```powershell
<#
  .SYNOPSIS
    The purpose of this file is to set extensionAttribute14 to flag users who should
    take the hospital's HIPAA training.

  .DESCRIPTION
    The users who take the hospital's HIPAA training are sent to the microsoft account
    every morning around 8am. This email has a subject of "IdentityIQ Report". For the time being,
    this file is manually saved to the $PSCriptRoot/FilesFromHospital directory. A file is being
    worked on that will automatically save files like this sent by email.

    Once the file is saved, this file goes through and sets extensionAttribute14 for those users.

  .PARAMETER HospitalFilPath
    Specifies the path to search for the latest hospital HIPAA file

  .PARAMETER Testing
    Used for additional logging features if development work is being done on this file.

  .DESCRIPTION
    Users who should take the hospital HIPAA training are sent to the microsoft@hsc.wvu.edu
    account once per day around 8am. This file is stored in the FilesFromHospital directory.


  .NOTES
    Set-HospitalHIPAATraining.ps1
    Last Modified by: Jeff Brusoe
    Last Modified: December 3, 2020
#>

[CmdletBinding()]
param (
  [ValidateNotNullOrEmpty()]
  [string]$HospitalFilePath = "$PSScriptRoot\FilesFromHospital\",

  [switch]$Testing
)

#Configure environment
Set-HSCEnvironment
Write-Output "Hospital File Path: $HospitalFilePath"

#Geneate list of users from hospital file
if (Test-Path $HospitalFilePath)
{
  $HospitalFile = Get-HSCLastFile -DirectoryPath $HospitalFilePath
  Write-Output "Hospital File: $HospitalFile"

  try {
    $WVUMUsers = Import-Csv $HospitalFile -ErrorAction Stop
  }
  catch {
    Write-Warning "Unable to open hospital file. Program is exiting."
    Invoke-HSCExitCommand -ErrorCount $Error.Count
  }
}
else {
  Write-Warning "Unable to access file from hospital. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

$FoundCount = 0
$NotFoundCount = 0

#Now loop through list of users
foreach ($WVUMUser in $WVUMUsers)
{
  $Username = $WVUMUser."User ID".Trim()
  Write-Output $("Username: " + $Username)

  $LDAPFilter = "(samaccountname=$Username)"
  Write-Output "LDAPFilter: $LDAPFilter"

  try
  {
    $GetADUserParams = @{
      LDAPFilter = $LDAPFilter
      Properties = "extensionAttribute14"
      ErrorAction = "Stop"
    }

    $ADUser = Get-ADUser @GetADUserParams

    Write-Output "User Found"
    Write-Output $("Sam Account Name: " + $ADUser.SamAccountName)
    Write-Output $("extensionAttribute14: " + $ADUser.extensionAttribute14)
  }
  catch {
    Write-Warning "Unable to find AD user"
    $ADUser = $null
  }

  if ($null -eq $ADUser)
  {
    Write-Output "User not found"
    $NotFoundCount++
  }
  elseif ($ADUser.extensionAttribute14 -eq "HospitalHIPAA")
  {
    Write-Output "extensionAttribute14 is already set."
    $FoundCount++
  }
  else
  {
    Write-Output "User found - Need to set extensionAttribute14"
    $ADUser | Set-ADUser -Clear extensionAttribute14

    Start-Sleep -s 1
    $ADUser | Set-ADUser -Add @{extensionAttribute14="HospitalHIPAA"}

    $FoundCount++
  }

  Write-Output "********************"

  if ($Error.Count -gt 0 -AND $Testing) {
    Invoke-HSCExitCommand -ErrorCount $Error.Count
  }
}

Write-Output "Found: $FoundCount"
Write-Output "Not Found: $NotFoundCount"

Invoke-HSCExitCommand -ErrorCount $Error.Count
```