
            if row:
                credit_score = row[0]
                if credit_score > 650:
                    cursor.execute("UPDATE Loan SET loan_status='Approved' WHERE
    loan_id=?", (loan_id,))
                    self.conn.commit()
                    print("Congratulations, Loan approved.")
                else:
                    cursor.execute("UPDATE Loan SET loan_status='Rejected' WHERE
    loan_id=?", (loan_id,))
                    self.conn.commit()
                    print("Sorry, Loan rejected due to low credit score.")
            else:
                raise InvalidLoanException("Loan not found.")
        except pyodbc.Error as ex:
            print(f"Error updating loan status: {ex}")


    def calculate_emi(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT principal_amount, interest_rate,
    loan_term_months FROM Loan WHERE loan_id=?", (loan_id,))
            row = cursor.fetchone()
            if row:
                principal_amount, interest_rate, loan_term_months = row
                principal_amount=float(principal_amount)
                interest_rate=float(interest_rate)
                R = interest_rate / 12 / 100
                N = loan_term_months
                emi = (principal_amount * R * (1+R)**N) / ((1+R)**N - 1)
                return emi
            else:
                raise InvalidLoanException("Loan not found.")
        except pyodbc.Error as ex:
            print(f"Error calculating EMI: {ex}")


    def loan_repayment(self, loan_id, amount):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT principal_amount, interest_rate,
    loan_term_months FROM Loan WHERE loan_id=?", (loan_id,))
            row = cursor.fetchone()
            if row:
                principal_amount, interest_rate, loan_term_months = row
                principal_amount=float(principal_amount)
                interest_rate=float(interest_rate)
                R = interest_rate / 12 / 100
                N = loan_term_months
                emi = (principal_amount * R * (1 + R) ** N) / ((1 + R) ** N - 1)
                no_of_emi = amount / float(emi)


                if amount < emi:
                    print("Amount is less than one EMI. Payment rejected.")
                else:
                    cursor.execute("UPDATE Loan SET NoOfEMI=NoOfEMI+? WHERE
    loan_id=?", (int(no_of_emi), loan_id))
                    self.conn.commit()
                    print(f"{int(no_of_emi)} EMI(s) paid successfully.")
            else:
                raise InvalidLoanException("Loan not found.")
        except pyodbc.Error as ex:
            print(f"Error processing loan repayment: {ex}")


    def get_all_loan(self):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT * FROM Loan")
            rows = cursor.fetchall()
            if rows:
                for row in rows:
                    print(f"Loan ID: {row[0]}, Customer ID: {row[1]}, Principal Amount: {row[2]}, Interest Rate: {row[3]}, Loan Term: {row[4]}, Loan Type: {row[5]}, Loan Status: {row[6]}")
            else:
                print("No loans found.")
        except pyodbc.Error as ex:
            print(f"Error fetching loans: {ex}")


    def get_loan_by_id(self, loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT * FROM Loan WHERE loan_id=?", (loan_id,))
            row = cursor.fetchone()
            if row:
                print(f"Loan ID: {row[0]}, Customer ID: {row[1]}, Principal Amount: {row[2]}, Interest Rate: {row[3]}, Loan Term: {row[4]}, Loan Type: {row[5]}, Loan Status: {row[6]}")
            else:
                raise InvalidLoanException("Loan not found.")
        except pyodbc.Error as ex:
            print(f"Error fetching loan: {ex}")


    def customer_details(self,loan_id):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT customer_id, name, phone_number,email, address from Customer WHERE customer_id=?", (loan_id,))
            row = cursor.fetchall()
            if row:
                print(f"{row[0]}")
            else:
                raise InvalidLoanException("Customer Not found")
        except pyodbc.Error as ex:
            print(f"Error fetching Customer:{ex}")



    def updateloancount(self):
        try:
            cursor = self.conn.cursor()
            cursor.execute("SELECT customer_id, name, phone_number from Customer")
            row = cursor.fetchone()
            if row:
                return row[0]
            else:
                return -1

        except pyodbc.Error as ex:
            print("Error")
```

6. 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 get_db_conn():
        """Establishes a connection to the database."""
        try:
            conn = pyodbc.connect('Driver={SQL Server};'
                                  'Server=NIKKYPC\\SQLEXPRESS;'  # Ensure double
    backslash for the server
                                  'Database=Loan_management_system;'
                                  'Trusted_Connection=yes;')
            print("Connection successful")
            return conn
        except pyodbc.Error as ex:
            print(f"Error: {ex}")
            return None
    @staticmethod
    def execute_query(query, params=None):
        """Executes a given SQL query using a cursor."""
        conn = DBUtil.get_db_conn()
        if conn:
            try:
                cursor = conn.cursor()  # Create a cursor object
                if params:
                    cursor.execute(query, params)  # Execute with parameters
                else:
                    cursor.execute(query)  # Execute without parameters
                conn.commit()
                print("Query executed successfully.")
            except pyodbc.Error as ex:
                print(f"Error executing query: {ex}")
            finally:
                cursor.close()
                conn.close()
        else:
            print("Unable to establish connection.")

if __name__ == "__main__":
    db = DBUtil()
    connection=db.get_db_conn()
```

7. Create an exception class module called ‘InvalidLoanException’ that is generated when loan is not found.

Code:

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

8. Create LoanManagement main class and perform following operation in 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:

```
import sys
sys.path.append(r"C:\Users\nikhi\OneDrive\Pictures\Documents\loan_project_oops
")


from entity.loan import CarLoan
from entity.loan import HomeLoan
from entity import loan
from dao.loanrepositoryimpl import LoanRepositoryImpl


from exception.invalidloanexception import InvalidLoanException



class LoanManagement:
    @staticmethod
    def main():
        loan_repo = LoanRepositoryImpl()
        loancount=101
```

```
while True:
    print("\nLoan Management System")
    print("Menu:")
    print("0. Display Customer Details")
    print("1. Apply for Loan")
    print("2. Calculate Interest")
    print("3. Check Loan Status")
    print("4. Calculate EMI")
    print("5. Loan Repayment")
    print("6. Get All Loans")
    print("7. Get Loan by ID")
    print("8. Exit")

    choice = input("Enter your choice: ")

    if choice == "0":
        loan_id=int(input("Enter Your Customer ID: "))
        print("Display Customer Details")
        loan_repo.customer_details(loan_id)

    elif choice == "1":




        # Apply for Loan
        print("\nApplying for Loan:")
        customer_id = input("Enter Customer ID: ")
```

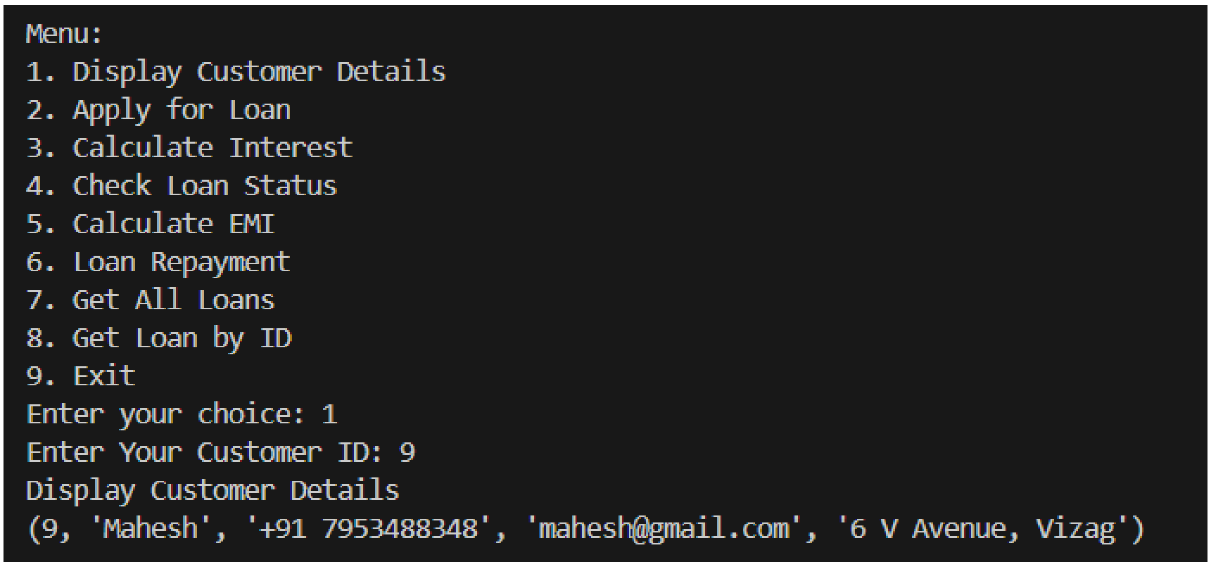
```
            principal_amount = float(input("Enter Principal Amount: "))
            interest_rate = float(input("Enter Interest Rate: "))
            loan_term_months = int(input("Enter Loan Term (in months): "))
            loan_type = input("Enter Loan Type (HomeLoan/CarLoan): ")
            property_address = input("Enter Property Address: ") if loan_type.lower() == "homeloan" else None
            property_value = float(input("Enter Property Value: ")) if loan_type.lower() == "homeloan" else None
            car_model = input("Enter Car Model: ") if loan_type.lower() == "carloan" else None
            car_value = float(input("Enter Car Value: ")) if loan_type.lower() == "carloan" else None
            if loan_type.lower() == "homeloan":
                loan = HomeLoan(str(loancount), customer_id, principal_amount, interest_rate, loan_term_months, property_address, property_value)
                loancount=loancount+1
            elif loan_type.lower() == "carloan":
                loan = CarLoan(str(loancount), customer_id, principal_amount, interest_rate, loan_term_months, car_model, car_value)
                loancount=loancount+1
            else:
                print("Invalid loan type.")
                continue
            loan_repo.apply_loan(loan)

        elif choice == "2":
            # Calculate Interest
            loan_id = input("Enter Loan ID: ")
            interest = loan_repo.calculate_interest(loan_id)
```

```
    print(f"Interest Amount: {interest}")

elif choice == "3":
    # Check Loan Status
    loan_id = input("Enter Loan ID: ")
    loan_repo.loan_status(loan_id)

elif choice == "4":
    # Calculate EMI
    loan_id = input("Enter Loan ID: ")
    emi = loan_repo.calculate_emi(loan_id)
    print(f"EMI Amount: {emi}")

elif choice == "5":
    # Loan Repayment
    loan_id = input("Enter Loan ID: ")
    amount = float(input("Enter Amount for Repayment: "))
    loan_repo.loan_repayment(loan_id, amount)

elif choice == "6":
    # Get All Loans
    loan_repo.get_all_loan()

elif choice == "7":
    # Get Loan by ID
    loan_id = input("Enter Loan ID: ")
    loan_repo.get_loan_by_id(loan_id)
```

```
            elif choice == "8":
                # Exit
                print("Exiting Loan Management System.")
                break

            else:
                print("Invalid choice. Please try again.")

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

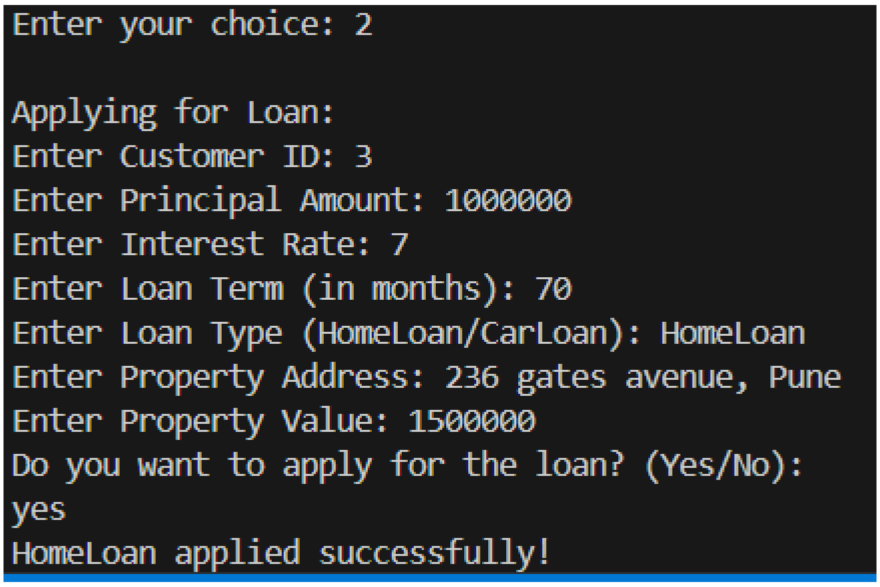
OUTPUTS:

1.Displaying Customer details by customer Id’s: (choice 1)



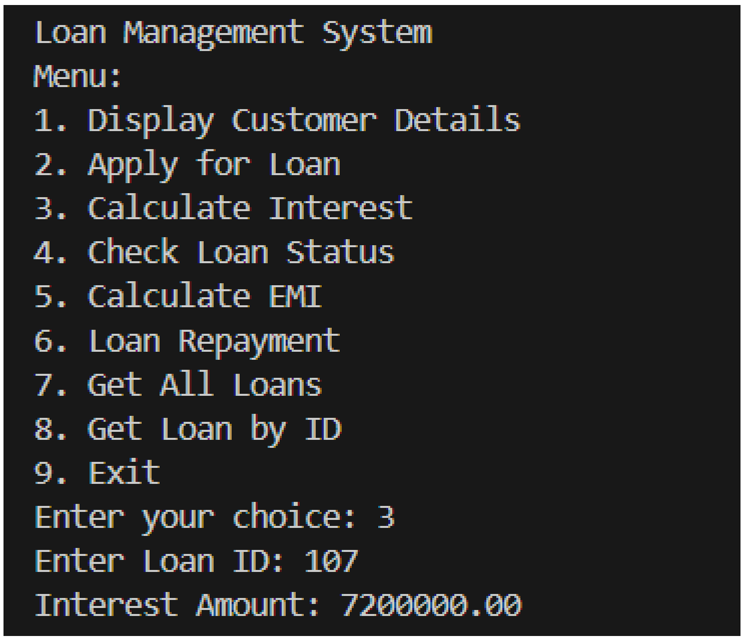
```
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 1
Enter Your Customer ID: 9
Display Customer Details
(9, 'Mahesh', '+91 7953488348', 'mahesh@gmail.com', '6 V Avenue, Vizag')
```

2. Applying for a loan: (Choice 2)



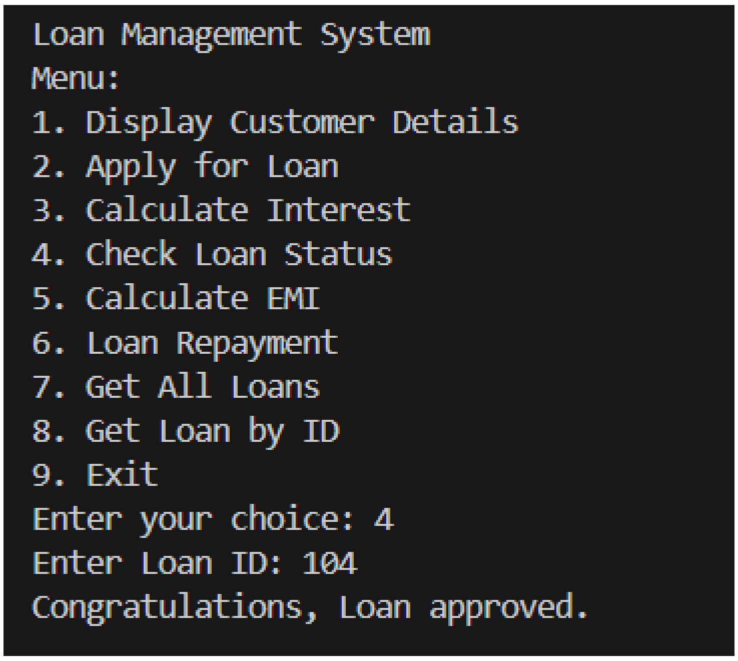
```
Enter your choice: 2

Applying for Loan:
Enter Customer ID: 3
Enter Principal Amount: 1000000
Enter Interest Rate: 7
Enter Loan Term (in months): 70
Enter Loan Type (HomeLoan/CarLoan): HomeLoan
Enter Property Address: 236 gates avenue, Pune
Enter Property Value: 1500000
Do you want to apply for the loan? (Yes/No):
yes
HomeLoan applied successfully!
```

3. Calculating interest amount: (choice 3)



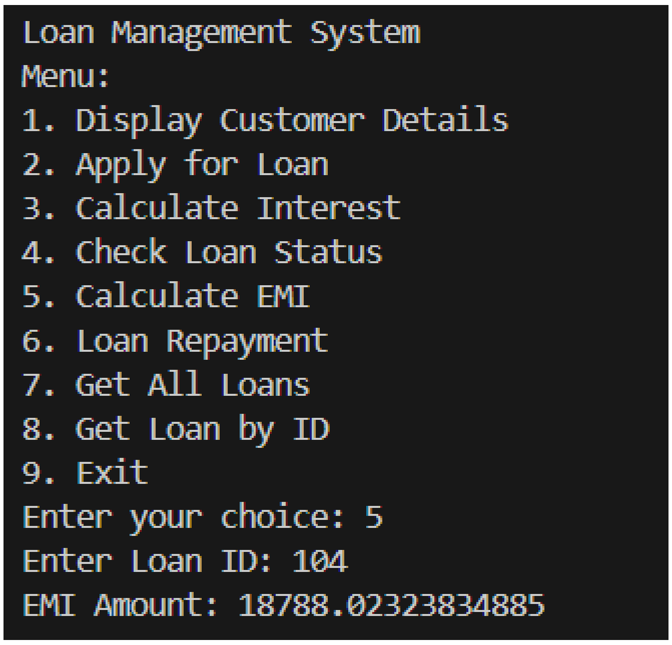
```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 3
Enter Loan ID: 107
Interest Amount: 7200000.00
```

4. Checking loan status: (choice 4)



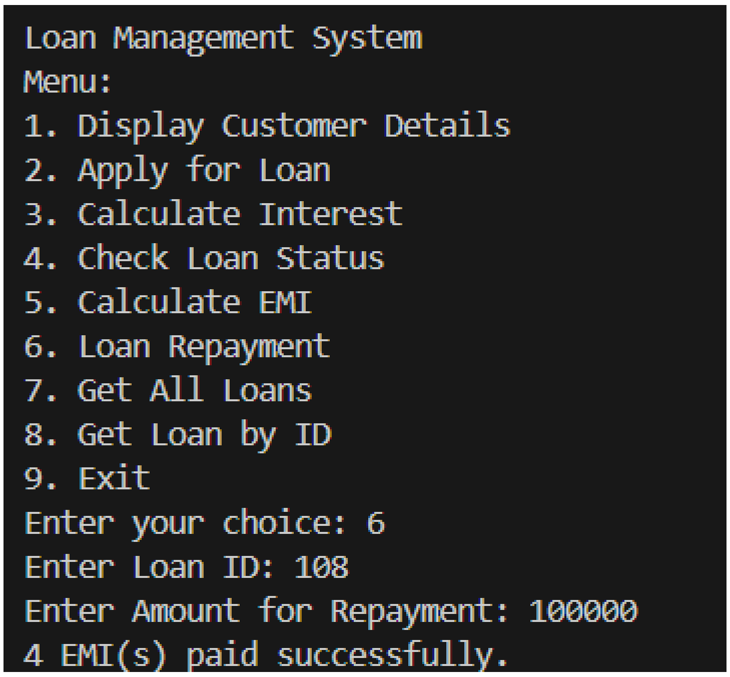
```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 4
Enter Loan ID: 104
Congratulations, Loan approved.
```

5. Calculating EMI: (choice 5)



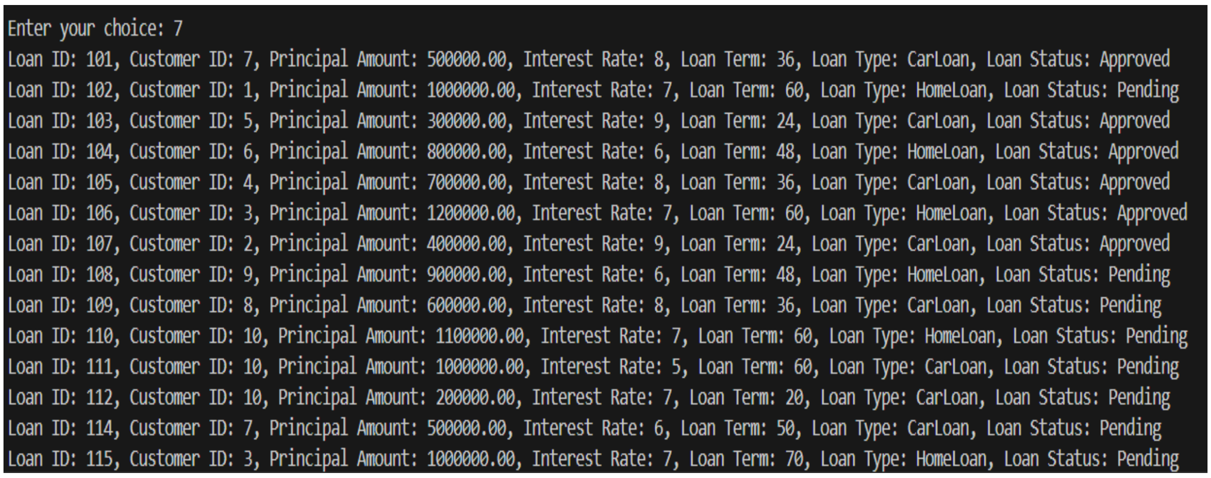
```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 5
Enter Loan ID: 104
EMI Amount: 18788.02323834885
```

6. Loan Repayment:(choice 6)



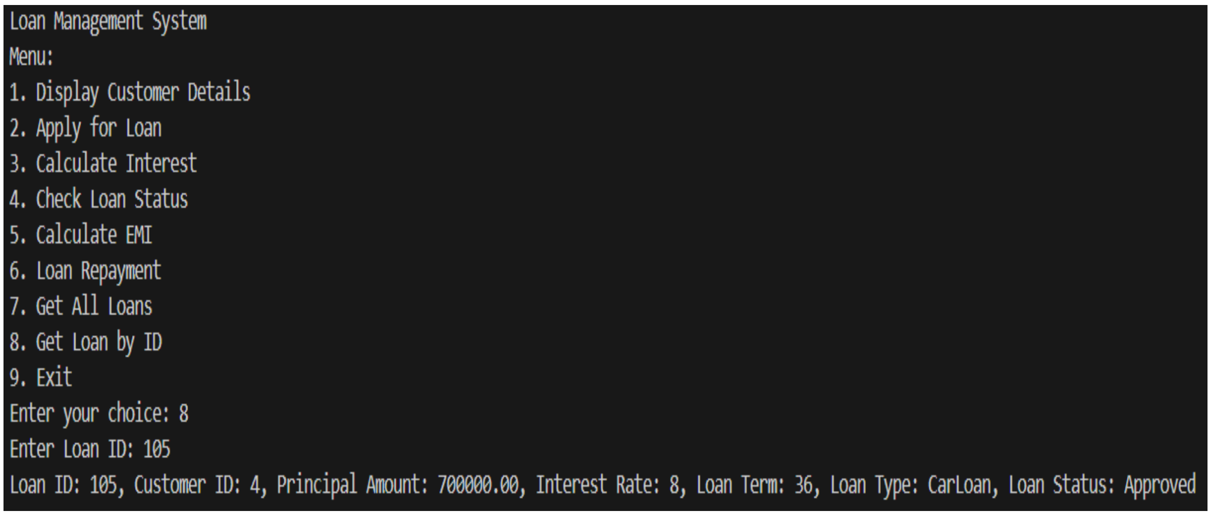
```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 6
Enter Loan ID: 108
Enter Amount for Repayment: 100000
4 EMI(s) paid successfully.
```

7. Getting all loans and their details: (choice 7)



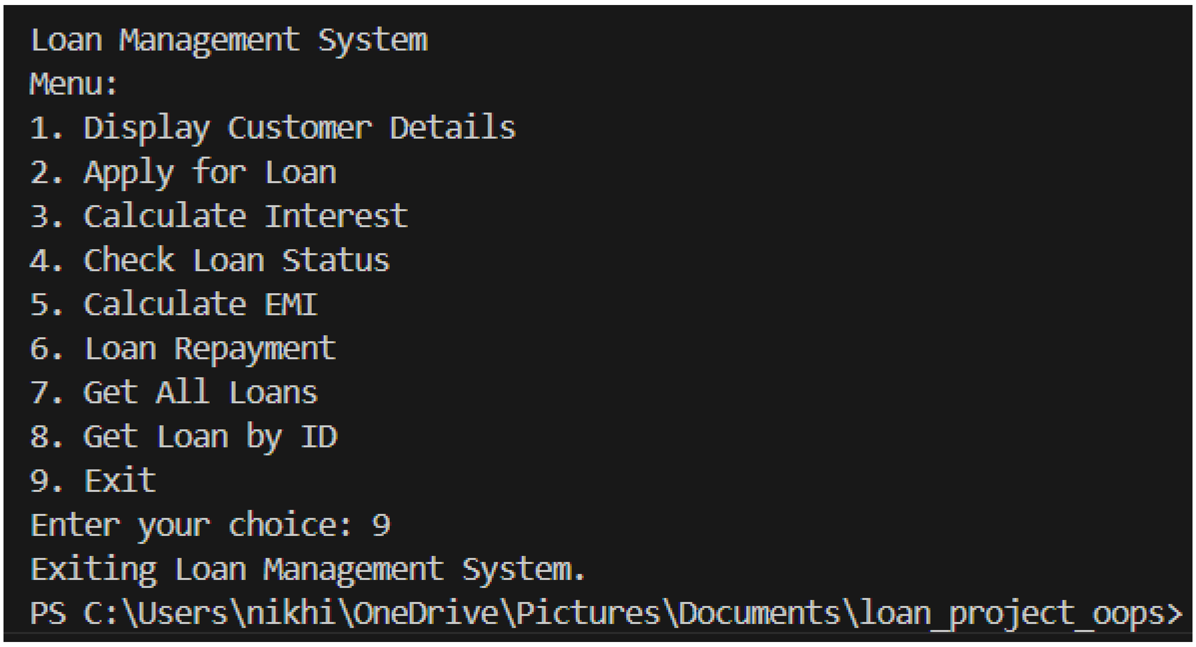
```
Enter your choice: 7
Loan ID: 101, Customer ID: 7, Principal Amount: 500000.00, Interest Rate: 8, Loan Term: 36, Loan Type: CarLoan, Loan Status: Approved
Loan ID: 102, Customer ID: 1, Principal Amount: 1000000.00, Interest Rate: 7, Loan Term: 60, Loan Type: HomeLoan, Loan Status: Pending
Loan ID: 103, Customer ID: 5, Principal Amount: 300000.00, Interest Rate: 9, Loan Term: 24, Loan Type: CarLoan, Loan Status: Approved
Loan ID: 104, Customer ID: 6, Principal Amount: 800000.00, Interest Rate: 6, Loan Term: 48, Loan Type: HomeLoan, Loan Status: Approved
Loan ID: 105, Customer ID: 4, Principal Amount: 700000.00, Interest Rate: 8, Loan Term: 36, Loan Type: CarLoan, Loan Status: Approved
Loan ID: 106, Customer ID: 3, Principal Amount: 1200000.00, Interest Rate: 7, Loan Term: 60, Loan Type: HomeLoan, Loan Status: Approved
Loan ID: 107, Customer ID: 2, Principal Amount: 400000.00, Interest Rate: 9, Loan Term: 24, Loan Type: CarLoan, Loan Status: Approved
Loan ID: 108, Customer ID: 9, Principal Amount: 900000.00, Interest Rate: 6, Loan Term: 48, Loan Type: HomeLoan, Loan Status: Pending
Loan ID: 109, Customer ID: 8, Principal Amount: 600000.00, Interest Rate: 8, Loan Term: 36, Loan Type: CarLoan, Loan Status: Pending
Loan ID: 110, Customer ID: 10, Principal Amount: 1100000.00, Interest Rate: 7, Loan Term: 60, Loan Type: HomeLoan, Loan Status: Pending
Loan ID: 111, Customer ID: 10, Principal Amount: 1000000.00, Interest Rate: 5, Loan Term: 60, Loan Type: CarLoan, Loan Status: Pending
Loan ID: 112, Customer ID: 10, Principal Amount: 200000.00, Interest Rate: 7, Loan Term: 20, Loan Type: CarLoan, Loan Status: Pending
Loan ID: 114, Customer ID: 7, Principal Amount: 500000.00, Interest Rate: 6, Loan Term: 50, Loan Type: CarLoan, Loan Status: Pending
Loan ID: 115, Customer ID: 3, Principal Amount: 1000000.00, Interest Rate: 7, Loan Term: 70, Loan Type: HomeLoan, Loan Status: Pending
```

8. Getting loan details by loan Id: (choice 8)



```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 8
Enter Loan ID: 105
Loan ID: 105, Customer ID: 4, Principal Amount: 700000.00, Interest Rate: 8, Loan Term: 36, Loan Type: CarLoan, Loan Status: Approved
```

9. Exit from interface: (choice 9)



```
Loan Management System
Menu:
1. Display Customer Details
2. Apply for Loan
3. Calculate Interest
4. Check Loan Status
5. Calculate EMI
6. Loan Repayment
7. Get All Loans
8. Get Loan by ID
9. Exit
Enter your choice: 9
Exiting Loan Management System.
PS C:\Users\nikhi\OneDrive\Pictures\Documents\loan_project_oops>
```