```python
from typing import List, Dict

# ----- Data Models -----

class Role:
    def __init__(self, name: str, permissions: List[str]):
        self.name = name
        self.permissions = permissions

class User:
    def __init__(self, username: str):
        self.username = username
        self.roles: List[Role] = []
        self.groups: List[str] = []

    def assign_role(self, role: Role):
        if role not in self.roles:
            self.roles.append(role)

    def add_to_group(self, group_name: str):
        if group_name not in self.groups:
            self.groups.append(group_name)

    def has_permission(self, permission: str) -> bool:
        return any(permission in role.permissions for role in self.roles)

class Group:
    def __init__(self, name: str):
        self.name = name
        self.members: List[User] = []

    def add_member(self, user: User):
        if user not in self.members:
            self.members.append(user)
            user.add_to_group(self.name)

# ----- Workflow System -----

class WorkflowStep:
    def __init__(self, name: str, required_role: str):
        self.name = name
        self.required_role = required_role
        self.completed_by: str = None
```

```python
    def complete(self, user: User):
        if any(role.name == self.required_role for role in user.roles):
            self.completed_by = user.username
            print(f"Step '{self.name}' completed by {user.username}")
        else:
            raise PermissionError(f"{user.username} lacks the role '{self.required_role}' to complete
this step.")

class Workflow:
    def __init__(self, steps: List[WorkflowStep]):
        self.steps = steps
        self.current_step = 0

    def advance(self, user: User):
        if self.current_step < len(self.steps):
            step = self.steps[self.current_step]
            step.complete(user)
            self.current_step += 1
        else:
            print("Workflow already completed.")

# ----- Example Setup -----

# Define roles
admin_ro_
```