**Week 5: Thursday InClass Assignment**

Ashok Nath

Rina Katuwal

Prashant Ojha

Priyanka Gupta

Meena Ale Magar

Bikram Khatiwada

Kaushal Kishore Rai

Master Information Technology, Atlantis University

MIT 563: Programming and Applications Development

Professor: Alex Lima

Sep 26, 2024

**1 Customer Service module and integration with HR module –**

**A diagram of customer service

Description automatically generated**

*Fig: Customer Service module with HR Integration*

Codebase: Github Repository

**2 Document: Integration of Customer Service Module and Human Resources in**

**2.1 Overview**

This document describes the integration of the Human Resources (HR) module with the Customer Service (CS) module. The integrated system tracks employee management and specific customer service metrics, demonstrating how the two systems collaborate to provide a complete overview of an organization's HR and customer service operations. The program ensures that HR employees belonging to the customer service department can be tracked for their customer-facing performance metrics, while other employees are managed through general HR operations.

**2.2 System Architecture**

1. **HR Module**:
   * Handles employee details like ID, name, role, hours worked, overtime, unavailable hours, training hours, promotion status, and vacation status.
   * Provides functionality to log these details for all employees across the organization.
2. **Customer Service Module**:
   * Extends HR functionality specifically for customer service employees.
   * Tracks clients served, average service time, customer satisfaction, and the specific service area.
3. **Integration**:
   * The integration ensures that employees added to the HR module who belong to the customer service department are also tracked in the CS module.
   * HR manages employee attributes such as hours worked, promotions, etc., while CS tracks customer service performance metrics.

**2.3 Integration Details**

* The CustomerServiceManager class interacts with employees managed by the HRManager class.
* When an employee is hired or added to the HR system, they can be registered in the CustomerServiceManager if they belong to the customer service department.
* Variables like employee ID, name, and role are common between both modules. HR focuses on general employee management (hours worked, promotions), and the CS module focuses on client interaction metrics (clients served, service quality, etc.).

**3 Python Code Implementation for Customer Service:**

| # -\*- coding: utf-8 -\*- """CustomerService This module manages employees in the customer-service department and tracks customer service metrics such as service quality, number of clients served, average service time, and customer satisfaction. """  class CustomerServiceEmployee:  def \_\_init\_\_(self, emp\_id, name, role):  self.emp\_id = emp\_id  self.name = name  self.role = role  self.clients\_served = 0  self.total\_service\_time = 0  self.customer\_satisfaction = []  self.service\_area = ""   def serve\_client(self, service\_time, satisfaction\_level):  self.clients\_served += 1  self.total\_service\_time += service\_time  self.customer\_satisfaction.append(satisfaction\_level)    def set\_service\_area(self, service\_area):  self.service\_area = service\_area    def calculate\_average\_service\_time(self):  return self.total\_service\_time / self.clients\_served if self.clients\_served > 0 else 0    def calculate\_average\_satisfaction(self):  return sum(self.customer\_satisfaction) / len(self.customer\_satisfaction) if self.customer\_satisfaction else 0    def display\_customer\_service\_info(self):  print(f"Customer Service Employee ID: {self.emp\_id}")  print(f"Name: {self.name}")  print(f"Role: {self.role}")  print(f"Clients Served: {self.clients\_served}")  print(f"Average Service Time: {self.calculate\_average\_service\_time()} minutes")  print(f"Average Customer Satisfaction Level: {self.calculate\_average\_satisfaction()}")  print(f"Service Area: {self.service\_area}")  print("\n")   class CustomerServiceManager:  def \_\_init\_\_(self):  self.cs\_employees = {}    def add\_cs\_employee(self, emp\_id, name, role):  self.cs\_employees[emp\_id] = CustomerServiceEmployee(emp\_id, name, role)    def log\_service(self, emp\_id, service\_time, satisfaction\_level):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].serve\_client(service\_time, satisfaction\_level)    def set\_service\_area(self, emp\_id, service\_area):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].set\_service\_area(service\_area)    def display\_cs\_employee\_info(self, emp\_id):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].display\_customer\_service\_info()   # Example usage if \_\_name\_\_ == "\_\_main\_\_":  customer\_service\_manager = CustomerServiceManager()    # Add customer service employees  customer\_service\_manager.add\_cs\_employee(1, "Chris Brown", "Customer Service Rep")  customer\_service\_manager.add\_cs\_employee(2, "Emma Wilson", "Customer Support Specialist")  customer\_service\_manager.add\_cs\_employee(3, "Derek Robinson", "Customer Support Specialist")  customer\_service\_manager.add\_cs\_employee(4, "Rachel Raynolds", "Customer Support Specialist")    # Log service information  customer\_service\_manager.log\_service(1, 15, 4.5) # 15 minutes call with 4.5 satisfaction  customer\_service\_manager.log\_service(1, 10, 4.0) # 10 minutes call with 4.0 satisfaction  customer\_service\_manager.log\_service(2, 20, 5.0) # 20 minutes call with 5.0 satisfaction  customer\_service\_manager.log\_service(3, 25, 4.8) # 25 minutes call with 4.8 satisfaction  customer\_service\_manager.log\_service(4, 30, 4.7) # 30 minutes call with 4.7 satisfaction  customer\_service\_manager.log\_service(4, 18, 4.3) # 18 minutes call with 4.3 satisfaction    # Set service area  customer\_service\_manager.set\_service\_area(1, "Product Support")  customer\_service\_manager.set\_service\_area(2, "Technical Assistance")  customer\_service\_manager.set\_service\_area(3, "Technical Support")  customer\_service\_manager.set\_service\_area(4, "Customer Relations")   # Display customer service info for employees  customer\_service\_manager.display\_cs\_employee\_info(1)  customer\_service\_manager.display\_cs\_employee\_info(2)  customer\_service\_manager.display\_cs\_employee\_info(3)  customer\_service\_manager.display\_cs\_employee\_info(4) |
| --- |

**Output:**

Customer Service Employee ID: 1

Name: Chris Brown

Role: Customer Service Rep

Clients Served: 2

Average Service Time: 12.5 minutes

Average Customer Satisfaction Level: 4.25

Service Area: Product Support

Customer Service Employee ID: 2

Name: Emma Wilson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 20.0 minutes

Average Customer Satisfaction Level: 5.0

Service Area: Technical Assistance

Customer Service Employee ID: 3

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Name: Derek Robinson

Role: Customer Support Specialist

Name: Derek Robinson

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Customer Service Employee ID: 4

Name: Rachel Raynolds

Role: Customer Support Specialist

Clients Served: 2

Average Service Time: 24.0 minutes

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Customer Service Employee ID: 4

Name: Rachel Raynolds

Role: Customer Support Specialist

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Customer Service Employee ID: 4

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Customer Service Employee ID: 4

Name: Rachel Raynolds

Role: Customer Support Specialist

Clients Served: 2

Average Service Time: 24.0 minutes

Average Customer Satisfaction Level: 4.5

Service Area: Customer Relations

**4 Employee Information from HR:**

| class Employee:  def \_\_init\_\_(self, emp\_id, name, role):  self.emp\_id = emp\_id  self.name = name  self.role = role  self.hours\_worked = 0  self.overtime\_hours = 0  self.unavailable\_hours = 0  self.training\_hours = 0  self.promoted = False  self.on\_vacation = False   def work(self, hours):  self.hours\_worked += hours   def log\_overtime(self, hours):  self.overtime\_hours += hours   def mark\_unavailable(self, hours):  self.unavailable\_hours += hours   def log\_training(self, hours):  self.training\_hours += hours   def promote(self):  self.promoted = True   def start\_vacation(self):  self.on\_vacation = True   def end\_vacation(self):  self.on\_vacation = False  class HRManager:  def \_\_init\_\_(self):  self.employees = {}  self.to\_be\_hired = []   def add\_employee(self, emp\_id, name, role):  self.employees[emp\_id] = Employee(emp\_id, name, role)   def remove\_employee(self, emp\_id):  if emp\_id in self.employees:  del self.employees[emp\_id]   def hire\_employee(self, name, role):  self.to\_be\_hired.append((name, role))   def start\_vacation(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].start\_vacation()   def end\_vacation(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].end\_vacation()   def log\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].work(hours)   def log\_overtime(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].log\_overtime(hours)   def log\_unavailable\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].mark\_unavailable(hours)   def log\_training\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].log\_training(hours)   def promote\_employee(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].promote()   def display\_employee\_info(self, emp\_id):  if emp\_id in self.employees:  emp = self.employees[emp\_id]  print(f"Employee ID: {emp.emp\_id}")  print(f"Name: {emp.name}")  print(f"Role: {emp.role}")  print(f"Hours Worked: {emp.hours\_worked}")  print(f"Overtime Hours: {emp.overtime\_hours}")  print(f"Unavailable Hours: {emp.unavailable\_hours}")  print(f"Training Hours: {emp.training\_hours}")  print(f"On Vacation: {'Yes' if emp.on\_vacation else 'No'}")  print(f"Promoted: {'Yes' if emp.promoted else 'No'}")  print("\n")   def display\_to\_be\_hired(self):  print("Employees to be Hired:")  for name, role in self.to\_be\_hired:  print(f"Name: {name}, Role: {role}")  # Example usage if \_\_name\_\_ == "\_\_main\_\_":  hr\_manager = HRManager()   # Add some employees  hr\_manager.add\_employee(1, "John Doe", "Developer")  hr\_manager.add\_employee(2, "Jane Smith", "Designer")   # Log hours worked  hr\_manager.log\_hours(1, 40)  hr\_manager.log\_overtime(1, 5)  hr\_manager.log\_training\_hours(2, 8)   # Start and end vacation  hr\_manager.start\_vacation(2)  hr\_manager.end\_vacation(2)   # Promote an employee  hr\_manager.promote\_employee(1)   # Display employee info  hr\_manager.display\_employee\_info(1)  hr\_manager.display\_employee\_info(2)   # Hire new employees  hr\_manager.hire\_employee("Alice Johnson", "Product Manager")  hr\_manager.hire\_employee("Bob Lee", "QA Engineer")  hr\_manager.display\_to\_be\_hired() |
| --- |

**Output:**

Employee ID: 1

Name: John Doe

Role: Developer

Hours Worked: 40

Overtime Hours: 5

Unavailable Hours: 0

Training Hours: 0

On Vacation: No

Promoted: Yes

Employee ID: 2

Name: Jane Smith

Role: Designer

Hours Worked: 0

Overtime Hours: 0

Unavailable Hours: 0

Training Hours: 8

On Vacation: No

Promoted: No

Employees to be Hired:

Name: Alice Johnson, Role: Product Manager

Name: Bob Lee, Role: QA Engineer

**5 Integrated Program for Customer Service, HR and Inventory:**

| # -\*- coding: utf-8 -\*- """CustomerService and Inventory Integration"""  # Existing Classes (Employee, HRManager, CustomerServiceEmployee, CustomerServiceManager)  class Employee:  def \_\_init\_\_(self, emp\_id, name, role):  self.emp\_id = emp\_id  self.name = name  self.role = role  self.hours\_worked = 10  self.overtime\_hours = 20  self.unavailable\_hours = 30  self.training\_hours = 40  self.promoted = False  self.on\_vacation = False   def work(self, hours):  self.hours\_worked += hours   def log\_overtime(self, hours):  self.overtime\_hours += hours   def mark\_unavailable(self, hours):  self.unavailable\_hours += hours   def log\_training(self, hours):  self.training\_hours += hours   def promote(self):  self.promoted = True   def start\_vacation(self):  self.on\_vacation = True   def end\_vacation(self):  self.on\_vacation = False  class HRManager:  def \_\_init\_\_(self):  self.employees = {}  self.to\_be\_hired = []   def add\_employee(self, emp\_id, name, role):  self.employees[emp\_id] = Employee(emp\_id, name, role)   def remove\_employee(self, emp\_id):  if emp\_id in self.employees:  del self.employees[emp\_id]   def hire\_employee(self, name, role):  self.to\_be\_hired.append((name, role))   def start\_vacation(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].start\_vacation()   def end\_vacation(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].end\_vacation()   def log\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].work(hours)   def log\_overtime(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].log\_overtime(hours)   def log\_unavailable\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].mark\_unavailable(hours)   def log\_training\_hours(self, emp\_id, hours):  if emp\_id in self.employees:  self.employees[emp\_id].log\_training(hours)   def promote\_employee(self, emp\_id):  if emp\_id in self.employees:  self.employees[emp\_id].promote()    def display\_employee\_info(self, emp\_id):  if emp\_id in self.employees:  emp = self.employees[emp\_id]  if isinstance(emp, CustomerServiceEmployee):  emp.display\_customer\_service\_info()  else:   print(f"Employee ID: {emp.emp\_id}")  print(f"Name: {emp.name}")  print(f"Role: {emp.role}")  print(f"Hours Worked: {emp.hours\_worked}")  print(f"Overtime Hours: {emp.overtime\_hours}")  print(f"Unavailable Hours: {emp.unavailable\_hours}")  print(f"Training Hours: {emp.training\_hours}")  print(f"On Vacation: {'Yes' if emp.on\_vacation else 'No'}")  print(f"Promoted: {'Yes' if emp.promoted else 'No'}")  print("\n")   def display\_to\_be\_hired(self):  print("Employees to be Hired:")  for name, role in self.to\_be\_hired:  print(f"Name: {name}, Role: {role}")  class CustomerServiceEmployee:  def \_\_init\_\_(self, emp\_id, name, role):  self.emp\_id = emp\_id  self.name = name  self.role = role  self.clients\_served = 0  self.total\_service\_time = 0  self.customer\_satisfaction = []  self.service\_area = ""   def serve\_client(self, service\_time, satisfaction\_level):  self.clients\_served += 1  self.total\_service\_time += service\_time  self.customer\_satisfaction.append(satisfaction\_level)   def set\_service\_area(self, service\_area):  self.service\_area = service\_area   def calculate\_average\_service\_time(self):  return self.total\_service\_time / self.clients\_served if self.clients\_served > 0 else 0   def calculate\_average\_satisfaction(self):  return sum(self.customer\_satisfaction) / len(self.customer\_satisfaction) if self.customer\_satisfaction else 0   def display\_customer\_service\_info(self):  print("\n")   print(f"Customer Service Employee ID: {self.emp\_id}")  print(f"Name: {self.name}")  print(f"Role: {self.role}")  print(f"Clients Served: {self.clients\_served}")  print(f"Average Service Time: {self.calculate\_average\_service\_time()} minutes")  print(f"Average Customer Satisfaction Level: {self.calculate\_average\_satisfaction()}")  print(f"Service Area: {self.service\_area}")  class CustomerServiceManager:  def \_\_init\_\_(self):  self.cs\_employees = {}   def add\_cs\_employee(self, emp\_id, name, role):  self.cs\_employees[emp\_id] = CustomerServiceEmployee(emp\_id, name, role)   def log\_service(self, emp\_id, service\_time, satisfaction\_level):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].serve\_client(service\_time, satisfaction\_level)   def set\_service\_area(self, emp\_id, service\_area):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].set\_service\_area(service\_area)   def display\_cs\_employee\_info(self, emp\_id):  if emp\_id in self.cs\_employees:  self.cs\_employees[emp\_id].display\_customer\_service\_info()   # Inventory Management System class Inventory:  def \_\_init\_\_(self):  self.products = {}   def add\_product(self, product\_name, quantity, price\_per\_unit):  if product\_name in self.products:  print(f"{product\_name} already exists. Updating quantity.")  self.products[product\_name]['quantity'] += quantity  else:  self.products[product\_name] = {'quantity': quantity, 'price\_per\_unit': price\_per\_unit}   def remove\_product(self, product\_name):  if product\_name in self.products:  del self.products[product\_name]  else:  print(f"Product {product\_name} not found.")   def mark\_damaged(self, product\_name, quantity):  if product\_name in self.products and self.products[product\_name]['quantity'] >= quantity:  self.products[product\_name]['quantity'] -= quantity  print(f"{quantity} units of {product\_name} marked as damaged.")  else:  print(f"Insufficient quantity of {product\_name}.")   def display\_inventory(self):  print("Inventory List:")  for product, details in self.products.items():  print(f"Product: {product}, Quantity: {details['quantity']}, Price: {details['price\_per\_unit']}")  # Integrating HR, Customer Service, and Inventory Management class CompanyManager:  def \_\_init\_\_(self):  self.hr\_manager = HRManager()  self.customer\_service\_manager = CustomerServiceManager()  self.inventory = Inventory()   def display\_company\_info(self):  print("HR Employees:")  for emp\_id in self.hr\_manager.employees:  self.hr\_manager.display\_employee\_info(emp\_id)    print("\nCustomer Service Employees:")  for emp\_id in self.customer\_service\_manager.cs\_employees:  self.customer\_service\_manager.display\_cs\_employee\_info(emp\_id)    print("\nInventory:")  self.inventory.display\_inventory()  # Example of usage if \_\_name\_\_ == "\_\_main\_\_":  company = CompanyManager()   # HR and Customer Service operations  company.hr\_manager.add\_employee(1, "John Doe", "Developer")  company.hr\_manager.add\_employee(2, "Jane Smith", "Designer")  company.customer\_service\_manager.add\_cs\_employee(1, "Chris Brown", "Customer Service Rep")   # Inventory operations  company.inventory.add\_product("Laptop", 50, 1200)  company.inventory.add\_product("Headset", 100, 50)  company.inventory.mark\_damaged("Laptop", 5)   # Display the entire company info  company.display\_company\_info() |
| --- |

**Output:**

Employee Information

Employee ID: 1

Name: John Doe

Role: Developer

Hours Worked: 50

Overtime Hours: 25

Unavailable Hours: 30

Training Hours: 40

On Vacation: No

Promoted: No

Employee ID: 2

Name: Jane Smith

Role: Designer

Hours Worked: 10

Overtime Hours: 20

Unavailable Hours: 30

Training Hours: 48

On Vacation: No

Promoted: No

Employees to be Hired:

Name: Alice Johnson, Role: Product Manager

Name: Bob Lee, Role: QA Engineer

Customer Service Employee ID: 1

Name: Chris Brown

Role: Customer Service Rep

Clients Served: 2

Average Service Time: 12.5 minutes

Average Customer Satisfaction Level: 4.25

Service Area: Product Support

Customer Service Employee ID: 2

Name: Emma Wilson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 20.0 minutes

Average Customer Satisfaction Level: 5.0

Service Area: Technical Assistance

Customer Service Employee ID: 3

Name: Derek Robinson

Role: Customer Support Specialist

Clients Served: 1

Average Service Time: 25.0 minutes

Average Customer Satisfaction Level: 4.8

Service Area: Technical Support

Customer Service Employee ID: 4

Name: Rachel Raynolds

Role: Customer Support Specialist

Clients Served: 2

Average Service Time: 24.0 minutes

Average Customer Satisfaction Level: 4.5

Service Area: Customer Relations

5 units of Laptop marked as damaged.

Inventory List:

Product: Laptop, Quantity: 45, Price: 1200

Product: Headset, Quantity: 100, Price: 50

**6 Conclusion:**

This integrated program provides a complete solution to manage general HR, customer service and inventory metrics in one system, producing concrete results about employee performance and customer service quality. This allows businesses to maintain efficient employee tracking and customer service optimization.