**FINAL LAB**

******

**SUBJECT**

**SOFTWARE DESIGN AND ARCHITECTURE**

**TEACHER**

**SIR MUKHAIR ZAMIN**

**SUBMITTED BY**

**NIMRA JADOON**

**FA22-BSE-011**

**DEADLINE**

**01. 06. 2025**

**Department of Software Engineering**

**COMSATS University Islamabad**

**Abbottabad campus**

The architecture shown in the diagram follows a combination of the following architectural styles:

1. **Layered Architecture**:
   * The system is divided into logical layers: User Interface, Scheduling Module, Monitoring System, Communication Network, and Control System.
   * Each layer has a distinct responsibility, promoting separation of concerns.
2. **Microservices Architecture**:
   * Each module (e.g., Scheduling, Monitoring, Communication, Control System) can be implemented as independent, modular services that communicate with each other.
3. **Client-Server Architecture**:
   * The User Interface (Individual User and Group User) interacts with the backend systems, such as the Scheduling Module and Monitoring System, in a client-server manner.
4. **Distributed Architecture**:
   * The Communication Network and Control System imply a distributed architecture where different systems (e.g., vehicles, central system) work together in real time.

But I preferred layered architecture and coding is given as:

**Coding:**

**1. Domain Layer (Represents the core entities)**

package domain;

import java.util.ArrayList;

import java.util.List;

public class Individual {

private String name;

private int age;

private String role;

private List<String> tasks;

private double efficiency;

public Individual(String name, int age, String role, double efficiency) {

this.name = name;

this.age = age;

this.role = role;

this.tasks = new ArrayList<>();

this.efficiency = efficiency;

}

public void performTask(String task) {

System.out.println(name + " is performing task: " + task);

tasks.add(task);

}

public void reportProgress() {

System.out.println(name + " has completed " + tasks.size() + " tasks.");

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

package domain;

public class Task {

private String description;

private String assignedTo;

private boolean isCompleted;

// Constructor

public Task(String description, String assignedTo) {

this.description = description;

this.assignedTo = assignedTo;

this.isCompleted = false; // Default value for a new task

}

// Mark task as completed

public void markAsCompleted() {

this.isCompleted = true;

System.out.println("Task: " + description + " has been marked as completed.");

}

// Getters and Setters

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getAssignedTo() {

return assignedTo;

}

public void setAssignedTo(String assignedTo) {

this.assignedTo = assignedTo;

}

public boolean isCompleted() {

return isCompleted;

}

public void setCompleted(boolean isCompleted) {

this.isCompleted = isCompleted;

}

}

package domain;

import java.util.ArrayList;

import java.util.List;

public class Plant {

private String name;

private String location;

private List<Individual> workers;

// Constructor

public Plant(String name, String location) {

this.name = name;

this.location = location;

this.workers = new ArrayList<>(); // Initialize the workers list

}

// Add worker to the plant

public void addWorker(Individual individual) {

workers.add(individual);

System.out.println("Worker " + individual.getName() + " added to the plant.");

}

// Remove worker from the plant

public void removeWorker(Individual individual) {

workers.remove(individual);

System.out.println("Worker " + individual.getName() + " removed from the plant.");

}

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public List<Individual> getWorkers() {

return workers;

}

}

**2. Service Layer (Business logic)**

package service;

import domain.Individual;

import domain.Plant;

import domain.Task;

public class PlantService {

private Plant plant;

public PlantService(Plant plant) {

this.plant = plant;

}

public void assignTaskToWorker(Task task, Individual worker) {

if (plant.getWorkers().contains(worker)) {

worker.performTask(task.getDescription());

task.markAsCompleted();

} else {

System.out.println("Worker " + worker.getName() + " is not part of the plant.");

}

}

public void reportPlantProgress() {

System.out.println("Plant: " + plant.getName() + " Progress Report");

for (Individual worker : plant.getWorkers()) {

worker.reportProgress();

}

}

}

**3. Data Access Layer (Handles persistence, here simulated)**

package data;

import domain.Plant;

import java.util.HashMap;

import java.util.Map;

public class PlantRepository {

private Map<String, Plant> plants = new HashMap<>();

public void savePlant(Plant plant) {

plants.put(plant.getName(), plant);

System.out.println("Plant " + plant.getName() + " saved.");

}

public Plant findPlantByName(String name) {

return plants.get(name);

}

}

**4. Main Class (To test the functionality)**

package main;

import domain.Individual;

import domain.Plant;

import domain.Task;

import service.PlantService;

import data.PlantRepository;

public class Main {

public static void main(String[] args) {

// Create Domain Objects

Individual worker1 = new Individual("Alice", 30, "Technician", 0.85);

Individual worker2 = new Individual("Bob", 35, "Supervisor", 0.9);

Plant plant = new Plant("Main Plant", "New York"); // Properly initialize the Plant

// Add workers to the plant

plant.addWorker(worker1);

plant.addWorker(worker2);

// Create Tasks

Task task1 = new Task("Fix Machine A", "Alice");

Task task2 = new Task("Inspect Machine B", "Bob");

// Service Layer

PlantService plantService = new PlantService(plant);

plantService.assignTaskToWorker(task1, worker1);

plantService.assignTaskToWorker(task2, worker2);

// Report Progress

plantService.reportPlantProgress();

// Data Access Layer

PlantRepository repository = new PlantRepository();

repository.savePlant(plant);

}

}

**Output:**

