Naveen Ravichandran

8982481

April 14, 2024

User onboarding and mail details

Table of Contents

[Research Task: 2](#_Toc164025913)

[Main Script: 4](#_Toc164025914)

[Implementation of Script: 12](#_Toc164025915)

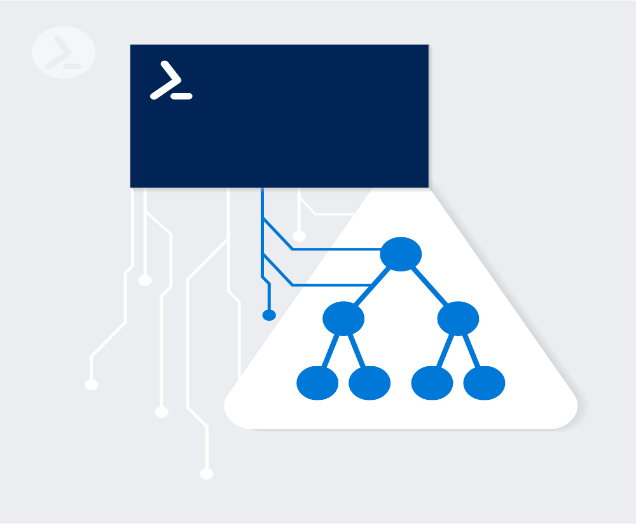
[Conclusion: 17](#_Toc164025916)

[References: 18](#_Toc164025917)

# Research Task:

As a scripting language, PowerShell has many modules to manage and deploy multiple ecosystems and technologies like Exchange, SQL, etc. Automation tasks include managing virtual machines, generating reports, transferring files across servers, and managing Active Directory users. **(P, 2022)**

Active Directory (AD) is Microsoft’s proprietary directory service. It runs on Windows Server, stores data as objects, and enables administrators to manage the users, groups, or applications. **(Wesley Chai, n.d.)**



Automating the onboarding process significantly reduces the time required to complete administrative tasks. Adding one or two users to the Active Directory may be an easy process, but on a large scale, manually adding users to the Active Directory is a time-consuming process for the IT department, especially in categorizing the users with specific departments, permissions, etc. We can use the PowerShell scripting to automate tasks for this type of consequence. In the meantime, HR can use the click-to-run script whenever needed without disturbing any other departments or interrupting during essential projects.

Automation ensures consistency across the onboarding process. Consistency helps to reduce the risk of errors or oversights. Automating onboarding streamlines processes and provides a more engaging and interactive experience for users like HR and new hires. This eliminates duplicate data entry, ensures the information flows smoothly between the system, and creates an efficient HR ecosystem. Moreover, details such as Name and Credentials for login to the organization are sent to new employees after a second of account creation, and it is very efficient for the organization and the new user. Automating the onboarding process can lead to cost savings for organizations.

I create an interactive tool for onboarding new users. Some pre-requirements, such as a Domain Controller and a Windows 10 Machine connected to the same domain, are required for efficient implementation. We have to enable the Windows features in our Windows 10 machine called **Active Directory Module for Windows PowerShell**. **(Brown, 2022)**

I created a PowerShell GUI for more efficient interaction with the script; even the non-technical guy could use this interface. I used Windows Forms to create the label, textbox, and button, implemented the Combo Box form for dropdown, and used **Get-ADOrganizationalUnit** to locate the Organizational Unit. So, HR can add the user with simple inputs such as name and department. This script creates a random password for the user login. To share the information with others (new employees or HR), I used the **Send-MailMessage** cmdlet to send mail messages to the user. I chose the SMTP server to send mail without issues in this work. For mail, the basic format is From, To, Subject, and body, and I used all of these to construct the proper mail. In this project, I slightly incorporated the HTML codes for the body part of the mail.

# Main Script:

<#

.SYNOPSIS

Script to add users to Active Directory via GUI and send email notification

.DESCRIPTION

This script creates a GUI form to input user details such as first name, last name, and department

It generates a random password and assigns an email address based on the first name and last name details

The script then adds the user to Active Directory and sends an email notification about the new user creation

.NOTES

Author: Naveen Ravichandran

Date: 14-04-2024

#>

# Load the necessary modules and assemblies

Add-Type -AssemblyName System.Windows.Forms

Add-Type -AssemblyName System.Drawing

Import-Module ActiveDirectory

#Variables and configurations

# Receiving domain credentials in an encrypted manner

$domainid = "naveen.local"

$username="naveen.local\helper"

$newencryptedpwd2 = get-content -path "C:\Users\itadmin\Desktop\Convertedpw\scriptpwd.txt"

$decryptedpw1 = ConvertTo-SecureString -String $newencryptedpwd2

$credentials =New-Object System.Management.Automation.PSCredential $username , $decryptedpw1

# setting and encrypting gmail (external) password

#$emailapppwd =Read-Host -AsSecureString

#$encryptedemailpwd = ConvertFrom-SecureString -SecureString $emailapppwd

#Set-Content -Path "C:\Users\itsupport\Desktop\Convertedpw\emailapppwd.txt" -value $encryptedemailpwd

$gmailusername = "naveenr1812@gmail.com"

$emailpwd =Get-Content -Path "C:\Users\itadmin\Desktop\Convertedpw\emailapppwd.txt"

$secureemailpwd =ConvertTo-SecureString -String $emailpwd

$outgoingemailcred =New-Object System.Management.Automation.PSCredential $gmailusername, $secureemailpwd

# Defining the form

$form = New-Object System.Windows.Forms.Form

$form.Text = "Add User to Active Directory"

$form.Size = New-Object System.Drawing.Size(400, 300)

$form.StartPosition = "CenterScreen"

# Defining form elements

$label1 = New-Object System.Windows.Forms.Label

$label1.Location = New-Object System.Drawing.Point(10, 20)

$label1.Size = New-Object System.Drawing.Size(150, 20)

$label1.Text = "Enter user details:"

# Defining the form for First name

$label2 = New-Object System.Windows.Forms.Label

$label2.Location = New-Object System.Drawing.Point(30, 50)

$label2.Size = New-Object System.Drawing.Size(70, 20)

$label2.Text = "First Name:"

# Defining form elements

$textboxFirstName = New-Object System.Windows.Forms.TextBox

$textboxFirstName.Location = New-Object System.Drawing.Point(110, 50)

$textboxFirstName.Size = New-Object System.Drawing.Size(100, 20)

# Defining the form for Last name

$label3 = New-Object System.Windows.Forms.Label

$label3.Location = New-Object System.Drawing.Point(230, 50)

$label3.Size = New-Object System.Drawing.Size(70, 20)

$label3.Text = "Last Name:"

# Defining form elements

$textboxLastName = New-Object System.Windows.Forms.TextBox

$textboxLastName.Location = New-Object System.Drawing.Point(310, 50)

$textboxLastName.Size = New-Object System.Drawing.Size(100, 20)

# Defining the form for Department

$label4 = New-Object System.Windows.Forms.Label

$label4.Location = New-Object System.Drawing.Point(30, 80)

$label4.Size = New-Object System.Drawing.Size(70, 20)

$label4.Text = "Department:"

# Define dropdown box for selecting departments

$dropdown = New-Object System.Windows.Forms.ComboBox

$dropdown.Location = New-Object System.Drawing.Point(110, 80)

$dropdown.Size = New-Object System.Drawing.Size(150, 20)

Get-ADOrganizationalUnit -filter \* | ForEach-Object {$dropdown.Items.Add($\_.Name)}

$form.Controls.AddRange(@($label4,$dropdown))

# Defining form elements

$textboxGroup = New-Object System.Windows.Forms.TextBox

$textboxGroup.Location = New-Object System.Drawing.Point(110, 80)

$textboxGroup.Size = New-Object System.Drawing.Size(150, 20)

# Defining the form for Generated Password

$label5 = New-Object System.Windows.Forms.Label

$label5.Location = New-Object System.Drawing.Point(30, 110)

$label5.Size = New-Object System.Drawing.Size(100, 30)

$label5.Text = "Generated Password:"

# Defining form elements

$passwordTextbox = New-Object System.Windows.Forms.TextBox

$passwordTextbox.Location = New-Object System.Drawing.Point(150, 110)

$passwordTextbox.Size = New-Object System.Drawing.Size(150, 20)

$passwordTextbox.ReadOnly = $true

# Defining the form for Generate button

$generatePasswordButton = New-Object System.Windows.Forms.Button

$generatePasswordButton.Location = New-Object System.Drawing.Point(310, 109)

$generatePasswordButton.Size = New-Object System.Drawing.Size(80, 22)

$generatePasswordButton.Text = "Generate"

# Defining the form for Email address

$label6 = New-Object System.Windows.Forms.Label

$label6.Location = New-Object System.Drawing.Point(30, 140)

$label6.Size = New-Object System.Drawing.Size(100, 20)

$label6.Text = "Email Address:"

# Defining form elements

$textboxEmailAddress = New-Object System.Windows.Forms.TextBox

$textboxEmailAddress.Location = New-Object System.Drawing.Point(150, 140)

$textboxEmailAddress.Size = New-Object System.Drawing.Size(150, 20)

$textboxEmailAddress.ReadOnly = $true

# Defining the form for Add User button

$button = New-Object System.Windows.Forms.Button

$button.Location = New-Object System.Drawing.Point(150, 200)

$button.Size = New-Object System.Drawing.Size(80, 30)

$button.Text = "Add User"

# Add form elements to the form

$form.Controls.Add($label1)

$form.Controls.Add($label2)

$form.Controls.Add($label3)

$form.Controls.Add($label4)

$form.Controls.Add($label5)

$form.Controls.Add($label6)

$form.Controls.Add($textboxFirstName)

$form.Controls.Add($textboxLastName)

$form.Controls.Add($dropdown)

$form.Controls.Add($textboxGroup)

$form.Controls.Add($passwordTextbox)

$form.Controls.Add($textboxEmailAddress)

$form.Controls.Add($generatePasswordButton)

$form.Controls.Add($button)

# Defining event handler for the "Generate Password" button click event

$generatePasswordButton.Add\_Click({

# Generate a random password

$passwordLength = 16 # Set the length of the Password

$validChars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()-\_'

$password = ""

for ($i = 0; $i -lt $passwordLength; $i++) {

$password += $validChars[(Get-Random -Minimum 0 -Maximum $validChars.Length)]

}

# Display the generated password in the password textbox

$passwordTextbox.Text = $password

})

# Defining event handler for the "Add User" button click event

$button.Add\_Click({

# Retrieve user input

$firstName = $textboxFirstName.Text

$lastName = $textboxLastName.Text

$OU = $dropdown.Text

$generatedPassword = $passwordTextbox.Text

# Check if required fields are not empty

if ($firstName -eq "" -or $lastName -eq "" -or $OU -eq "") {

[System.Windows.Forms.MessageBox]::Show("Please fill in all required fields.", "Error")

} else {

# Generate email address

$emailAddress = "$firstName.$lastName@$domainid"

$textboxEmailAddress.Text = $emailAddress

# Send email through SMTP using html for body part

try {

$Subject = "Account Details for $firstName $lastName"

$body = @"

<html>

<p>Hello,<br><br>

<p>New User account has been Created</p>

<p>Firstname: $firstName </p>

<p>Lastname: $lastName </p>

<p>EmailAddress: $emailAddress </p>

<p>Password: $generatedPassword </p>

<p>Thanks</p>

"@

$SmtpServer = "smtp.gmail.com"

$smtpPort = "587"

$senderemail= "naveenr1812@gmail.com"

$receiveremail= "naveenu192709@gmail.com"

#Send email

Send-MailMessage -from $senderemail -to $receiveremail -Subject $Subject -Body $body -BodyAsHtml -SmtpServer $SmtpServer -Port $smtpPort -UseSsl -Credential $outgoingemailcred

$username = $firstName.Substring(0,1) + $lastName

$ouPath = "OU=$OU,DC=naveen,DC=local" # Change this to your OU path

New-ADUser -Name "$firstName $lastName" -GivenName $firstName -Surname $lastName -SamAccountName $username -EmailAddress $emailAddress -AccountPassword (ConvertTo-SecureString -AsPlainText $generatedPassword -Force) -Enabled $true -Path $ouPath

[System.Windows.Forms.MessageBox]::Show("User created successfully.", "Success")

[System.Windows.Forms.MessageBox]::Show("Email sent successfully.")

} catch {

[System.Windows.Forms.MessageBox]::Show("Failed to create user: $\_", "Error")

}

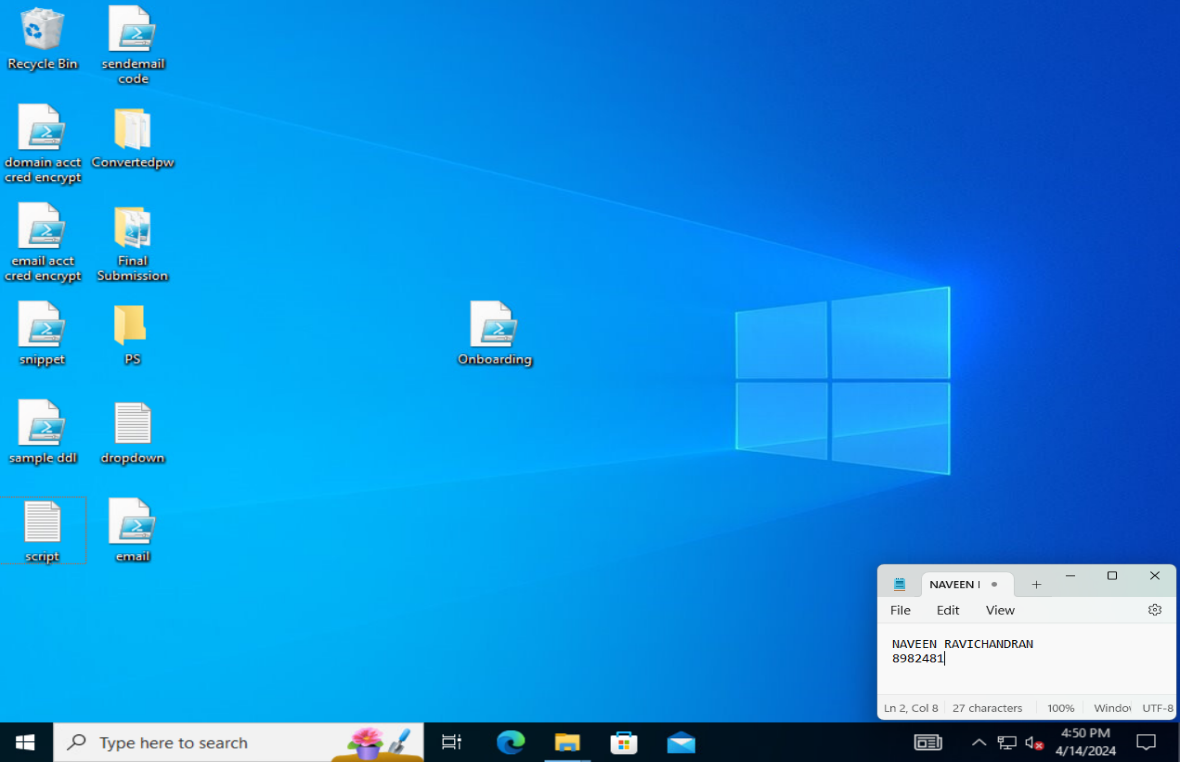
}

})

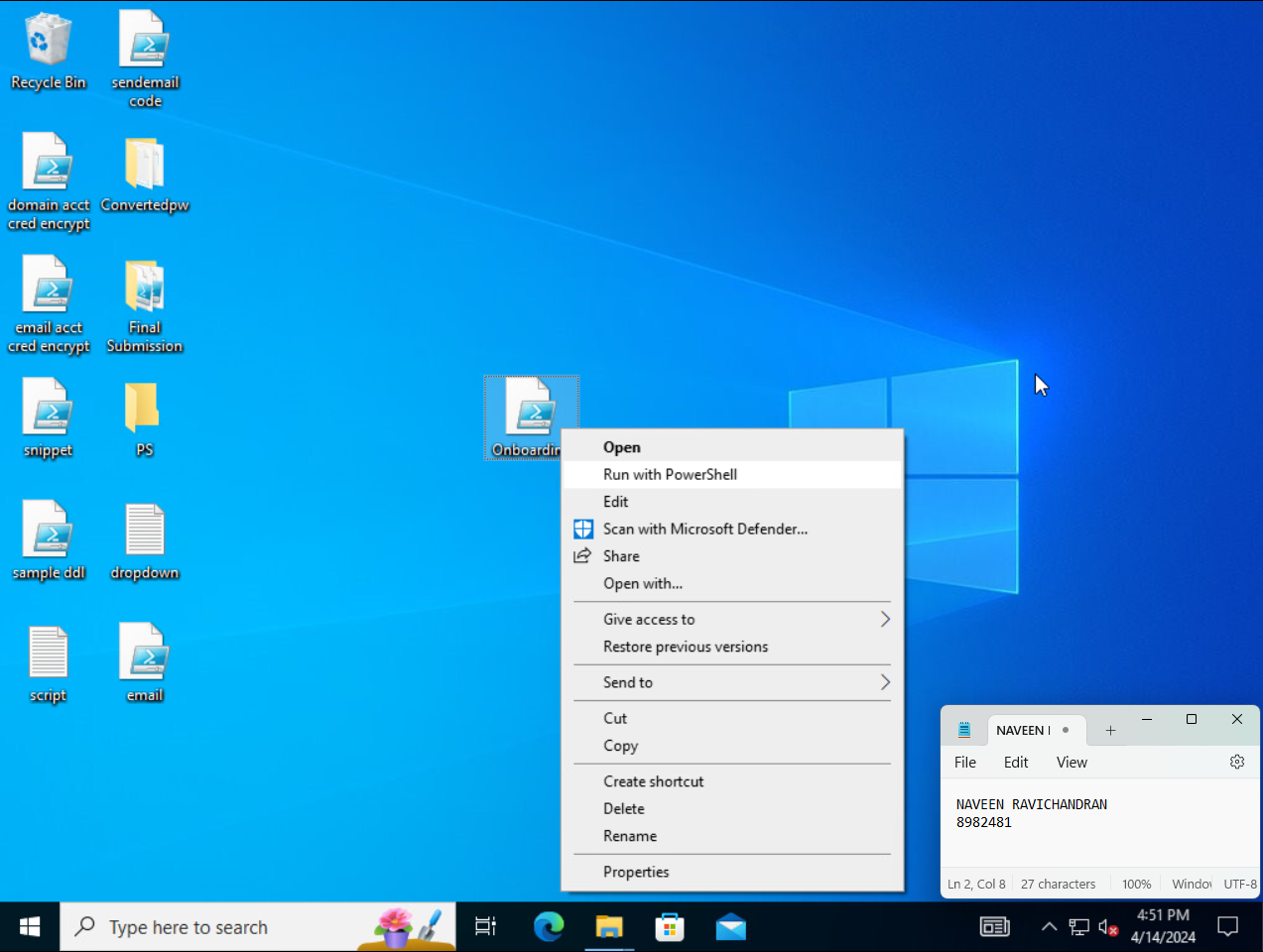
# Display the form

$form.ShowDialog() | Out-Null

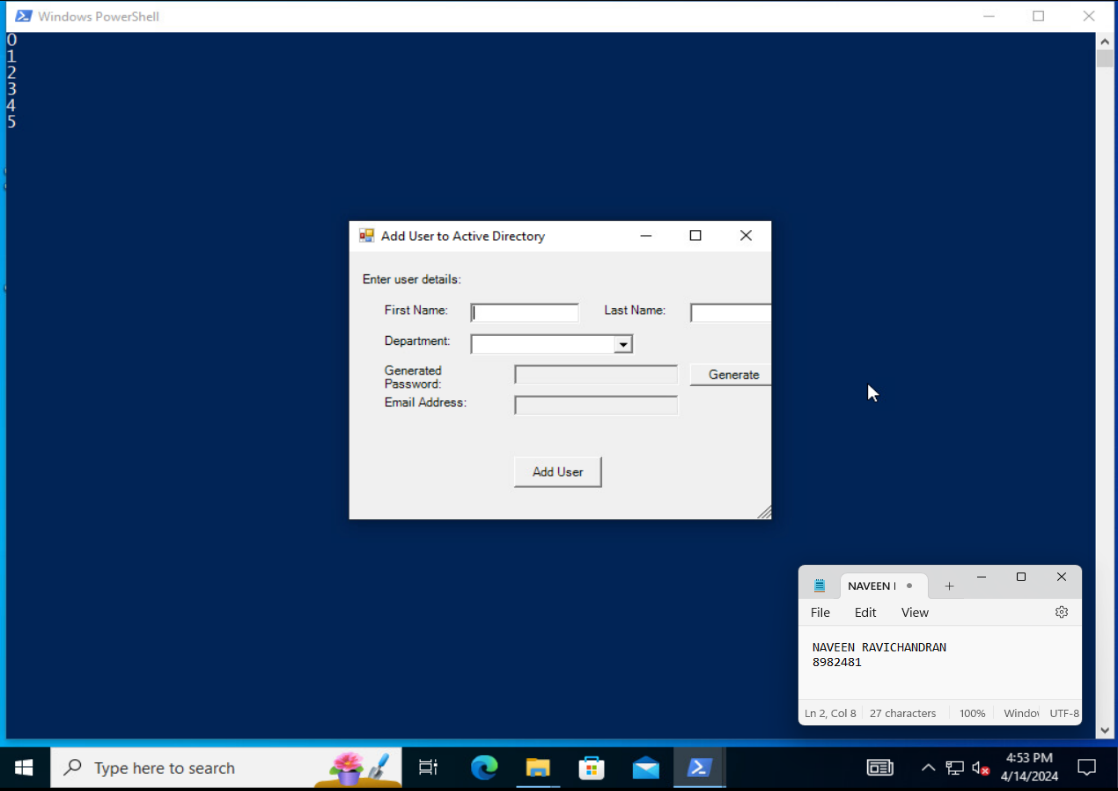
# Implementation of Script:



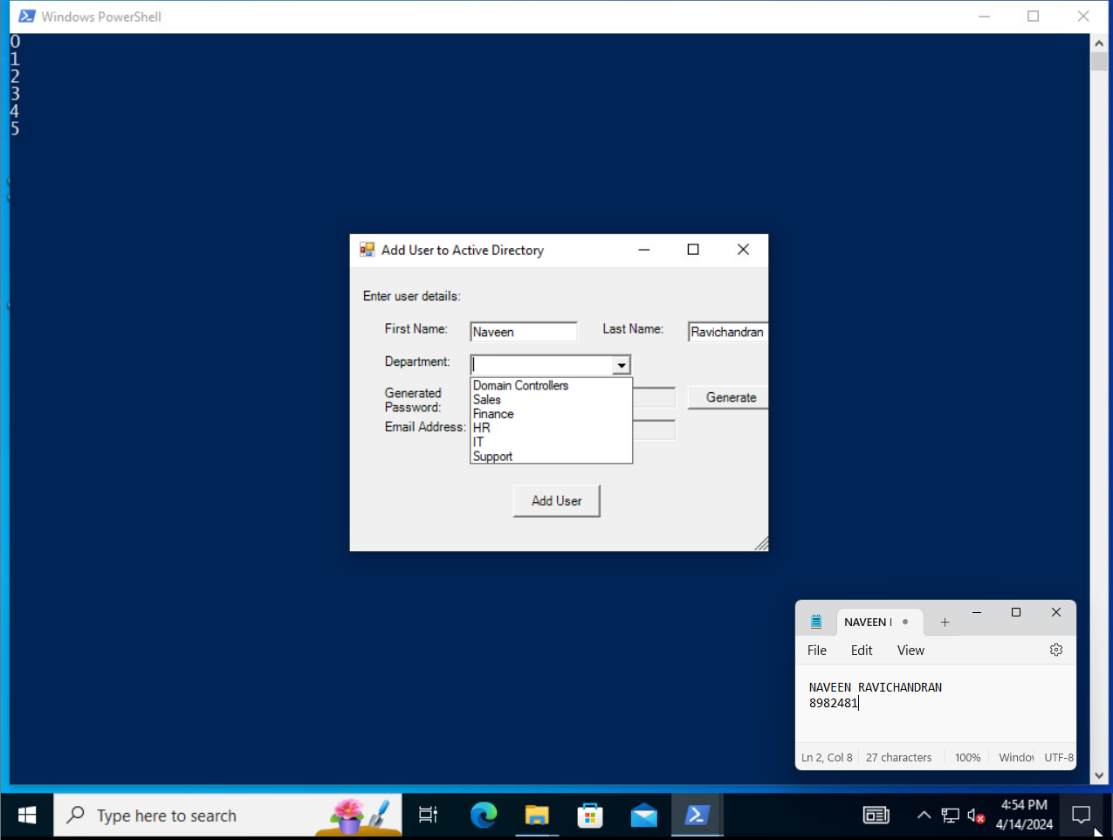
* In this above image, the Onboarding is the main script file.



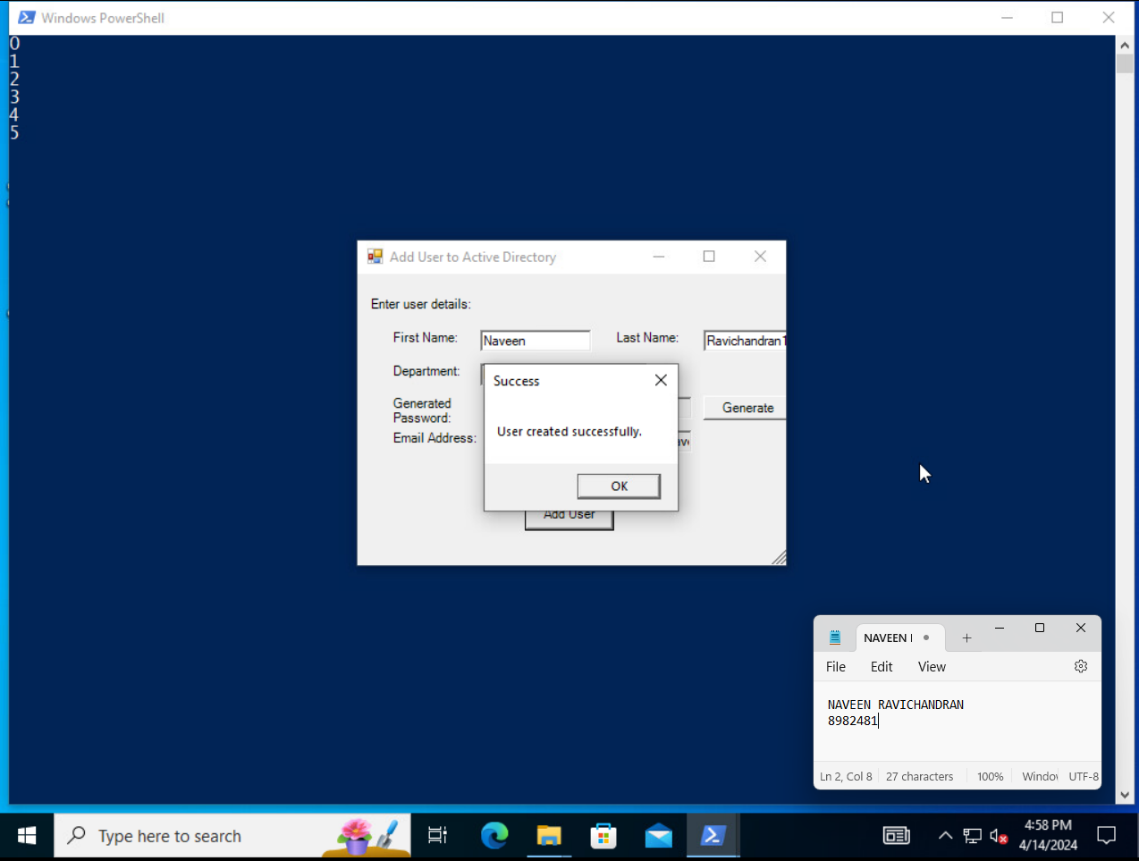
* To implement, user have to right-click the file and select Run with PowerShell



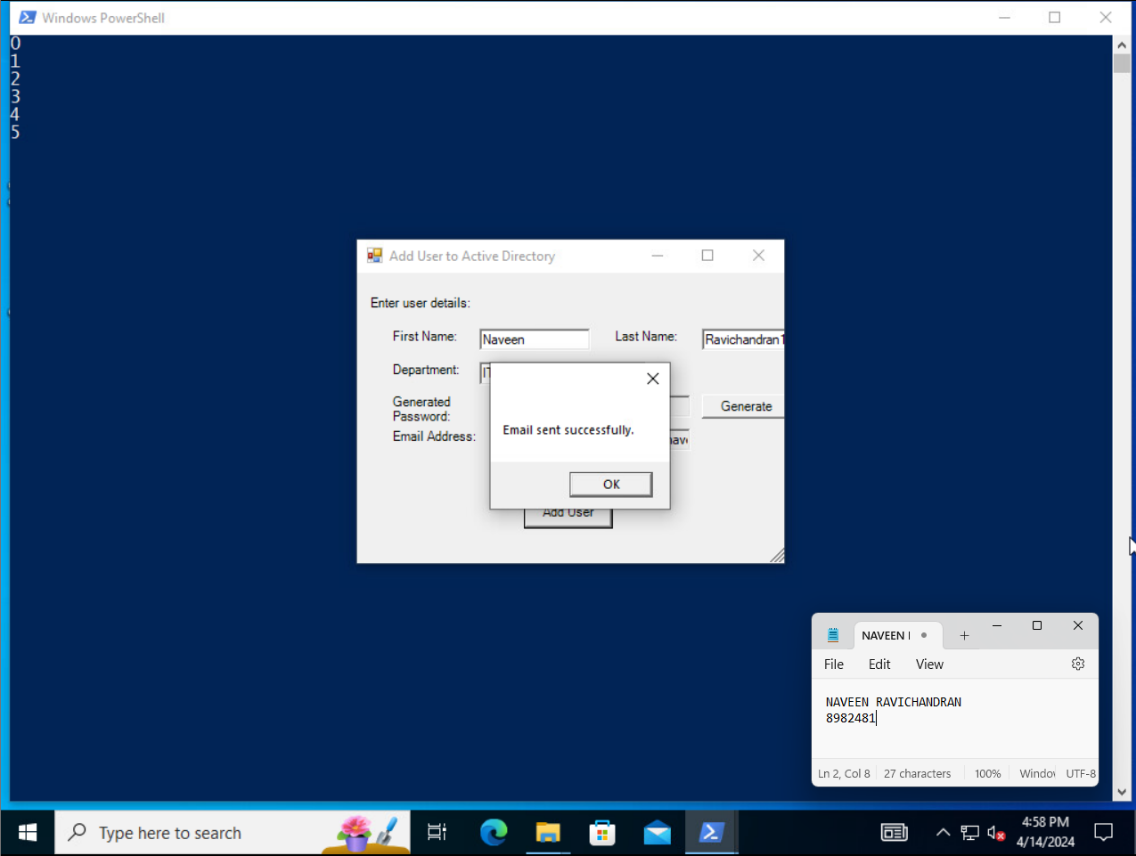
* In the next few seconds, this GUI will appear and we have to fill the required fields.



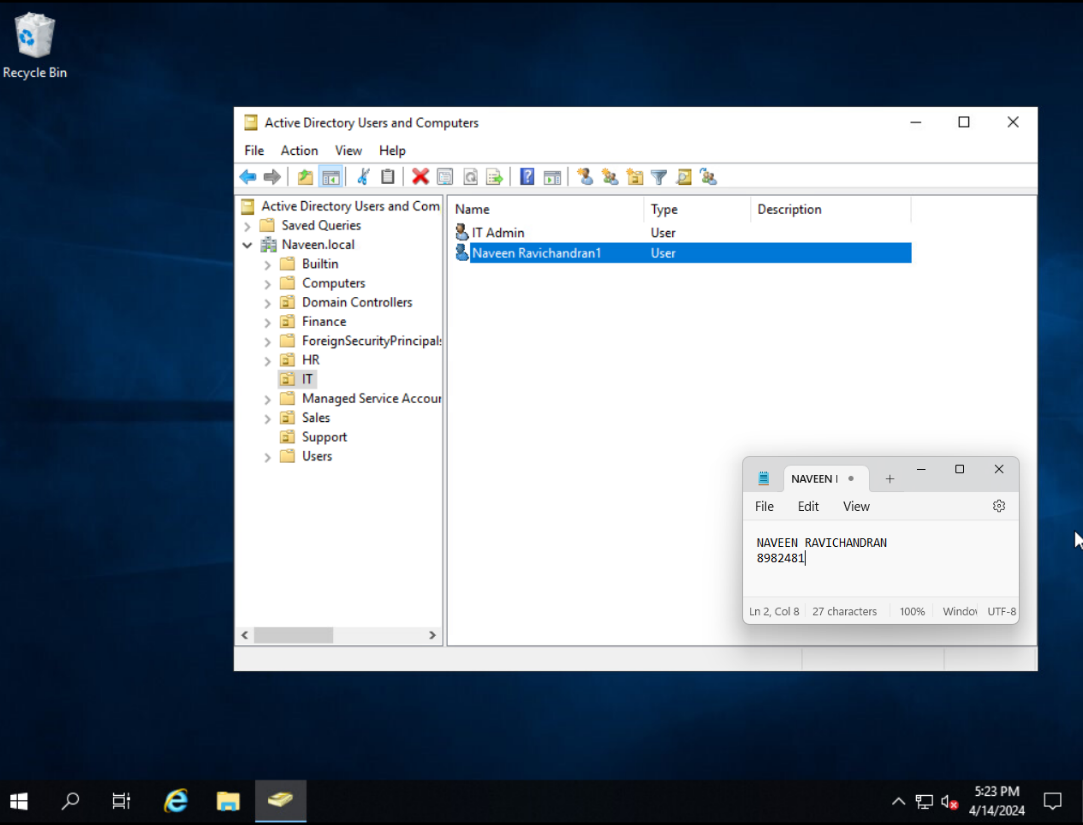
* In the dropdown list, we can easily select the department. After this Generate button is there to create the random password for user.



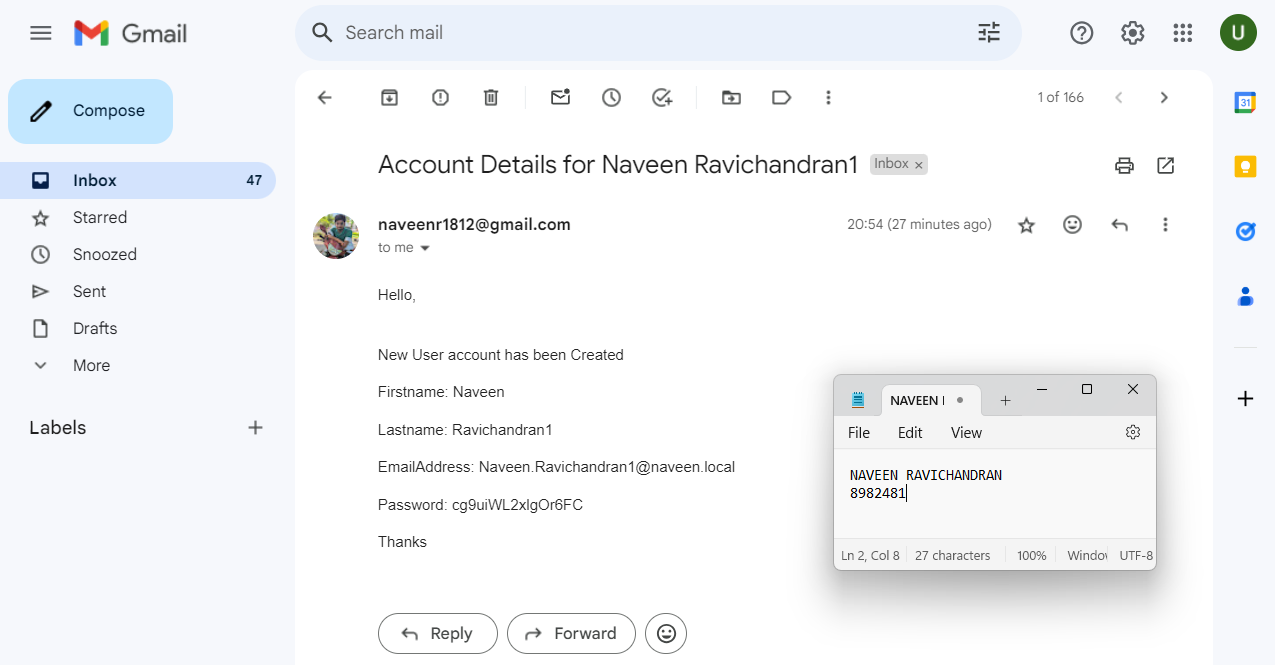
* Then the user created successfully.



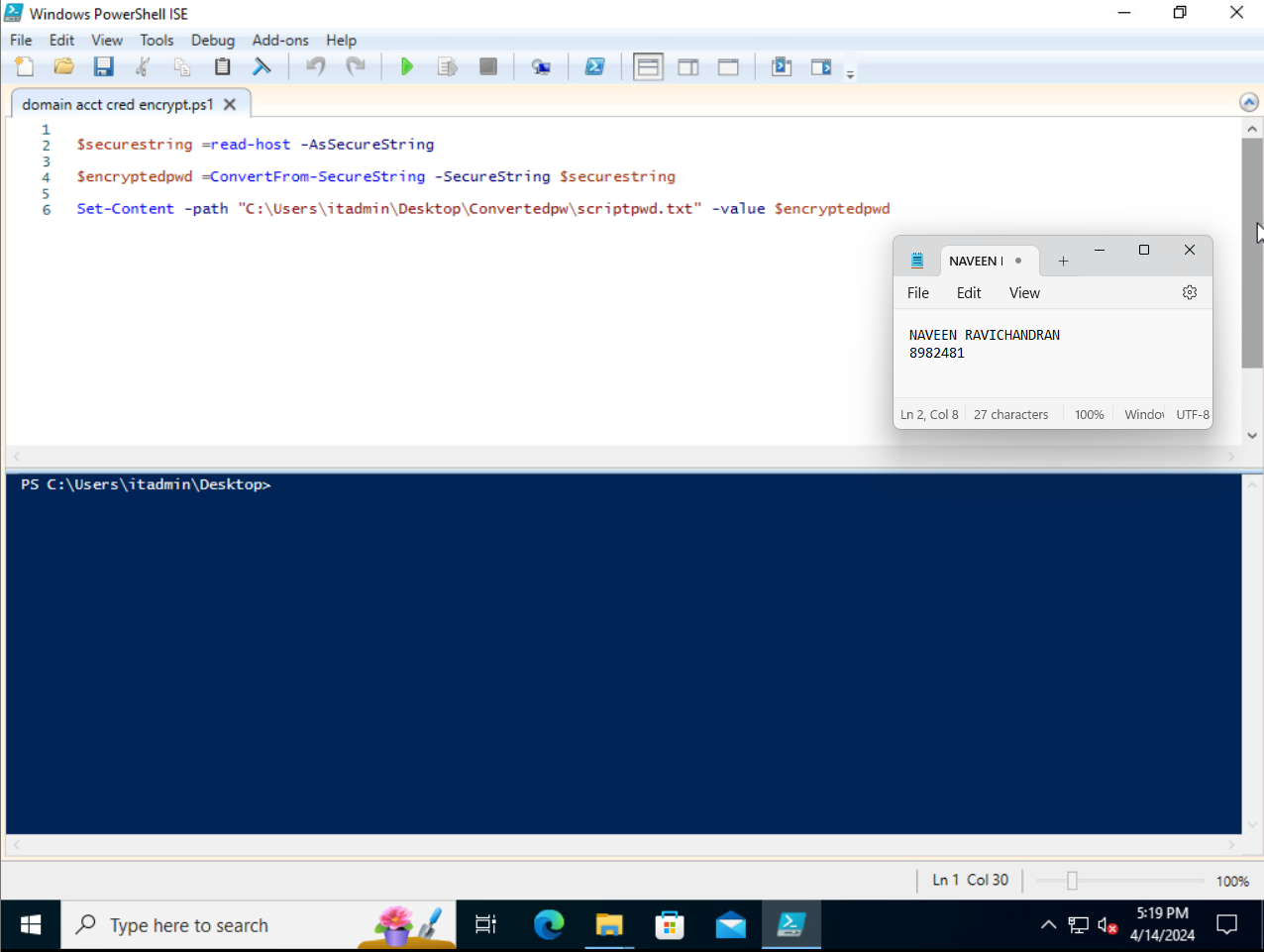
* At the mean time, email sent to user successfully.



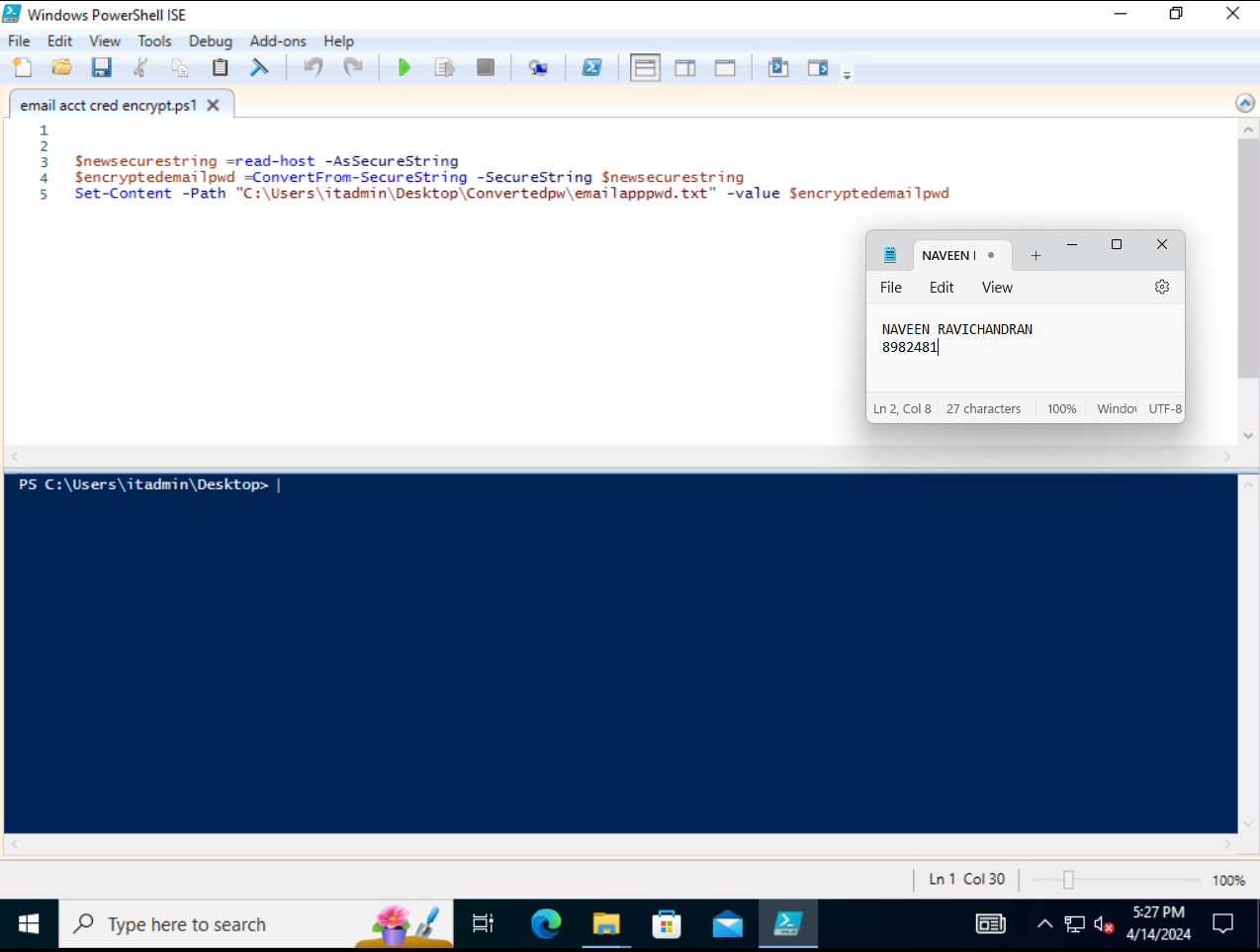
* User created successfully to the Active Directory.



* Email with user credentials also received by the user (In this case, I used my mail id to implement this)



* I Used the above commands to convert the plain text to secure string for my domain.



* I used the above command to save my email credential

# Conclusion:

For this particular script, once the variables and other pre requirements satisfied, it is very easy to add the users to active directory without need of IT support. This script helps in reducing the time in which an account needs to be created. Automation of this task also helps in reducing the workload for the system administration team and opens up their bandwidth to focus on more critical tasks.

# References:

* Ricardo P. (2022, July 1). (Pg. No:2) How to Easily Automate Tasks with PowerShell. <https://www.serveracademy.com/blog/how-to-easily-automate-tasks-with-powershell/>
* Wesley Chai, Alexander S. Gillis. (n.d.). Active directory. (Pg. No:2) <https://www.techtarget.com/searchwindowsserver/definition/Active-Directory#:~:text=Active%20Directory%20(AD)%20is%20Microsoft's,device%20such%20as%20a%20printer>.
* Jeff Brown. (2022, January 10). (Pg. No:3) How to Install and Import Active Directory PowerShell Module. <https://www.varonis.com/blog/powershell-active-directory-module>
* Ashley Gwilliam. (2023, August 10). Active Directory Pros and Cons. <https://jumpcloud.com/blog/active-directory-pros-cons>