class Employee:

def \_\_init\_\_(self, employee\_id, name):

self.employee\_id = employee\_id

self.name = name

self.assigned\_tasks = []

self.permissions = {}

def complete\_task(self, task\_id):

task = self.get\_task\_by\_id(task\_id)

if task and self.permissions.get(task\_id, False):

task.complete()

else:

raise PermissionError("No access to this task")

def request\_help(self, query):

return {

'employee\_id': self.employee\_id,

'query': query

}

def view\_progress(self):

progress = {task.task\_id: task.get\_progress() for task in self.assigned\_tasks}

return progress

def get\_task\_by\_id(self, task\_id):

for task in self.assigned\_tasks:

if task.task\_id == task\_id:

return task

return None

class Task:

def \_\_init\_\_(self, task\_id, name, deadline):

self.task\_id = task\_id

self.name = name

self.deadline = deadline

self.completed = False

self.resources = []

def complete(self):

self.completed = True

def get\_progress(self):

return 'Completed' if self.completed else 'In Progress'

def add\_resource(self, resource):

self.resources.append(resource)

class TaskAndResourceManager:

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

self.employees = []

self.tasks = []

self.priority\_set = []

self.remaining\_tasks = []

self.resource\_set = []

def assign\_task(self, employee, task):

employee.assigned\_tasks.append(task)

employee.permissions[task.task\_id] = True

self.remaining\_tasks.append(task)

def track\_progress(self):

progress = {task.task\_id: task.get\_progress() for task in self.tasks}

return progress

def reassign\_task(self, task, new\_employee):

self.remove\_task\_from\_employee(task)

self.assign\_task(new\_employee, task)

def handle\_query(self, query):

# Forward the query to SubManager or Manager based on logic

pass

def allocate\_resources(self, task, resources):

task.add\_resource(resources)

self.resource\_set.append(resources)

def manage\_priority\_set(self):

# Prioritize tasks to minimize deadline misses

pass

def update\_task\_status(self, task, status):

task.completed = status

def remove\_task\_from\_employee(self, task):

for employee in self.employees:

if task in employee.assigned\_tasks:

employee.assigned\_tasks.remove(task)

del employee.permissions[task.task\_id]

class SubManager:

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

self.tasks = []

def assign\_task(self, task, task\_manager):

task\_manager.tasks.append(task)

def supervise\_task(self, task):

return task.get\_progress()

def forward\_query(self, query):

# Forward query to Manager if necessary

pass

def update\_task\_status(self, task, status):

task.completed = status

class Manager:

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

self.sub\_managers = []

self.tasks = []

def delegate\_task(self, task, sub\_manager):

sub\_manager.assign\_task(task)

def supervise\_task(self, task):

return task.get\_progress()

def forward\_query(self, query):

# Forward query to Admin if necessary

pass

def update\_task\_status(self, task, status):

task.completed = status

class Admin:

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

self.managers = []

def grant\_permissions(self, user, permissions):

user.permissions.update(permissions)

def update\_task\_status(self, task, status):

task.completed = status

def handle\_query(self, query):

# Handle query or forward it to appropriate manager

pass