User & Group Management – Report

1. Introduction

User and group management is a foundational task in any Active Directory (AD) environment, enabling secure and structured access to IT resources. This project involved a series of practical administrative operations using PowerShell and GUI tools within a Windows Server domain environment. The operations ranged from user account creation to group management and password policy enforcement. Maintaining consistency through structured naming conventions, role-based group assignments, and proper account lifecycle handling is essential for ensuring security, manageability, and compliance in enterprise networks.

The tasks were executed in a controlled AD setup, simulating real-world administrative scenarios. The use of PowerShell scripting ensured automation, repeatability, and accuracy in managing user identities and permissions. These tasks reflect core administrative responsibilities including account provisioning, access control, and enforcement of security policies such as password expiration and change requirements.

2. Objective

The primary objective of this project was to implement essential user and group management practices in Active Directory. This included creating standardized user accounts, managing password resets with enforced updates at login, disabling unused accounts to improve security, creating and managing role-based security groups, and verifying group memberships. Each step was designed to simulate common administrative duties that contribute to a secure and well-managed IT environment.

Through these tasks, the goal was to demonstrate the application of best practices in identity and access management using both command-line and graphical tools. The project ensures that accounts are not only created and maintained correctly but also that they adhere to password and group policy standards, laying the groundwork for a secure and scalable domain structure.

Key goals include:

- **Standardized User Account Creation**
  To establish a structured and consistent user identity format, accounts were created using a First.Last naming convention. This improves clarity and traceability across the organization. Consistent naming also supports easier automation and integration with scripts and tools.

- **Password Security Enforcement**
  The project enforced password resets and expiration policies to uphold domain security standards. Users were required to change passwords at first login, ensuring credentials remain private. This step simulates real-world onboarding and credential hygiene.

- **Role-Based Access via Security Groups**
  Three security groups were created (Finance_RW, HR_Admins, IT_Helpdesk) to reflect departmental roles. Users were assigned based on job function, enforcing the principle of least privilege. This aligns user permissions with business needs and supports scalable access control.

- **Deactivation of Inactive Accounts**
  Inactive accounts were identified and disabled to prevent unauthorized access. This reduces the attack surface within the network. Proper documentation ensured accountability for each action taken.

- **Verification and Auditing via PowerShell**
  PowerShell commands like Get-ADGroupMember and Get-ADUser were used to verify account and group settings. This enabled real-time validation and quick troubleshooting. Auditing enhances administrative transparency and compliance tracking.

3. Methodology

The project was conducted in a controlled Active Directory environment using Windows Server tools and PowerShell scripting. User accounts were created and managed through command-line automation to ensure consistency and efficiency. Group memberships, password policies, and account statuses were verified using AD-specific cmdlets, with documentation maintained for auditing purposes. A combination of GUI and CLI methods was used for flexibility and validation.

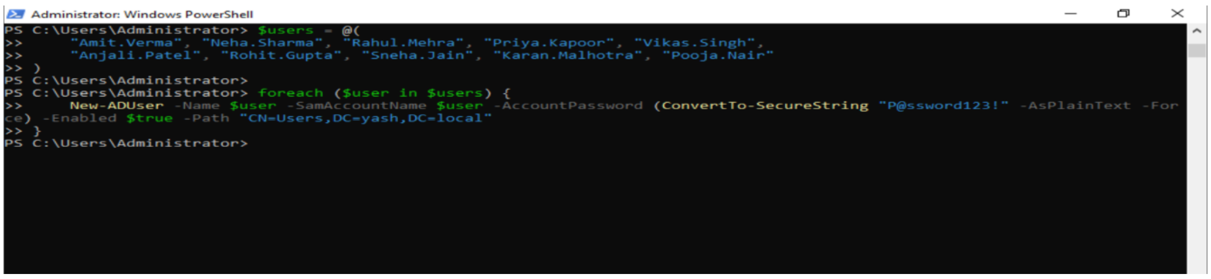
3.1 User Account Creation and Verification

This phase focuses on the structured creation of user accounts in Active Directory using PowerShell. It ensures consistency in naming and lays the foundation for future group assignments and security policy enforcement. CLI-based verification follows to confirm successful creation.

- **Create User Accounts**

  PowerShell was used to create 10 users following the First.Last naming convention. This ensured uniformity and automated provisioning of user attributes like name, username, and password. The script also enabled the accounts after creation.

  (Displaying the PowerShell script output after user creation.)

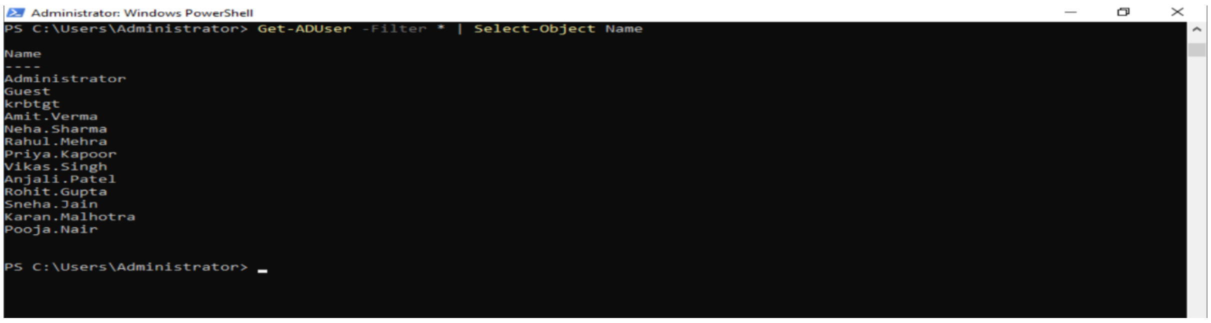


```
Administrator: Windows PowerShell
PS C:\Users\Administrator> $users = @(
>>     "Amit.Verma", "Neha.Sharma", "Rahul.Mehra", "Priya.Kapoor", "Vikas.Singh",
>>     "Anjali.Patel", "Rohit.Gupta", "Sneha.Jain", "Karan.Malhotra", "Pooja.Nair"
>> )
PS C:\Users\Administrator>
PS C:\Users\Administrator> foreach ($user in $users) {
>>     New-ADUser -Name $user -SamAccountName $user -AccountPassword (ConvertTo-SecureString "P@ssword123!" -AsPlainText -For
ce) -Enabled $true -Path "CN=Users,DC=yash,DC=local"
>> }
PS C:\Users\Administrator>
```

- **Verify Created Users**

  Using Get-ADUser, all newly created users were verified. This included checking properties like SamAccountName and UserPrincipalName to ensure naming standards and domain bindings were correct.

  (PowerShell output listing created users.)



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADUser -Filter * | Select-Object Name

Name
----
Administrator
Guest
krbtgt
Amit.Verma
Neha.Sharma
Rahul.Mehra
Priya.Kapoor
Vikas.Singh
Anjali.Patel
Rohit.Gupta
Sneha.Jain
Karan.Malhotra
Pooja.Nair

PS C:\Users\Administrator> _
```

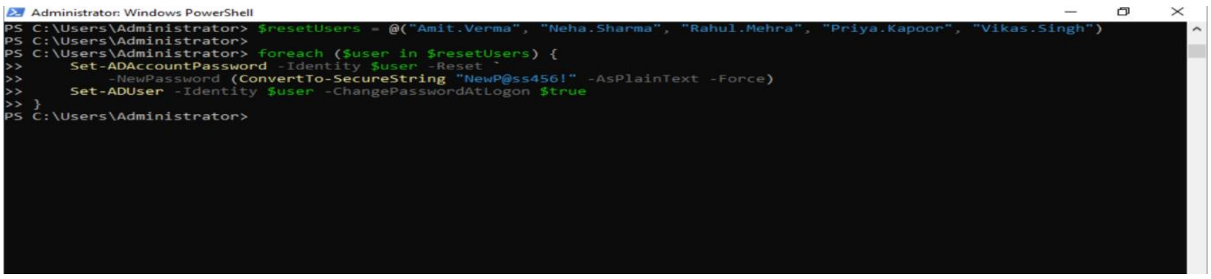
3.2 Password Management

In this phase, passwords for selected users were reset to a default secure value, and password change enforcement was applied. These actions ensured compliance with security policies and that users would update credentials upon login.

- **Reset Passwords & Enforce Change at Logon**

  Passwords for 5 selected accounts were reset to a default secure value. The Set-ADUser cmdlet was used with the -ChangePasswordAtLogon flag to mandate a change at the next sign-in.

  (Output of password reset and enforcement commands.)

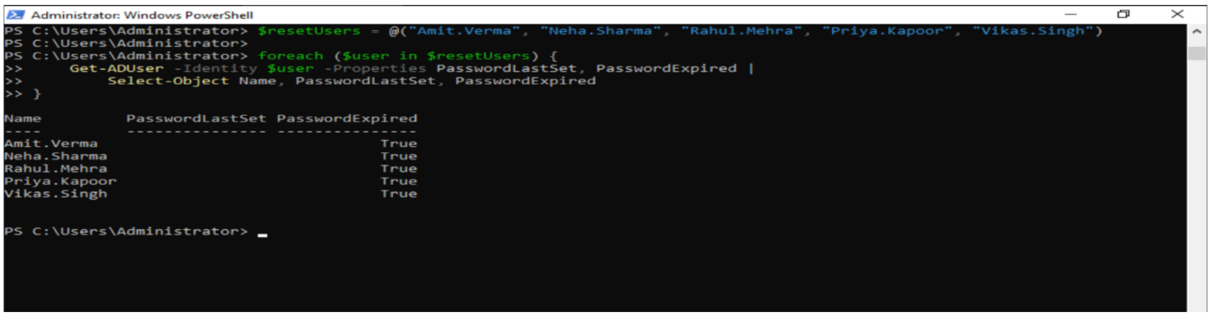


```
Administrator: Windows PowerShell
PS C:\Users\Administrator> $resetUsers = @("Amit.Verma", "Neha.Sharma", "Rahul.Mehra", "Priya.Kapoor", "Vikas.Singh")
PS C:\Users\Administrator>
PS C:\Users\Administrator> foreach ($user in $resetUsers) {
>>     Set-ADAccountPassword -Identity $user -Reset `
>>         -NewPassword (ConvertTo-SecureString "NewP@ss456!" -AsPlainText -Force)
>>     Set-ADUser -Identity $user -ChangePasswordAtLogon $true
>> }
PS C:\Users\Administrator>
```

- **Verify Password Reset & Policy Enforcement**

  The reset was verified using Get-ADUser to check PasswordLastSet and related properties. This confirms whether the reset was successful and enforcement is in place.

  (Display of PasswordLastSet (users haven't logged in to update their passwords yet.) and related properties.)



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> $resetUsers = @("Amit.Verma", "Neha.Sharma", "Rahul.Mehra", "Priya.Kapoor", "Vikas.Singh")
PS C:\Users\Administrator>
PS C:\Users\Administrator> foreach ($user in $resetUsers) {
>>     Get-ADUser -Identity $user -Properties PasswordLastSet, PasswordExpired |
>>         Select-Object Name, PasswordLastSet, PasswordExpired
>> }

Name         PasswordLastSet PasswordExpired
----         --------------- ---------------
Amit.Verma                              True
Neha.Sharma                             True
Rahul.Mehra                             True
Priya.Kapoor                            True
Vikas.Singh                             True


PS C:\Users\Administrator> _
```

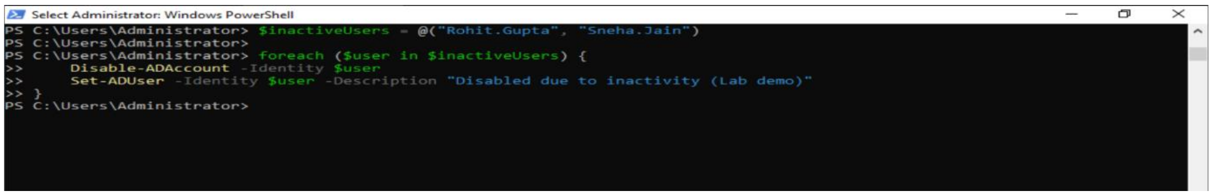
3.3 Account Deactivation and Documentation

Two inactive or unnecessary user accounts were disabled to prevent unauthorized access. The reasons for deactivation were documented, supporting proper audit practices and administrative accountability.

- **Disable 2 Inactive Accounts**

  Two users who were inactive or unused were disabled using the Disable-ADAccount cmdlet. This is a critical step in limiting access for dormant accounts.

  (Output showing successful account disabling.)



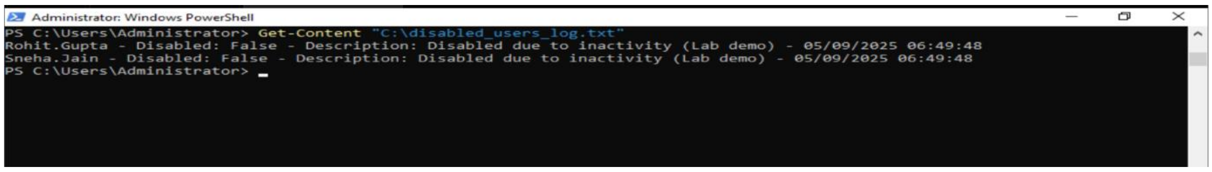
Select Administrator: Windows PowerShell

```
PS C:\Users\Administrator> $inactiveUsers = @("Rohit.Gupta", "Sneha.Jain")
PS C:\Users\Administrator>
PS C:\Users\Administrator> foreach ($user in $inactiveUsers) {
>>     Disable-ADAccount -Identity $user
>>     Set-ADUser -Identity $user -Description "Disabled due to inactivity (Lab demo)"
>> }
PS C:\Users\Administrator>
```

- **Document Deactivation Reasons**

  For accountability, reasons for disabling accounts were logged. This step supports audit trails and administrative transparency.

  (Text file showing disabled accounts and reasons.)



Administrator: Windows PowerShell

```
PS C:\Users\Administrator> Get-Content "C:\disabled_users_log.txt"
Rohit.Gupta - Disabled: False - Description: Disabled due to inactivity (Lab demo) - 05/09/2025 06:49:48
Sneha.Jain - Disabled: False - Description: Disabled due to inactivity (Lab demo) - 05/09/2025 06:49:48
PS C:\Users\Administrator> _
```

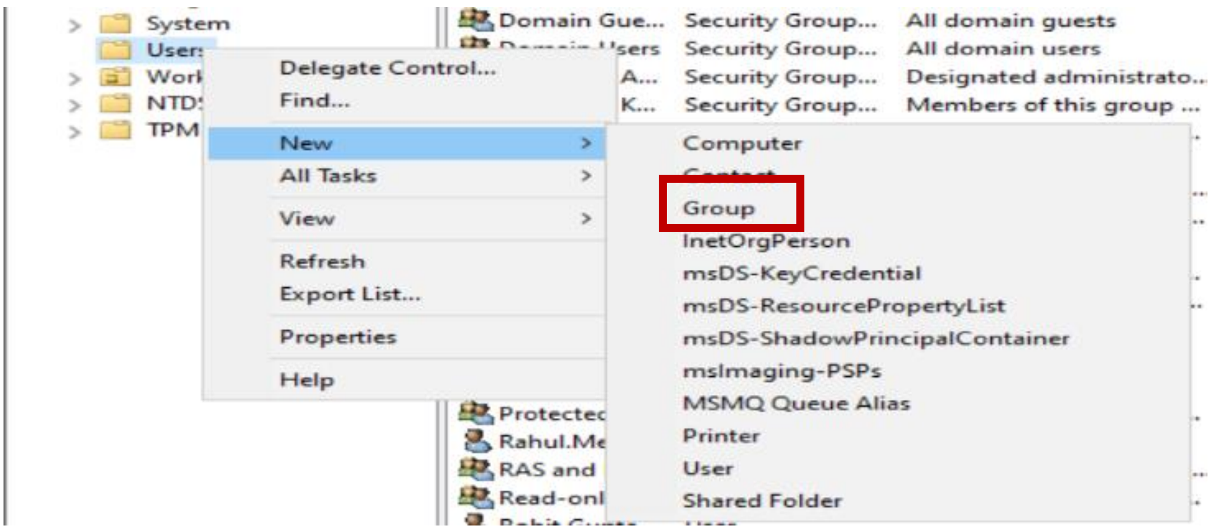
3.4 Group Creation (GUI-Based)

Security groups were created using the ADUC graphical interface. Groups such as Finance_RW, HR_Admins, and IT_Helpdesk were set up to manage access and permissions based on departmental roles.

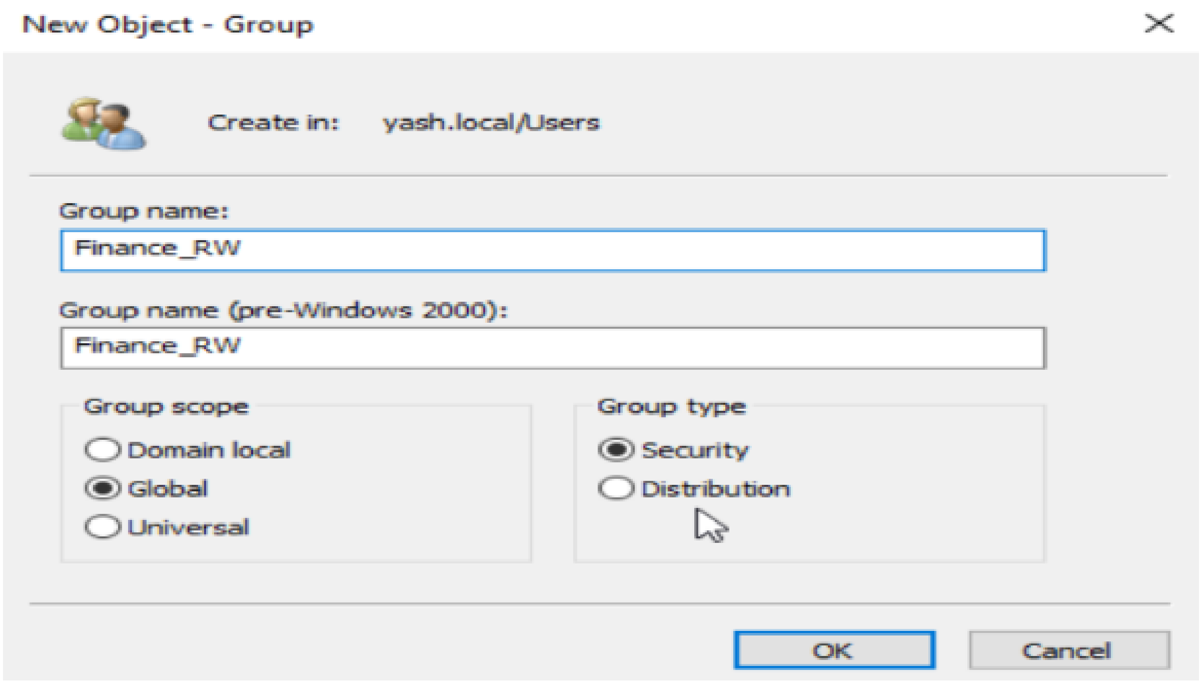
- **Create Security Groups**

  Three security groups were created: Finance_RW, HR_Admins, and IT_Helpdesk. This classification allows for precise permission control across departments.

(ADUC interface showing created groups.)


System
Users
Work
NTDS
TPM
Domain Gue... Security Group... All domain guests
Domain Users Security Group... All domain users
A... Security Group... Designated administrato...
K... Security Group... Members of this group ...
Delegate Control...
Find...
New
All Tasks
View
Refresh
Export List...
Properties
Help
Computer
Contact
Group
InetOrgPerson
msDS-KeyCredential
msDS-ResourcePropertyList
msDS-ShadowPrincipalContainer
msImaging-PSPs
MSMQ Queue Alias
Printer
User
Shared Folder
Protected
Rahul.Me
RAS and
Read-onl

(Group created -Finance_RW)


New Object - Group

Create in: yash.local/Users

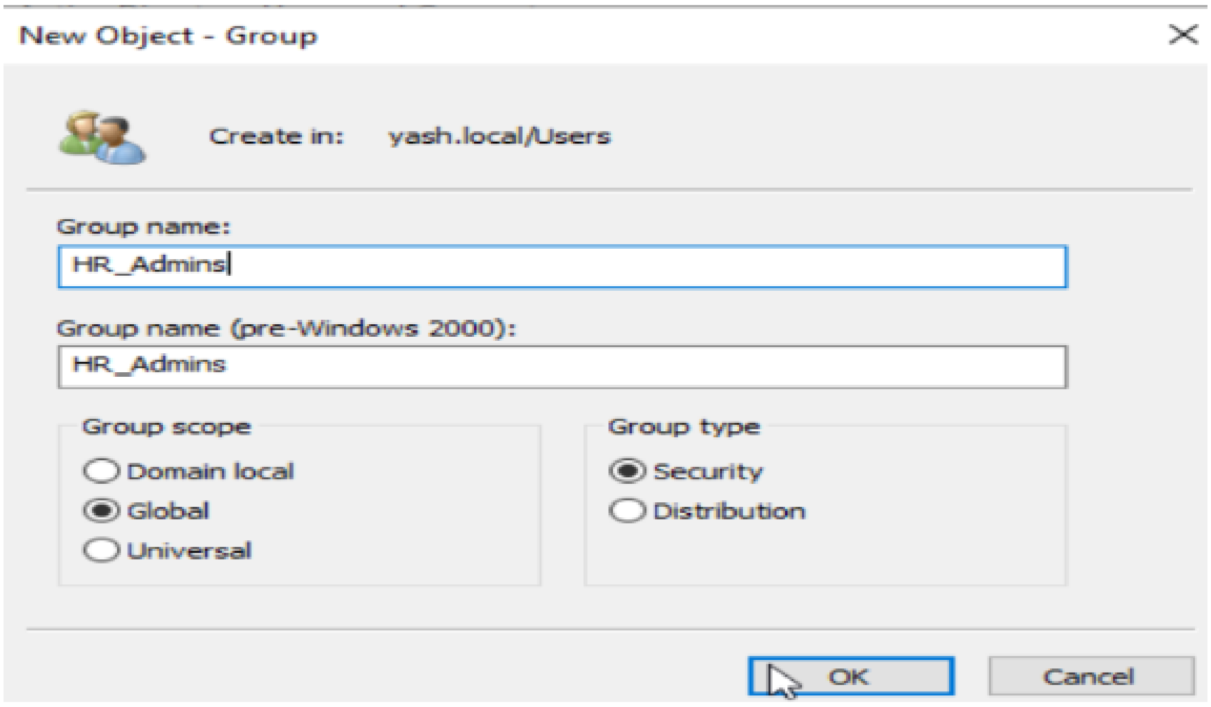
Group name:
Finance_RW

Group name (pre-Windows 2000):
Finance_RW

Group scope: ( ) Domain local [✓ Global] ( ) Universal

Group type: [✓ Security] ( ) Distribution

(Group created -HR_Admins)


New Object - Group

Create in: yash.local/Users

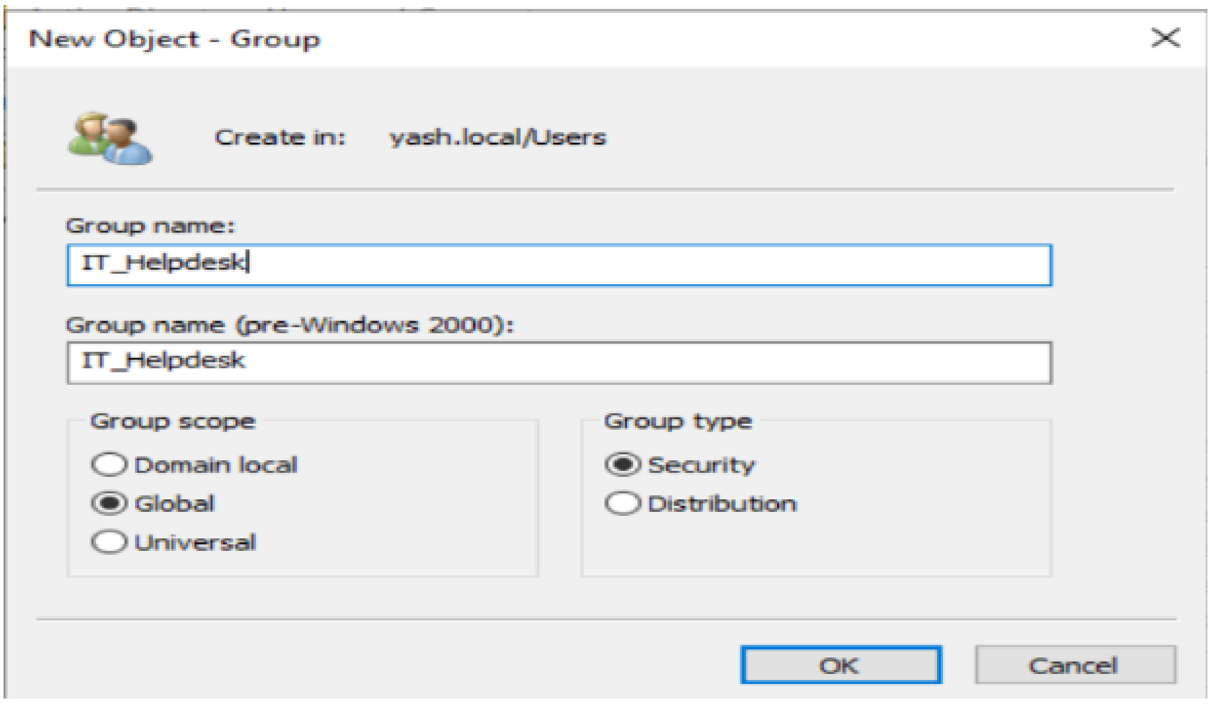
Group name: HR_Admins

Group name (pre-Windows 2000): HR_Admins

Group scope: ○ Domain local ● Global ○ Universal

Group type: ● Security ○ Distribution

(Group created -IT_Helpdesk)


New Object - Group

Create in: yash.local/Users

Group name: IT_Helpdesk

Group name (pre-Windows 2000): IT_Helpdesk

Group scope: ○ Domain local ● Global ○ Universal

Group type: ● Security ○ Distribution

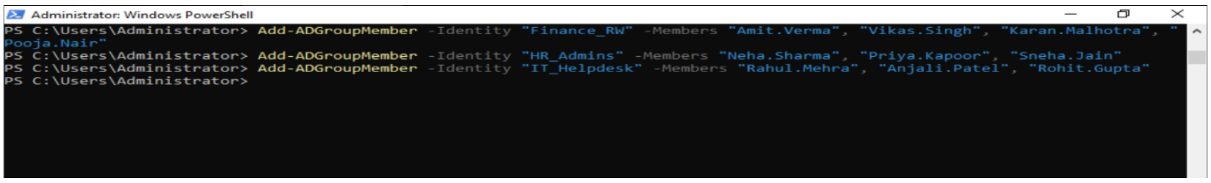
3.5 User-Group Assignment

Users were added to the appropriate security groups based on their job functions. This streamlined access control and ensured that users had permissions relevant to their department.

- **Add Users to Groups**

  Using Add-ADGroupMember, users were added to their respective groups. This automates group membership management and ties users to role-based access.

  (PowerShell output confirming group membership additions.)



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Add-ADGroupMember -Identity "Finance_RW" -Members "Amit.Verma", "Vikas.Singh", "Karan.Malhotra", "
Pooja.Nair"
PS C:\Users\Administrator> Add-ADGroupMember -Identity "HR_Admins" -Members "Neha.Sharma", "Priya.Kapoor", "Sneha.Jain"
PS C:\Users\Administrator> Add-ADGroupMember -Identity "IT_Helpdesk" -Members "Rahul.Mehra", "Anjali.Patel", "Rohit.Gupta"
PS C:\Users\Administrator>
```

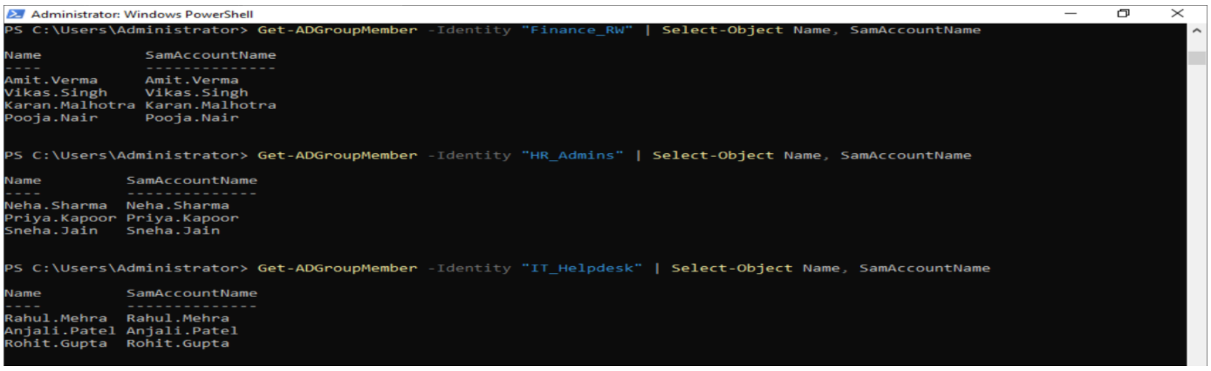
3.6 Final Verification and Auditing

Group memberships were verified using PowerShell to ensure accuracy. This final check confirmed that all users were assigned correctly and that the security structure reflected organizational requirements.

- **Verify Group Membership**

  Get-ADGroupMember was used to confirm that all users were in the correct groups. This ensures that the security group structure reflects organizational roles accurately.

  (Output listing group members.)



```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-ADGroupMember -Identity "Finance_RW" | Select-Object Name, SamAccountName

Name           SamAccountName
----           --------------
Amit.Verma     Amit.Verma
Vikas.Singh    Vikas.Singh
Karan.Malhotra Karan.Malhotra
Pooja.Nair     Pooja.Nair

PS C:\Users\Administrator> Get-ADGroupMember -Identity "HR_Admins" | Select-Object Name, SamAccountName

Name         SamAccountName
----         --------------
Neha.Sharma  Neha.Sharma
Priya.Kapoor Priya.Kapoor
Sneha.Jain   Sneha.Jain

PS C:\Users\Administrator> Get-ADGroupMember -Identity "IT_Helpdesk" | Select-Object Name, SamAccountName

Name         SamAccountName
----         --------------
Rahul.Mehra  Rahul.Mehra
Anjali.Patel Anjali.Patel
Rohit.Gupta  Rohit.Gupta
```

4. Results and Findings

- **Successful User Creation via PowerShell**
  All 10 users were created successfully using PowerShell with a consistent naming format. This demonstrated efficient bulk user provisioning using command-line tools. The accounts were structured with appropriate naming conventions like First.Last. This improves organization and clarity in the Active Directory environment.

- **Password Reset and Logon Policy Applied**
  Passwords for five selected accounts were reset, and enforcement of password change at next login was configured. However, since the users didn't log in, PasswordLastSet remained empty. This highlighted the importance of interactive login to complete the policy cycle.

- **Inactive Account Management**
  Two user accounts were identified as inactive and disabled using PowerShell. The reason for their deactivation was documented for audit purposes. This step showcased good practice in lifecycle management of user accounts.

- **Group Creation and Role Assignment**
  Three groups—Finance_RW, HR_Admins, and IT_Helpdesk—were created, and users were assigned based on departmental roles. This helped establish role-based access control (RBAC) to streamline permission management.

- **Verification and Audit through CLI Tools**
  All group memberships and user account statuses were verified using PowerShell commands like Get-ADUser and Get-ADGroupMember. These verifications confirmed that all configurations were accurately applied as per the project plan.

5. Recommendations

- **Enforce Initial User Login Post-Creation**
  Users should be instructed to log in after account creation to trigger password change enforcement. This ensures policies like ChangePasswordAtLogon are completed and reflected in directory attributes.

- **Integrate Logging and Audit Mechanisms**
  Implement logging of account creation, password resets, and group changes. Maintaining logs ensures traceability and aids in future audits or troubleshooting.

- **Automate Group Assignment by Department**
  Use scripts or automation tools to dynamically assign users to groups based on attributes like Department. This minimizes human error and speeds up onboarding.

- **Periodic Review of Disabled Accounts**
  Regularly review and clean up disabled or stale accounts. This improves security by reducing the attack surface and maintaining a clean directory.

- **Explore GUI and CLI Hybrid Management**
  For long-term administration, blend GUI-based tools with PowerShell for flexibility and control. While CLI is powerful for bulk tasks, GUI can simplify one-off or visual configurations.

6. Conclusion

This project aimed to establish a foundational understanding and implementation of Active Directory user and group management, leveraging both CLI (PowerShell) and GUI-based tools. The tasks included creating user accounts with standardized naming conventions, resetting passwords, enforcing logon policies, disabling inactive accounts, setting up role-based security groups, and verifying configurations using command-line queries. Through these activities, we demonstrated critical administrative practices that are essential for any enterprise-level network environment.

A key outcome of this project was the ability to streamline and automate routine administrative tasks using PowerShell, significantly improving efficiency over manual GUI operations. The project also emphasized the importance of access control by organizing users into clearly defined security groups aligned with departmental roles. Additionally, we gained insights into the role of password policies, user lifecycle management, and verification procedures to ensure accurate implementation.

While the technical objectives were met, the experience also underscored the need for consistent follow-up actions, such as user logins post-password reset, and documentation of account changes. These elements are crucial for enforcing policy compliance and ensuring system integrity.
Overall, this project successfully enhanced practical skills in Active Directory management, fostered a deeper understanding of Windows Server administration, and prepared the groundwork for more advanced topics like Group Policy Objects (GPO), auditing, and access delegation in a secure domain environment.