import datetime

print ("Welcome to Stakeholder Communication Platform!")

print ("-------------------------------------------------------------------------")

class User:

def \_\_init\_\_(self, user\_id, username, password, email, role):

self.user\_id = user\_id

self. username = username

self.password = password

self.email = email

self.role = role

def change\_password(self, new\_password):

self.password = new\_password

def \_\_repr\_\_(self):

return f"User(ID: {self.user\_id}, Username: {self.username}, Email: {self.email}, Role: {self.role})"

class UserManager:

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

self.users = {}

def create\_user(self, user\_id, username, password, email, role):

if user\_id in self.users:

print("User ID already exists!")

else:

self.users[user\_id] = User(user\_id, username, password, email, role)

print(f"User {username} created successfully!")

def read\_user(self, user\_id):

user = self.users.get(user\_id)

if user:

print(user)

else:

print("User not found!")

def update\_user(self, user\_id, username=None, password=None, email=None, role=None):

user = self.users.get(user\_id)

if user:

if username:

user.username = username

if password:

user.change\_password(password)

if email:

user.email = email

if role:

user.role = role

print(f"User {user\_id} updated successfully!")

else:

print("User not found!")

def delete\_user(self, user\_id):

if user\_id in self.users:

del self.users[user\_id]

print(f"User {user\_id} deleted successfully!")

else:

print("User not found!")

class CommunicationLog:

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

self.logs = {}

def create\_log(self, log\_id, stakeholder\_id, communication\_type, date):

if log\_id in self.logs:

print("Log ID already exists!")

else:

self.logs[log\_id] = {

'stakeholder\_id': stakeholder\_id,

'communication\_type': communication\_type,

'date': date,

}

print(f"Communication log {log\_id} created successfully!")

def read\_log(self, log\_id):

log = self.logs.get(log\_id)

if log:

print(f"Log: Log ID: {log\_id}, Communication Type: {log['communication\_type']}, Date: {log['date']}")

else:

print("Log not found!")

def update\_log(self, log\_id, communication\_type=None, date=None):

log = self.logs.get(log\_id)

if log:

if communication\_type:

log['communication\_type'] = communication\_type

if date:

log['date'] = date

print(f"Log {log\_id} updated successfully!")

else:

print("Log not found!")

def delete\_log(self, log\_id):

if log\_id in self.logs:

del self.logs[log\_id]

print(f"Log {log\_id} deleted successfully!")

else:

print("Log not found!")

class Stakeholder:

def \_\_init\_\_(self, stakeholder\_id, name, email):

self.stakeholder\_id = stakeholder\_id

self.name = name

self.email = email

self.feedback = []

class Feedback:

def \_\_init\_\_(self, feedback\_id, stakeholder\_id, date, feedback\_message):

self.feedback\_id = feedback\_id

self.stakeholder\_id = stakeholder\_id

self.date = date

self.feedback\_message = feedback\_message

class Task:

def \_\_init\_\_(self, name, schedule, status, purpose):

self.name = name

self.schedule = schedule

self.status = status

self.purpose = purpose

def \_\_str\_\_(self):

return f"Task: {self.name}\nSchedule: {self.schedule}\nStatus: {self.status}\nPurpose: {self.purpose}"

class ScheduleManager:

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

self.tasks = []

def add\_task(self):

name = input("Enter task name: ")

schedule = input("Enter task schedule (YYYY-MM-DD): ")

status = input("Enter task status: ")

purpose = input("Enter task purpose: ")

new\_task = Task(name, schedule, status, purpose)

self.tasks.append(new\_task)

print("Task added successfully!\n")

def view\_tasks(self):

if not self.tasks:

print("No tasks available.\n")

return

for i, task in enumerate(self.tasks):

print(f"Task {i + 1}:\n{task}\n")

def update\_task(self):

self.view\_tasks()

if not self.tasks:

return

try:

task\_index = int(input("Enter the task number to update: ")) - 1

if 0 <= task\_index < len(self.tasks):

schedule = input("Enter new schedule (leave blank to keep current): ")

status = input("Enter new status (leave blank to keep current): ")

purpose = input("Enter new purpose (leave blank to keep current): ")

if schedule:

self.tasks[task\_index].schedule = schedule

if status:

self.tasks[task\_index].status = status

if purpose:

self.tasks[task\_index].purpose = purpose

print("Task updated successfully!\n")

else:

print("Invalid task number.\n")

except ValueError:

print("Please enter a valid number.\n")

class StakeholderCommunicationPlatform:

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

self.logs = CommunicationLog()

self.feedback\_list = []

self.stakeholders = {}

def add\_feedback(self, stakeholder\_id, feedback\_message):

feedback\_id = len(self.feedback\_list) + 1

date = datetime.date.today().isoformat()

feedback = Feedback(feedback\_id, stakeholder\_id, date, feedback\_message)

self.feedback\_list.append(feedback)

print("Feedback added successfully!")

def read\_feedback(self, stakeholder\_id):

for feedback in self.feedback\_list:

if feedback.stakeholder\_id == stakeholder\_id:

print(f"Feedback ID: {feedback.feedback\_id}, Date: {feedback.date}, Message: {feedback.feedback\_message}")

print("End of feedback.")

def analyze\_communication\_effectiveness(self, stakeholder\_id):

total\_logs = 0

communication\_types = {}

for log in self.logs.logs.values():

if log['stakeholder\_id'] == stakeholder\_id:

total\_logs += 1

communication\_type = log['communication\_type']

communication\_types[communication\_type] = communication\_types.get(communication\_type, 0) + 1

if total\_logs == 0:

print(f"No communication logs found for Stakeholder ID: {stakeholder\_id}.")

return

print(f"Communication Analysis for Stakeholder ID: {stakeholder\_id}")

print(f"Total Communications: {total\_logs}")

for comm\_type, count in communication\_types.items():

print(f"{comm\_type.capitalize()}: {count} ({(count / total\_logs) \* 100:.2f}%)")

def main\_menu():

platform = StakeholderCommunicationPlatform()

user\_manager = UserManager()

schedule\_manager = ScheduleManager()

while True:

print("\nWelcome To Stakeholder Communication Platform!")

print("1. Stakeholder Details")

print("2. Communication Log")

print("3. Feedback")

print("4. Schedule")

print("5. Communication Analysis")

print("6. Exit")

choice = input("Select an option (1-6): ")

if choice == '1':

while True:

print("\nOptions:")

print("1. Create User")

print("2. Read User")

print("3. Update User")

print("4. Delete User")

print("5. Change Password")

print("6. Exit")

choice = input("Select an option (1-6): ")

if choice == '1':

user\_id = input("Enter User ID: ")

username = input("Enter Username: ")

password = input("Enter Password: ")

email = input("Enter Email: ")

role = input("Enter Role: ")

user\_manager.create\_user(user\_id, username, password, email, role)

elif choice == '2':

user\_id = input("Enter User ID to read: ")

user\_manager.read\_user(user\_id)

elif choice == '3':

user\_id = input("Enter User ID to update: ")

username = input("Enter new Username : ")

password = input("Enter new Password : ")

email = input("Enter new Email : ")

role = input("Enter new Role (leave blank to skip): ")

user\_manager.update\_user(user\_id, username or None, password or None, email or None, role or None)

elif choice == '4':

user\_id = input("Enter User ID to delete: ")

user\_manager.delete\_user(user\_id)

elif choice == '5':

user\_id = input("Enter User ID to change password: ")

new\_password = input("Enter new Password: ")

user\_manager.update\_user(user\_id, password=new\_password)

elif choice == '6':

break

else:

print("Invalid option. Please try again.")

elif choice == '2':

while True:

print("\nCommunication Log Options:")

print("1. Create Log")

print("2. Read Log")

print("3. Update Log")

print("4. Delete Log")

print("5. Exit")

log\_choice = input("Select an option: ")

if log\_choice == '1':

log\_id = input("Enter Log ID: ")

stakeholder\_id = input("Enter Stakeholder ID: ")

communication\_type = input("Enter Communication type (email/phone/text): ")

date = input("Enter Date of communication (YYYY-MM-DD): ")

platform.logs.create\_log(log\_id, stakeholder\_id, communication\_type, date)

elif log\_choice == '2':

log\_id = input("Enter Log ID to read: ")

platform.logs.read\_log(log\_id)

elif log\_choice == '3':

log\_id = input("Enter Log ID: ")

communication\_type = input("Enter new Communication type (leave blank to skip): ")

date = input("Enter new Date (leave blank to skip): ")

platform.logs.update\_log(log\_id, communication\_type or None, date or None)

elif log\_choice == '4':

log\_id = input("Enter Log ID to delete: ")

platform.logs.delete\_log(log\_id)

elif log\_choice == '5':

break

else:

print("Invalid option. Please try again.")

elif choice == '3':

while True:

print("\nFeedback Options:")

print("1. Add Feedback")

print("2. Read Feedback")

print("3. Exit")

feedback\_choice = input("Select an option: ")

if feedback\_choice == '1':

stakeholder\_id = input("Enter Stakeholder ID: ")

feedback\_message = input("Enter Feedback Message: ")

platform.add\_feedback(stakeholder\_id, feedback\_message)

elif feedback\_choice == '2':

stakeholder\_id = input("Enter Stakeholder ID to read feedback: ")

platform.read\_feedback(stakeholder\_id)

elif feedback\_choice == '3':

break

else:

print("Invalid option. Please try again.")

elif choice == '4':

while True:

print("\nSchedule Options:")

print("1. Add Task")

print("2. View Tasks")

print("3. Update Task")

print("4. Exit")

schedule\_choice = input("Select an option: ")

if schedule\_choice == '1':

schedule\_manager.add\_task()

elif schedule\_choice == '2':

schedule\_manager.view\_tasks()

elif schedule\_choice == '3':

schedule\_manager.update\_task()

elif schedule\_choice == '4':

break

else:

print("Invalid option. Please try again.")

elif choice == '5':

stakeholder\_id = input("Enter Stakeholder ID to analyze: ")

platform.analyze\_communication\_effectiveness(stakeholder\_id)

elif choice == '6':

print("Exiting program.")

break

else:

print ("Invalid option. Please try again.")

main\_menu()

print ("Thank you!")

print("Visit Again!")