**FINAL PROJECT STEP 2**

Rajeswari Tadepalli

Department of Computer Science, Monroe College, King Graduate School

CS625: Object-Oriented Software

Prof. Ronald Adomako

November 18, 2023

Final Project - CS625 – Step 2

The Project Presentation is on July 27th (Week 14) at New Rochelle Campus. Use PowerPoint Slides for the presentation because we are going to have some students online,

The Final project report includes the Python code is due on July 27th (Week 14).

Hands-on Project

Note: There are no requirement for the number of pages or references for the project:

The final project presents an opportunity for students to solve a practical problem using python programming language.

Final Project Scoring Rubric - To do List. Total points 200

* Requirement Analysis Phase 50/50
* Design Phase 50/50
* Implementation and Testing Phase 50/50
* Project Presentation (in class) 50/50

Design Phase (Due on July 20th) – Step 2

* 1. The project must at least 4 or more classes
  2. We will use the following Object-oriented programming and data structure techniques.
     1. Classes
     2. Inheritance
     3. Polymorphism
     4. Data structure technique (use at least one data structure method to organize your data, example, stack, queue, tree, graph, etc…).
     5. Include the class diagram, activity diagram and use case

diagram.

Code:

from datetime import datetime

# Class 1: Employee

class Employee:

def \_\_init\_\_(self, emp\_id, emp\_name):

self.emp\_id = emp\_id

self.emp\_name = emp\_name

self.attendance\_log = [] # Data structure for storing attendance records

def mark\_attendance(self, date, status, time\_in=None, time\_out=None):

self.attendance\_log.append({"date": date, "status": status, "time\_in": time\_in, "time\_out": time\_out})

# Class 2: AttendanceTracker

class AttendanceTracker:

def \_\_init\_\_(self):

self.employees = [] # List to store instances of Employee class

def add\_employee(self, employee):

self.employees.append(employee)

def track\_attendance(self, emp\_id, date, status, time\_in=None, time\_out=None):

for employee in self.employees:

if employee.emp\_id == emp\_id:

employee.mark\_attendance(date, status, time\_in, time\_out)

return f"Attendance tracked for {employee.emp\_name} on {date}."

return f"Employee with ID {emp\_id} not found."

# Class 3: AttendanceAnalyzer

class AttendanceAnalyzer:

def generate\_report(self, employee):

# Analyze attendance data and generate a detailed report

print(f"\nAttendance Report for Employee {employee.emp\_name} (ID: {employee.emp\_id}):")

for entry in employee.attendance\_log:

print(f"Date: {entry['date']}, Status: {entry['status']}, Time In: {entry['time\_in']}, Time Out: {entry['time\_out']}")

def analyze(self, employees):

for employee in employees:

self.generate\_report(employee)

# Class 4: AttendanceSystem

class AttendanceSystem:

def \_\_init\_\_(self):

self.attendance\_tracker = AttendanceTracker()

self.attendance\_analyzer = AttendanceAnalyzer()

def run(self):

# Example usage of the system

employee1 = Employee(emp\_id=1, emp\_name="Rajeswari")

employee2 = Employee(emp\_id=2, emp\_name="Anushka")

self.attendance\_tracker.add\_employee(employee1)

self.attendance\_tracker.add\_employee(employee2)

self.attendance\_tracker.track\_attendance(emp\_id=1,date="2023-11-01", status="Present",time\_in="10:30",time\_out="4:00")

self.attendance\_tracker.track\_attendance(emp\_id=2,date="2023-11-02", status="Absent")

employees = self.attendance\_tracker.employees

self.attendance\_analyzer.analyze(employees)

attendance\_system = AttendanceSystem()

attendance\_system.run()

Class Diagram:

A diagram of a company

Description automatically generated

Use Case Diagram:

A white paper with black text

Description automatically generated

Activity Diagram:

A flowchart of a program

Description automatically generated