Manav Rachna International Institute of Research and Studies

SCHOOL OF COMPUTER APPLICATION

MCA - B 1st SEM

PROJECT REPORT

(Course Code: 6.0CA102C01H)

SUBMITTED BY:

Nitesh Kumar 25/SCA/MCAN/075

Shreya 25/SCA/MCAN/074

Akriti 25/SCA/MCAN/076

SUBMITTED TO:

Dr. Ritu Sachdeva

Nitesh Kumar

Shreya

Akriti

∇ System that Automates loan eligibility checking, interest estimation, and repayment scheduling.

Code here: -

```
import java.util.*;
import java.text.DecimalFormat;

/*
 * LoanSystem.java
 * A simple console-based Java program that automates:
 *  - loan eligibility checking
 *  - interest estimation
 *  - repayment scheduling (amortization / EMI schedule)
 *
 * Designed to run in VS Code (or any Java-capable editor/terminal).
 * Single-file project for easy testing. Contains multiple non-public classes
 * and a public LoanSystem class with main().
 *
 * Features:
 *  - Basic eligibility rules based on income, credit score and existing EMI
 *  - Interest rate suggestion based on credit score and tenure
 *  - EMI calculation (standard formula) and amortization schedule
 *  - Option to export schedule to CSV (saved in working directory)
 *
 * Note: This is an educational example — replace/extend rules with real
 * bank policies and validations for production use.
 */

public class LoanSystem {
    private static final Scanner scanner = new Scanner(System.in);
    private static final DecimalFormat df = new DecimalFormat("#.##");

    public static void main(String[] args) {
        System.out.println("=== Loan Automation System (Console) ===\n");

        while (true) {
            System.out.println("Menu:");
            System.out.println("1) Enter applicant data");
            System.out.println("2) Check eligibility & simulate loan");
            System.out.println("3) Exit");
            System.out.print("Choose an option: ");

            String choice = scanner.nextLine().trim();
```

```
            switch (choice) {
                case "1":
                    ApplicantStore.promptAndStoreApplicant();
                    break;
                case "2":
                    Applicant a = ApplicantStore.getApplicantOrPrompt();
                    simulateLoanFlow(a);
                    break;
                case "3":
                    System.out.println("Goodbye!");
                    return;
                default:
                    System.out.println("Invalid option. Try again.\n");
            }
        }
    }

    private static void simulateLoanFlow(Applicant app) {
        System.out.println("\n-- Applicant summary --");
        System.out.println(app);

        // Eligibility check
        EligibilityResult er = EligibilityChecker.checkEligibility(app);
        System.out.println("\nEligibility: " + (er.isEligible ? "ELIGIBLE" : "NOT ELIGIBLE"));
        System.out.println("Reason: " + er.message);

        if (!er.isEligible) {
            System.out.println("Cannot simulate loan since applicant is not eligible.\n");
            return;
        }

        // Input loan parameters
        System.out.print("Enter desired loan amount (principal) in INR: ");
        double principal = readPositiveDouble();

        System.out.print("Enter tenure in years: ");
        int years = readPositiveInt();

        // Suggest interest rate based on credit score and tenure
        double suggestedRate = InterestSuggester.suggestRate(app.creditScore, years);
        System.out.println("Suggested annual interest rate: " + df.format(suggestedRate) + "% per
annum");

        System.out.print("Accept suggested rate? (Y/n): ");
        String accept = scanner.nextLine().trim();
        double annualRate;
```

```
        if (accept.equalsIgnoreCase("n")) {
            System.out.print("Enter annual interest rate (%) to use: ");
            annualRate = readPositiveDouble();
        } else {
            annualRate = suggestedRate;
        }

        System.out.print("Choose repayment type - 1) EMI (fixed)  2) Bullet (interest only with principal
at end) : ");
        String repType = scanner.nextLine().trim();

        LoanDetails loan = new LoanDetails(principal, annualRate, years, repType.equals("2") ?
LoanType.BULLET : LoanType.EMI);

        // Compute
        if (loan.type == LoanType.EMI) {
            double emi = LoanCalculator.calculateEMI(loan.principal, loan.annualRatePercent,
loan.tenureYears);
            System.out.println("\nEstimated monthly EMI: INR " + df.format(emi));
            System.out.println("Total payment (principal + interest): INR " + df.format(emi *
loan.tenureYears * 12));

            // Generate amortization schedule
            System.out.print("Generate and save amortization schedule to CSV? (Y/n): ");
            String gen = scanner.nextLine().trim();
            if (!gen.equalsIgnoreCase("n")) {
                List<RepaymentEntry> schedule = RepaymentSchedule.generateAmortizationSchedule(loan);
                String filename = "amortization_schedule.csv";
                CSVExporter.exportScheduleToCSV(schedule, filename);
                System.out.println("Schedule saved to: " + filename);
            }

        } else {
            double monthlyInterest = loan.principal * (loan.annualRatePercent / 100.0) / 12.0;
            System.out.println("\nInterest-only monthly payment: INR " + df.format(monthlyInterest));
            System.out.println("Principal due at end of tenure: INR " + df.format(loan.principal));

            System.out.print("Generate and save payment schedule to CSV? (Y/n): ");
            String gen = scanner.nextLine().trim();
            if (!gen.equalsIgnoreCase("n")) {
                List<RepaymentEntry> schedule = RepaymentSchedule.generateBulletSchedule(loan);
                String filename = "bullet_schedule.csv";
                CSVExporter.exportScheduleToCSV(schedule, filename);
                System.out.println("Schedule saved to: " + filename);
            }
        }
```

Nitesh Kumar

Shreya

Akriti

```
        System.out.println("\nSimulation done.\n");
    }

    private static double readPositiveDouble() {
        while (true) {
            try {
                String s = scanner.nextLine().trim();
                double v = Double.parseDouble(s);
                if (v > 0) return v;
            } catch (Exception e) {
                // fallthrough
            }
            System.out.print("Please enter a positive number: ");
        }
    }

    private static int readPositiveInt() {
        while (true) {
            try {
                String s = scanner.nextLine().trim();
                int v = Integer.parseInt(s);
                if (v > 0) return v;
            } catch (Exception e) {
            }
            System.out.print("Please enter a positive integer: ");
        }
    }
}

// ---------- Supporting classes ----------

class ApplicantStore {
    private static Applicant stored = null;
    private static final Scanner sc = new Scanner(System.in);

    public static void promptAndStoreApplicant() {
        System.out.print("Full name: ");
        String name = sc.nextLine().trim();

        System.out.print("Monthly gross income (INR): ");
        double income = readDouble();

        System.out.print("Existing monthly EMI obligations (INR): ");
        double existingEmi = readDouble();
```

Nitesh Kumar

Shreya

Akriti

```
        System.out.print("Credit score (300 - 900): ");
        int creditScore = readIntInRange(300, 900);

        stored = new Applicant(name, income, existingEmi, creditScore);
        System.out.println("Applicant saved.\n");
    }

    public static Applicant getApplicantOrPrompt() {
        if (stored == null) {
            System.out.println("No stored applicant found. Please enter applicant data first.");
            promptAndStoreApplicant();
        }
        return stored;
    }

    private static double readDouble() {
        while (true) {
            try {
                String s = sc.nextLine().trim();
                double v = Double.parseDouble(s);
                if (v >= 0) return v;
            } catch (Exception e) {}
            System.out.print("Enter a valid non-negative number: ");
        }
    }

    private static int readIntInRange(int lo, int hi) {
        while (true) {
            try {
                String s = sc.nextLine().trim();
                int v = Integer.parseInt(s);
                if (v >= lo && v <= hi) return v;
            } catch (Exception e) {}
            System.out.print("Enter a number between " + lo + " and " + hi + ": ");
        }
    }
}

class Applicant {
    String name;
    double monthlyIncome; // gross
    double existingMonthlyEMI;
    int creditScore; // 300-900

    public Applicant(String name, double monthlyIncome, double existingMonthlyEMI, int creditScore)
{
```

Nitesh Kumar

Shreya

Akriti

```
        this.name = name;
        this.monthlyIncome = monthlyIncome;
        this.existingMonthlyEMI = existingMonthlyEMI;
        this.creditScore = creditScore;
    }

    @Override
    public String toString() {
        return String.format("Name: %s\nMonthly Income: INR %.2f\nExisting EMI: INR %.2f\nCredit
Score: %d",
            name, monthlyIncome, existingMonthlyEMI, creditScore);
    }
}

class EligibilityResult {
    boolean isEligible;
    String message;

    EligibilityResult(boolean isEligible, String message) {
        this.isEligible = isEligible;
        this.message = message;
    }
}

class EligibilityChecker {
    /*
     * Very simple eligibility rules (example):
     *  - Minimum monthly income: INR 15,000
     *  - Maximum Debt-to-Income ratio (DTI): existingEmi + newEmi <= 50% of income
     *  - Minimum credit score: 550 (with better rates if higher)
     */
    public static EligibilityResult checkEligibility(Applicant a) {
        if (a.monthlyIncome < 15000) {
            return new EligibilityResult(false, "Monthly income below minimum required (INR 15,000)");
        }
        if (a.creditScore < 450) {
            return new EligibilityResult(false, "Credit score too low for lending");
        }
        return new EligibilityResult(true, "Meets basic criteria; run loan simulation to confirm DTI for
chosen amount/tenure");
    }
}

class InterestSuggester {
    // Suggest a base rate depending on credit score and tenure (years)
    public static double suggestRate(int creditScore, int tenureYears) {
```

```
        double base = 12.0; // base APR in percent for average customers
        if (creditScore >= 800) base = 8.0;
        else if (creditScore >= 700) base = 9.5;
        else if (creditScore >= 650) base = 10.5;
        else if (creditScore >= 600) base = 11.5;
        else base = 13.5;

        // Slightly higher for longer tenures
        if (tenureYears > 5) base += 0.5;
        if (tenureYears > 10) base += 0.5;

        return base;
    }
}

enum LoanType { EMI, BULLET }

class LoanDetails {
    double principal;
    double annualRatePercent;
    int tenureYears;
    LoanType type;

    LoanDetails(double principal, double annualRatePercent, int tenureYears, LoanType type) {
        this.principal = principal;
        this.annualRatePercent = annualRatePercent;
        this.tenureYears = tenureYears;
        this.type = type;
    }
}

class LoanCalculator {
    // EMI formula:
    // EMI = P * r * (1+r)^n / ((1+r)^n - 1)
    // where r is monthly rate (decimal), n = months
    public static double calculateEMI(double principal, double annualRatePercent, int tenureYears) {
        double monthlyRate = (annualRatePercent / 100.0) / 12.0;
        int n = tenureYears * 12;
        if (monthlyRate == 0) return principal / n;
        double factor = Math.pow(1 + monthlyRate, n);
        double emi = principal * monthlyRate * factor / (factor - 1);
        return emi;
    }
}

class RepaymentEntry {
```

```
    int installmentNumber;
    double openingBalance;
    double emi; // for EMI type
    double principalComponent;
    double interestComponent;
    double closingBalance;

    public String toCSVLine() {
        return String.format(Locale.US, "%d,%.2f,%.2f,%.2f,%.2f,%.2f",
            installmentNumber, openingBalance, emi, principalComponent, interestComponent,
closingBalance);
    }

    @Override
    public String toString() {
        return String.format("%3d | Open: %.2f | EMI: %.2f | Principal: %.2f | Interest: %.2f | Close:
%.2f",
            installmentNumber, openingBalance, emi, principalComponent, interestComponent,
closingBalance);
    }
}

class RepaymentSchedule {
    public static List<RepaymentEntry> generateAmortizationSchedule(LoanDetails loan) {
        List<RepaymentEntry> schedule = new ArrayList<>();
        double monthlyRate = (loan.annualRatePercent / 100.0) / 12.0;
        int n = loan.tenureYears * 12;
        double emi = LoanCalculator.calculateEMI(loan.principal, loan.annualRatePercent,
loan.tenureYears);

        double balance = loan.principal;
        for (int i = 1; i <= n; i++) {
            RepaymentEntry e = new RepaymentEntry();
            e.installmentNumber = i;
            e.openingBalance = balance;
            e.emi = emi;
            double interest = balance * monthlyRate;
            double principalComp = emi - interest;
            balance = balance - principalComp;
            if (balance < 1e-8) balance = 0;
            e.interestComponent = interest;
            e.principalComponent = principalComp;
            e.closingBalance = balance;
            schedule.add(e);
        }
        return schedule;
```

Nitesh Kumar

Shreya

Akriti

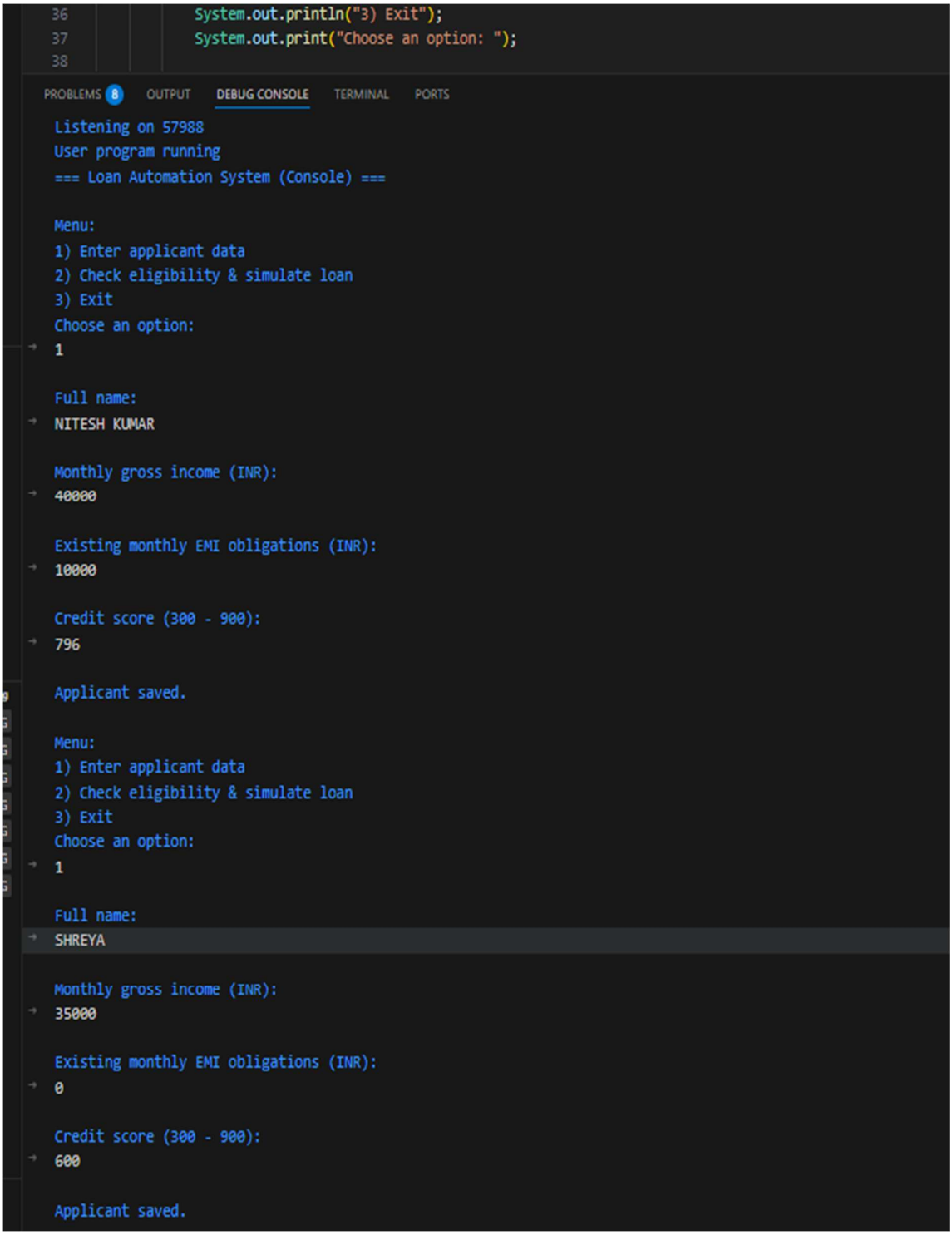
```
    }

    public static List<RepaymentEntry> generateBulletSchedule(LoanDetails loan) {
        List<RepaymentEntry> schedule = new ArrayList<>();
        double monthlyInterest = loan.principal * (loan.annualRatePercent / 100.0) / 12.0;
        int n = loan.tenureYears * 12;
        double balance = loan.principal;

        for (int i = 1; i <= n; i++) {
            RepaymentEntry e = new RepaymentEntry();
            e.installmentNumber = i;
            e.openingBalance = balance;
            e.emi = monthlyInterest; // interest-only monthly
            e.interestComponent = monthlyInterest;
            e.principalComponent = 0;
            e.closingBalance = balance;
            schedule.add(e);
        }
        // final repayment entry to show principal repayment
        RepaymentEntry finalEntry = new RepaymentEntry();
        finalEntry.installmentNumber = n + 1;
        finalEntry.openingBalance = balance;
        finalEntry.emi = balance; // principal paid at end
        finalEntry.interestComponent = 0;
        finalEntry.principalComponent = balance;
        finalEntry.closingBalance = 0;
        schedule.add(finalEntry);
        return schedule;
    }
}

class CSVExporter {
    // Save schedule to CSV in current directory
    public static void exportScheduleToCSV(List<RepaymentEntry> schedule, String filename) {
        try (java.io.PrintWriter pw = new java.io.PrintWriter(new java.io.File(filename))) {
            pw.println("Installment,OpeningBalance,EMI,PrincipalComponent,InterestComponent,ClosingB
alance");
            for (RepaymentEntry e : schedule) pw.println(e.toCSVLine());
        } catch (Exception ex) {
            System.out.println("Failed to write CSV: " + ex.getMessage());
        }
    }
}
```

Nitesh Kumar

Shreya

Akriti

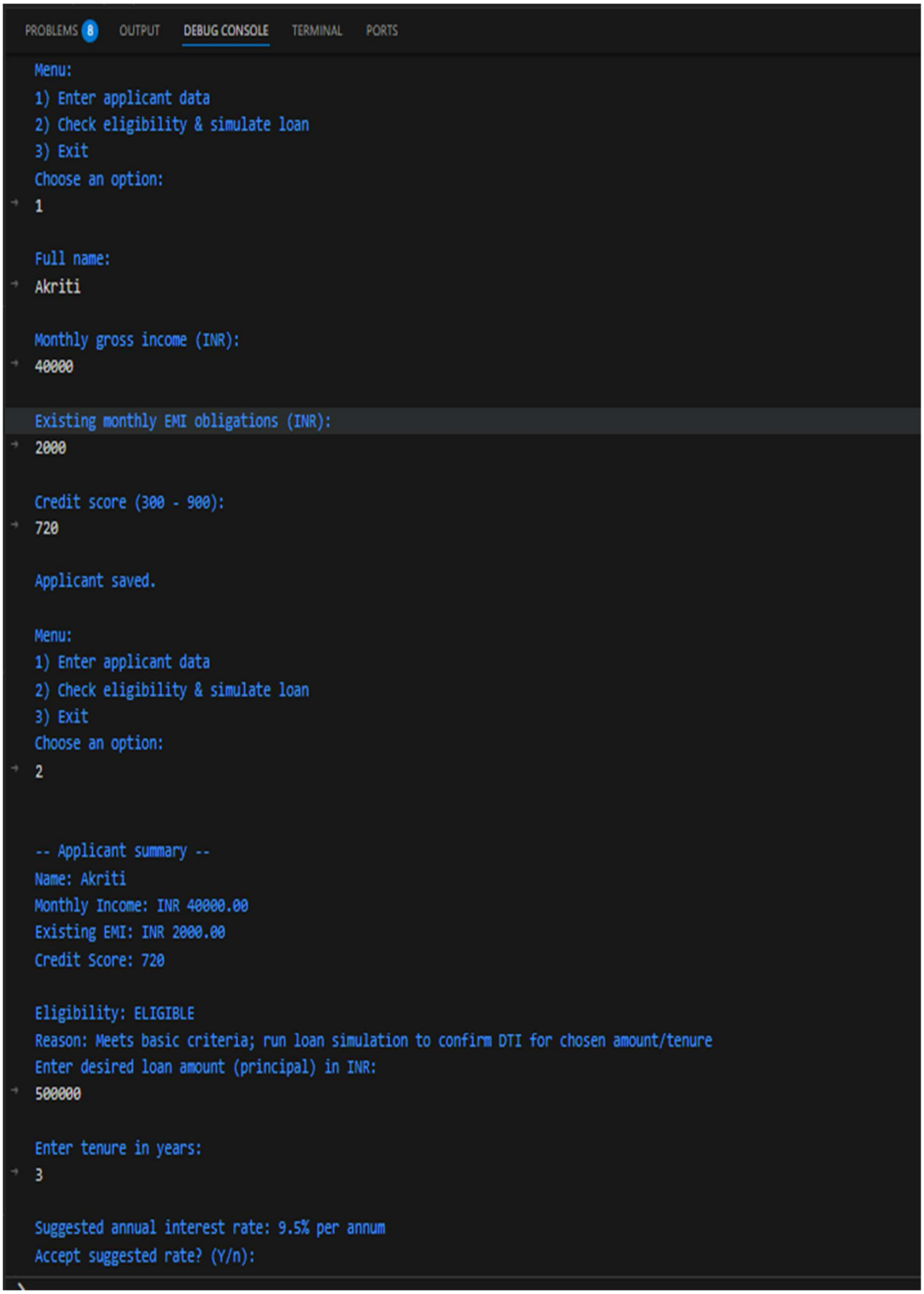
Output screen shot: -



Nitesh Kumar

Shreya

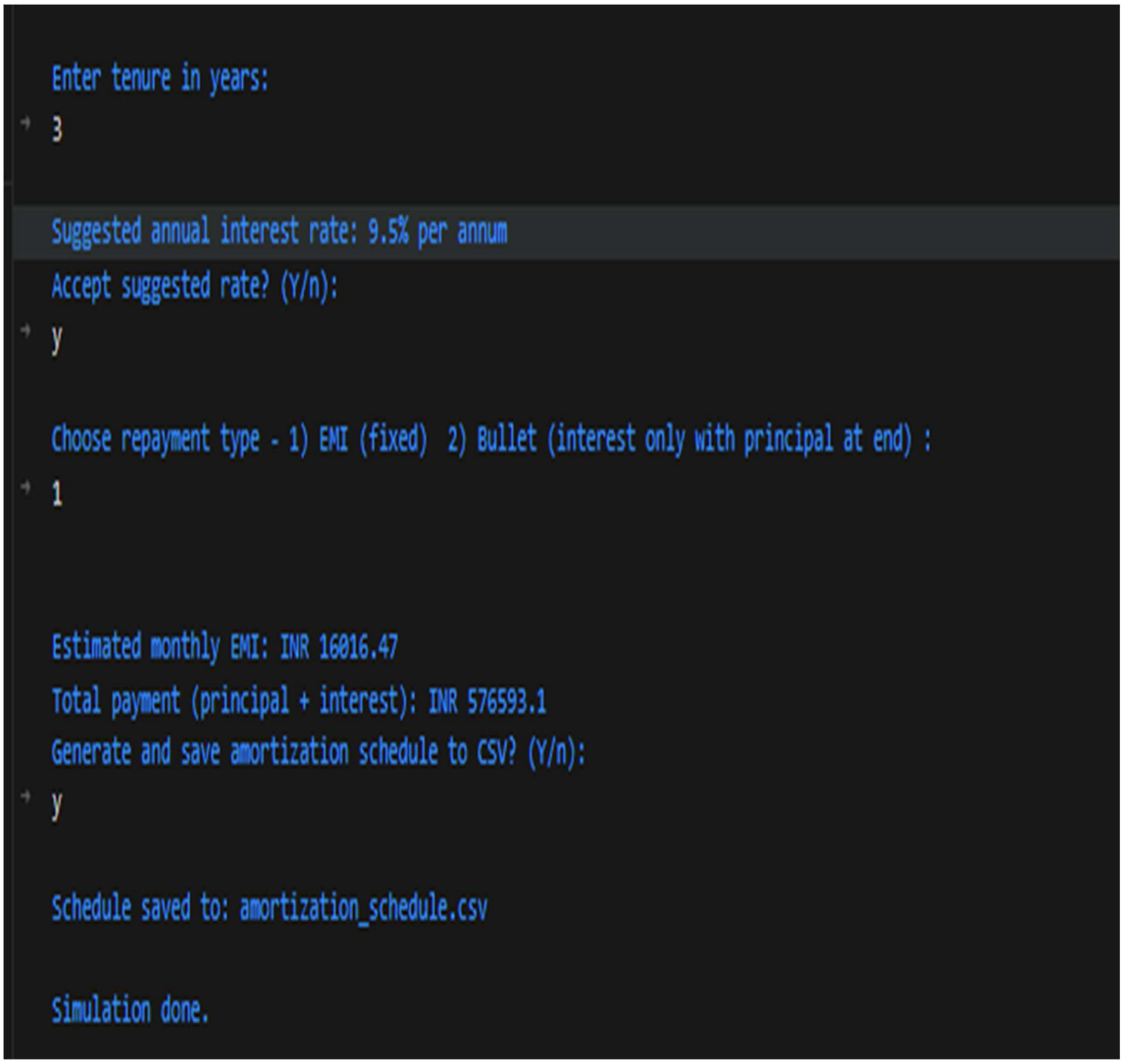
Akriti



Nitesh Kumar

Shreya

Akriti



Nitesh Kumar

Shreya

Akriti