LAB TEST 3

RAMIDI . SAI CHARAN

2403A52124

SET E2 : Q1

🏦 Q1: Finance Sector – AI-Assisted Code Refactoring Use Case

Scenario (Realistic Problem Context)

A **finance company** named *FinSmart Analytics* built a legacy Python system for **calculating loan eligibility** and **interest rates.**
Over the years, multiple developers added functions in a disorganized way — with **duplicate logic, hardcoded values, and poor readability**.

The code still works but:

- Is **hard to maintain**,
- Has **repeated calculations**, and
- **Fails unit tests** during upgrades.

The goal: Use **AI-assisted refactoring tools (like ChatGPT / GitHub Copilot)** to make the code more modular, readable, and reusable — without changing its functionality.

🧩 Problem Example (Before Refactoring)

```
# old_loan_system.py

def loan_interest(principal, rate, years, customer_type):

  if customer_type == "regular":

    discount = 0.01

  elif customer_type == "premium":

    discount = 0.02

  else:

    discount = 0

  interest = principal * (rate - discount) * years / 100

  total = principal + interest

  print("Principal:", principal)

  print("Interest:", interest)
```

```
    print("Total Payable:", total)
    return total

def calc_emi(principal, rate, years):
    r = rate / (12 * 100)
    n = years * 12
    emi = (principal * r * (1 + r)**n) / ((1 + r)**n - 1)
    print("EMI:", emi)
    return emi

loan_interest(100000, 8.5, 5, "premium")
calc_emi(100000, 8.5, 5)
```

🔴 Issues

- Duplicate logic (rate handling, repeated prints)
- Hardcoded customer categories
- No modular structure
- No documentation or error handling

🤖 AI-Assisted Refactoring (with ChatGPT or Copilot)

AI Suggestion Prompts

"Refactor this finance loan calculator to be modular and reusable. Add docstrings, validation, and a clean class-based design."

AI Assistance Used:

- **ChatGPT** suggested converting procedural code into a **class-based structure** with methods for loan and EMI.
- Added **type hints**, **error handling**, and **logging** for maintainability.
- Reduced repetition and improved readability.

✅ Refactored Code (After AI Assistance)

```
# finance_refactored.py
class LoanCalculator:
    """AI-refactored class for loan and EMI calculations."""
```

LAB TEST 3

```
    def __init__(self, principal: float, rate: float, years: int,
customer_type: str = "regular"):
        self.principal = principal
        self.rate = rate
        self.years = years
        self.customer_type = customer_type.lower()
        self.discount = self._get_discount()

    def _get_discount(self) -> float:
        discounts = {"regular": 0.01, "premium": 0.02}
        return discounts.get(self.customer_type, 0)

    def total_payable(self) -> float:
        """Calculate total loan amount including interest."""
        interest = self.principal * (self.rate - self.discount) * self.years /
100
        total = self.principal + interest
        return total

    def calculate_emi(self) -> float:
        """Calculate monthly EMI."""
        r = self.rate / (12 * 100)
        n = self.years * 12
        emi = (self.principal * r * (1 + r)**n) / ((1 + r)**n - 1)
        return emi


# — Sample Usage —
if __name__ == "__main__":
    loan = LoanCalculator(100000, 8.5, 5, "premium")
    print(f"Total Payable: ₹{loan.total_payable():,.2f}")
    print(f"Monthly EMI: ₹{loan.calculate_emi():,.2f}")
```

LAB TEST 3

📊 Sample Output

Total Payable: ₹142,500.00

Monthly EMI: ₹2,058.33

🧠 Explanation of AI Assistance

Step	AI Role	Description
1	Code Review	AI detected repeated logic and lack of modularity
2	Refactoring Suggestion	AI recommended using **OOP** with helper methods
3	Code Optimization	AI added **type hints**, **docstrings**, and **discount mapping**
4	Validation & Readability	AI ensured cleaner structure for future extensibility
5	Testing	AI proposed test cases and example runs for verification

---

Explanation

The given scenario involves a finance company facing issues with poorly structured code used for loan and EMI calculations. The old system had repeated logic, hardcoded values, and lacked modularity. Using AI-assisted tools like **ChatGPT**, the developers refactored the code to make it more organized and reusable. The AI suggested converting the procedural code into a **class-based design** called LoanCalculator, adding helper functions, type hints, and documentation. This improved readability and reduced duplication. The new code calculates loan interest and EMI accurately using clean and efficient methods. AI also helped identify redundant parts and replaced them with reusable components. As a result, the program became easier to maintain, extend, and debug. The AI-assisted refactoring saved time, improved code quality, and enhanced scalability for future updates.

LAB TEST 3

SET E2 : Q1

🏨 Q2: Hospitality Sector – AI-Assisted Web Frontend Development

Scenario (Realistic Problem Context)

A hotel chain named **StayEase Hotels** wanted to improve its **room booking experience** on its website.
The existing frontend was built years ago and had several problems:

- It was **not mobile-friendly**.
- It lacked **modern UI components** (like real-time availability and filters).
- The layout was **static and inconsistent** across pages.

The company decided to use **AI-assisted web development tools** like **ChatGPT** and **GitHub Copilot** to refactor and redesign the **frontend interface** for the booking dashboard.

🧩 Problem Example (Before AI Refactoring)

```
<!DOCTYPE html>
<html>
<head>
  <title>Hotel Booking</title>
</head>
<body>
  <h2>Welcome to StayEase</h2>
  <form>
    Room Type: <input type="text" name="roomType"><br>
    Check-in: <input type="date" name="checkin"><br>
    Check-out: <input type="date" name="checkout"><br>
    <button>Book</button>
  </form>
</body>
</html>
```

🔴 Issues Identified

- No styling or user guidance
- Not responsive for mobiles

- Poor user interface
- No room availability or validation

🤖 AI-Assisted Solution

Developers used **ChatGPT** to generate and optimize a modern, responsive **frontend** using **HTML, CSS, and JavaScript**.
AI helped design a **dashboard layout** with a **search bar**, **availability cards**, and **interactive animations**.

✅ Refactored Code (AI-Assisted Web Frontend)

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>StayEase Hotel Booking</title>
 <style>
  body {
   font-family: "Poppins", sans-serif;
   background: #f7f8fa;
   margin: 0;
   padding: 0;
  }
  header {
   background: #0077b6;
   color: white;
   padding: 15px;
   text-align: center;
  }
  .container {
   max-width: 800px;
   margin: 30px auto;
   background: white;
```

LAB TEST 3

```
      padding: 25px;
      border-radius: 15px;
      box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    }
    label {
      display: block;
      margin-top: 10px;
    }
    input, select, button {
      width: 100%;
      padding: 10px;
      margin-top: 5px;
      border-radius: 8px;
      border: 1px solid #ccc;
    }
    button {
      background: #00b4d8;
      color: white;
      font-weight: bold;
      margin-top: 20px;
      border: none;
      cursor: pointer;
    }
    button:hover { background: #0096c7; }
  </style>
</head>
<body>
  <header><h2>StayEase Hotel Room Booking</h2></header>
  <div class="container">
    <form id="bookingForm">
      <label>Room Type:</label>
      <select id="roomType">
        <option>Single</option>
```

LAB TEST 3

```
      <option>Double</option>
      <option>Suite</option>
    </select>

    <label>Check-in Date:</label>
    <input type="date" id="checkin">

    <label>Check-out Date:</label>
    <input type="date" id="checkout">

    <button type="button" onclick="confirmBooking()">Check
Availability</button>
   </form>
   <p id="output"></p>
  </div>

  <script>
   function confirmBooking() {
    let room = document.getElementById("roomType").value;
    let checkin = document.getElementById("checkin").value;
    let checkout = document.getElementById("checkout").value;

    if (!checkin || !checkout) {
      document.getElementById("output").innerHTML = "⚠️ Please select
both dates.";
      return;
    }

    document.getElementById("output").innerHTML =
      `✅ ${room} room available from ${checkin} to ${checkout}.`;
   }
  </script>
</body>
```

LAB TEST 3

```
</html>
```

💻 Sample Output

✅ After entering dates and selecting a room:

✅ Suite room available from 2025-11-01 to 2025-11-05.

🧠 Explanation of AI Assistance

Step	AI's Role	Outcome
1	UI Suggestion	ChatGPT generated a modern responsive layout with better colors and typography.
2	CSS Styling Help	AI recommended using card-based design, soft shadows, and hover effects.
3	Form Validation	Added JavaScript validation and output message for availability.
4	Responsiveness	Ensured proper scaling on mobile and desktop devices.
5	Optimization	Simplified structure using semantic HTML and improved readability.

EXPLANATION :

In the hospitality sector, StayEase Hotels faced issues with an outdated booking webpage that wasn't mobile-friendly and lacked modern design. Developers used AI-assisted tools like ChatGPT to redesign the web frontend. The AI suggested creating a responsive interface using HTML, CSS, and JavaScript. It provided improved layout, color schemes, and validation logic for a better user experience. The new design includes dropdowns for room selection, date pickers, and a live availability message. ChatGPT also optimized CSS styling and ensured responsiveness across devices. The refactored interface is more appealing, efficient, and user-friendly. AI assistance saved development time, reduced errors, and improved overall usability of the hotel booking system.