TraceRoute Script

```
# Function to perform ping and traceroute
function Get-ConnectionInfo {
    param (
        [string]$ComputerName
    )

    # Perform ping to check if the computer is online
    $pingResult = Test-Connection -ComputerName $ComputerName -Count 1 -Quiet

    if ($pingResult) {
        # Perform traceroute
        $tracerouteResult = tracert $ComputerName | Select-Object -Skip 2

        # Calculate the hop count
        $hopCount = ($tracerouteResult.Count - 4)

        # Return the results based on hop count
        if ($hopCount -ge 5) {
            return @("Remote", $hopCount)
        } else {
            return @("On-site", $hopCount)
        }
    } else {
        return @("Offline or does not exist", 0)
    }
}

# Function to output colored text
function Write-ColoredText {
    param (
        [string]$Text,
        [string]$Color
    )
    Write-Host $Text -ForegroundColor $Color
}

# Specify the full path to the text file
$filePath = "C:\Users\jwn2cpt\Desktop\LocationFinderPREP.txt"

# Initialize hash tables to group results
$offlineUsers = @{}
```

```
$remoteUsers = @{}
$onsiteUsers = @{}

if (Test-Path $filePath) {
   # Read the content of the text file
   $fileContent = Get-Content -Path $filePath

   foreach ($line in $fileContent) {
      # Extract workstation and user information
      $workstation, $user = $line -split ' - '

      # Get the network status for the user
      $networkStatus, $hopCount = Get-ConnectionInfo -ComputerName $workstation

      # Group results based on network status
      switch ($networkStatus) {
         "Remote" { $remoteUsers[$user] = $workstation }
         "On-site" { $onsiteUsers[$user] = $workstation }
         "Offline or does not exist" { $offlineUsers[$user] = $workstation }
      }
   }
} else {
   Write-ColoredText "File not found at $filePath" "Red"
   return
}

# Display grouped results
Write-Output "Location Status Report"
Write-Output "----------------------"

Write-ColoredText "`n--- Offline Users ---" "Red"
foreach ($user in $offlineUsers.Keys) {
   Write-Host "$user"
}

Write-ColoredText "`n--- Remote Users ---" "Yellow"
foreach ($user in $remoteUsers.Keys) {
   Write-Host "$user"
}

Write-ColoredText "`n--- Onsite Users ---" "Green"
foreach ($user in $onsiteUsers.Keys) {
   Write-Host "$user"
}
```

```
# Note: The following section is redundant and can be removed if not needed.
$filePath2 = "C:\Users\jwn2cpt\Desktop\LocationFinderPREP.txt"

if (Test-Path $filePath2) {
    # Filter lines that are not blank and do not start with "#"
    $computerNames = Get-Content -Path $filePath2 | Where-Object { $_ -notmatch '^#' -and $_
-ne '' }
} else {
    Write-ColoredText "File not found at $filePath2" "Red"
    return
}
```

IP Matching Script

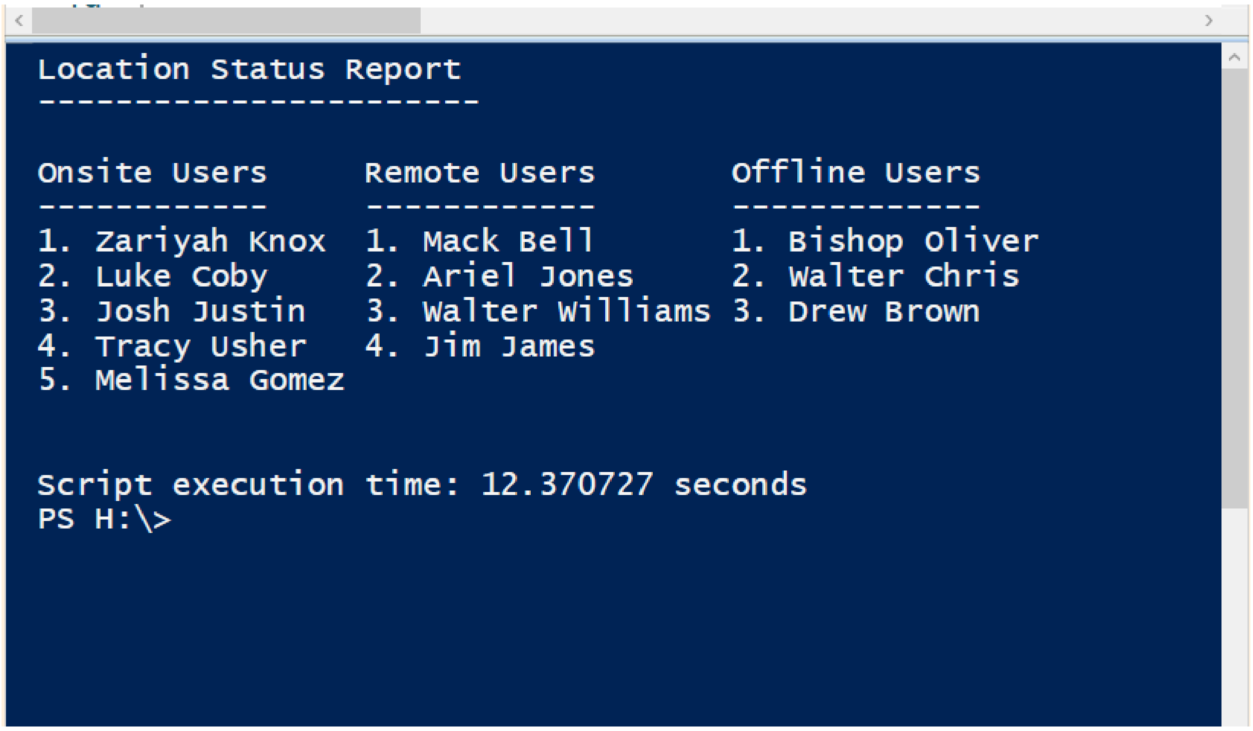
```
# Sleeping for 5 seconds to display the banner
Start-Sleep -Seconds 5

# Clearing the console screen
Clear-Host

# Function to get the location status of a computer
function Get-LocationStatus {
    param (
        [string]$computerName
    )

    # Testing connection to the computer
    $pingResult = Test-Connection -ComputerName $computerName -Count 1 -ErrorAction
SilentlyContinue

    if ($pingResult) {
        $ipAddress = $pingResult.IPV4Address.IPAddressToString

        # Checking if the IP address indicates onsite or remote location
        if ($ipAddress -like '*10.224*') {
            Write-Output "Onsite"
        } else {
            Write-Output "Remote"
        }
    } else {
        Write-Output "Offline"
    }
}

# Path to the text file containing computer names and user names
$textFilePath = "C:\Users\jwn2cpt\Desktop\LocationFinderPREP.txt"

# Arrays to store users based on location status
$onlineUsers = @()
$remoteUsers = @()
$offlineUsers = @()

# Read computer names and user names from the text file
$computerUserPairs = Get-Content -Path $textFilePath | ForEach-Object {
    $parts = ($_ -split ' - ', 2).Trim()
    [PSCustomObject]@{
        ComputerName = $parts[0]
        UserName = $parts[1]
    }
}

# Loading message
Write-Host "Loading..."

# Start timer to measure script execution time
$startTime = Get-Date

# Iterate through each computer and user pair
foreach ($pair in $computerUserPairs) {
    $locationStatus = Get-LocationStatus -computerName $pair.ComputerName

    # Sorting users based on location status
    if ($locationStatus -eq "Onsite") {
        $onlineUsers += $pair.UserName
    } elseif ($locationStatus -eq "Remote") {
        $remoteUsers += $pair.UserName
    } else {
        $offlineUsers += $pair.UserName
    }
```

```
}

# Stop timer
$executionTime = (Get-Date) - $startTime

# Clear loading message
Clear-Host

# Determine the maximum count among the arrays
$maxCount = $onlineUsers.Count, $remoteUsers.Count, $offlineUsers.Count | Measure-Object
-Maximum | Select-Object -ExpandProperty Maximum

# Combine the arrays into a table format without the "User Number" column
$resultTable = @()

for ($i = 0; $i -lt $maxCount; $i++) {
    $resultTable += [PSCustomObject]@{
        "Onsite Users"  = if ($i -lt $onlineUsers.Count) { "$($i + 1). $($onlineUsers[$i])" } else { '' }
        "Remote Users"  = if ($i -lt $remoteUsers.Count) { "$($i + 1). $($remoteUsers[$i])" } else { ''
}
        "Offline Users" = if ($i -lt $offlineUsers.Count) { "$($i + 1). $($offlineUsers[$i])" } else { '' }
    }
}

# Output the results in a table-like format without the "User Number" column
Write-Output "Location Status Report"
Write-Output "-----------------------"
$resultTable | Format-Table

# Display execution time
Write-Host "Script execution time: $($executionTime.TotalSeconds) seconds"
```

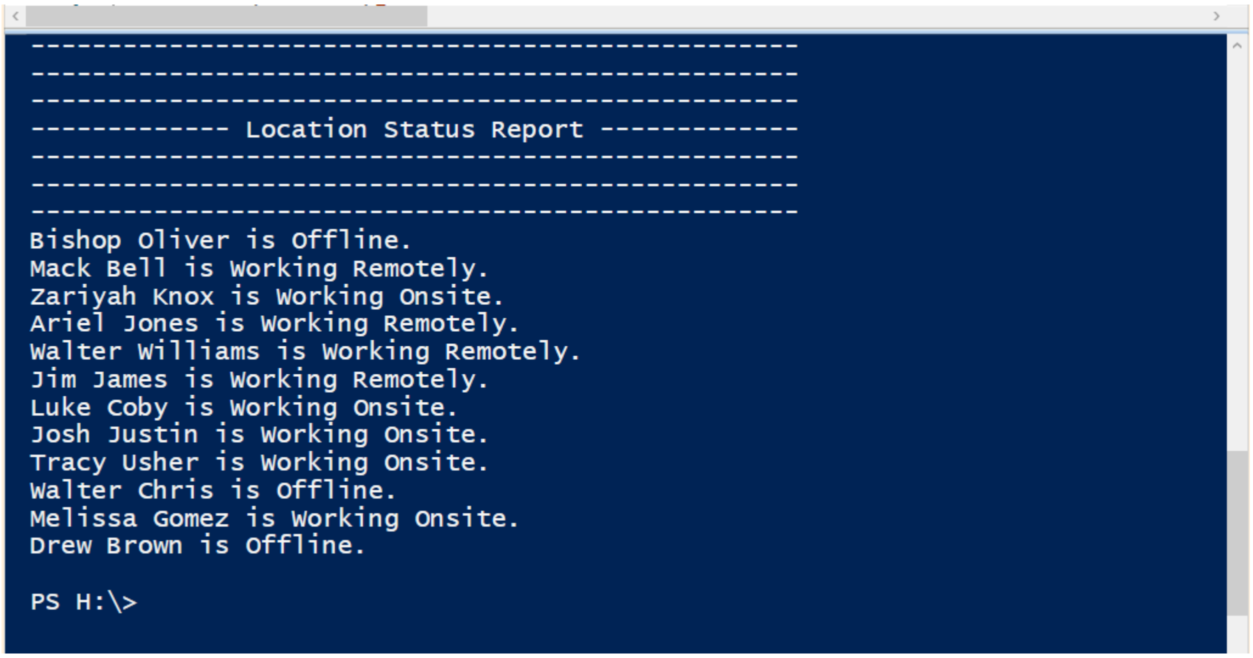
Network Adaptor Script

```
# Function to check if a remote computer is using a VPN
function GetNetworkStatus {
    param (
        [string]$computerName
    )

    try {
        # Test if the computer is online
        $isOnline = Test-Connection -ComputerName $computerName -Count 2 -Quiet

        if ($isOnline) {
            # Get network adapters on the remote computer
            $adapters = Get-WmiObject Win32_NetworkAdapter -ComputerName $computerName
-Filter 'NetConnectionStatus = 2' -ErrorAction Stop | Select-Object -ExpandProperty Description

            # Check if any adapter description contains keywords indicating a VPN
            if ($adapters -match 'VPN' -or $adapters -match 'Virtual') {
                return 'Working Remotely'  # VPN connection detected
            } else {
                return 'Working Onsite'  # Direct network connection detected
            }
        } else {
            return 'Offline'
        }
    } catch {
        # Handle RPC server unavailability errors
        if ($_.Exception.ErrorCode -eq 1722 -or $_.Exception.ErrorCode -eq 1355) {
            # Ignore RPC server unavailability errors
            return 'Offline'
        } else {
            # Other errors are ignored
            return 'Offline'
        }
    }
}


# Specify the full path to the text file
$filePath = "C:\Users\jwn2cpt\Desktop\LocationFinderPREP.txt"
Write-Output "-----------------------------------------------"
Write-Output "-----------------------------------------------"
Write-Output "-----------------------------------------------"
Write-Output "------------ Location Status Report ------------"
Write-Output "-----------------------------------------------"
Write-Output "-----------------------------------------------"
Write-Output "-----------------------------------------------"
```

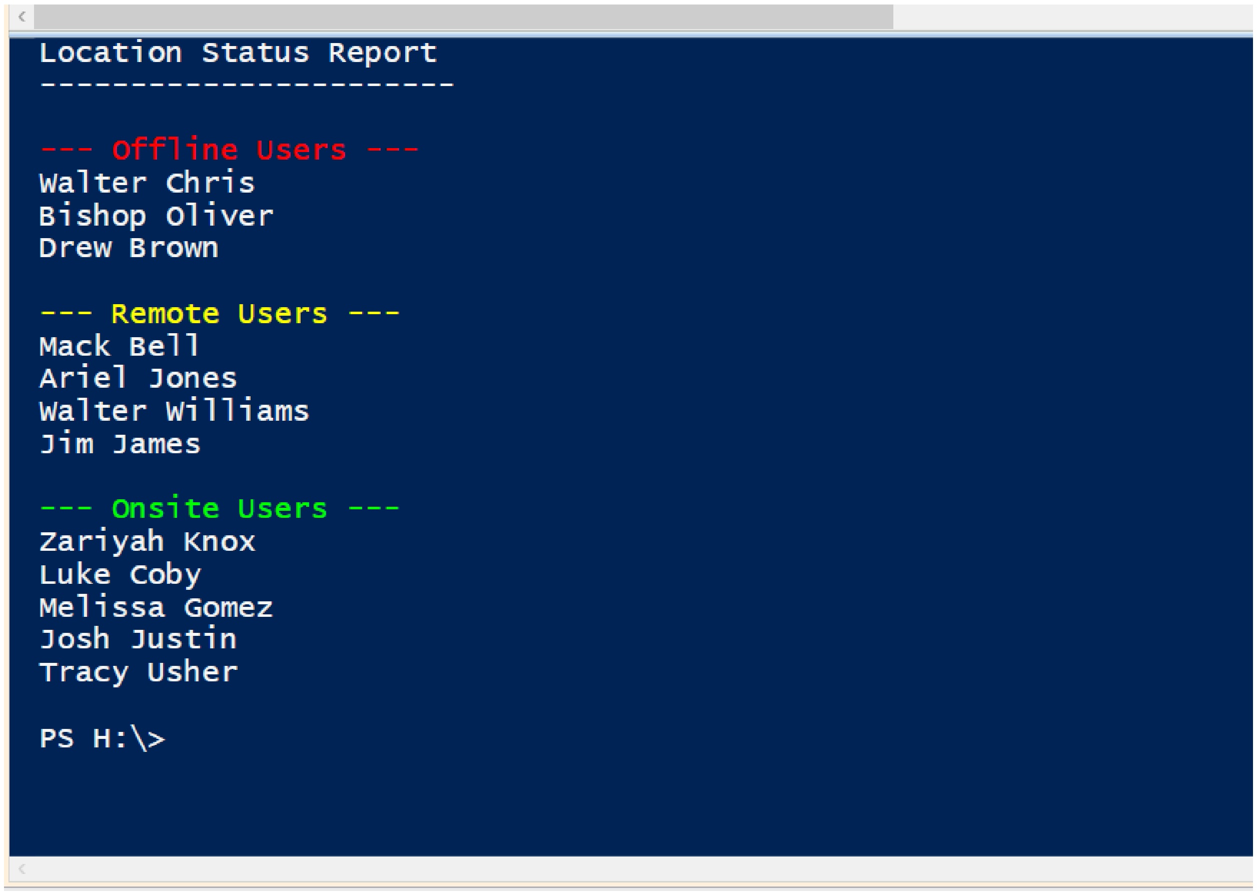
```
if (Test-Path $filePath) {
    # Read the content of the text file
    $fileContent = Get-Content -Path $filePath

    foreach ($line in $fileContent) {
        # Extract workstation and user information
        $workstation, $user = $line -split ' - '

        # Get the network status for the user
        $networkStatus = GetNetworkStatus -computerName $workstation

        # Display the result with specific phrases
        Write-Host "$user is $networkStatus."
    }
} else {
    Write-Host "File not found at $filePath"
}
```



```
Location Status Report
-----------------------

Onsite Users     Remote Users       Offline Users
------------     ------------       -------------
1. Zariyah Knox  1. Mack Bell       1. Bishop Oliver
2. Luke Coby     2. Ariel Jones     2. Walter Chris
3. Josh Justin   3. Walter Williams 3. Drew Brown
4. Tracy Usher   4. Jim James
5. Melissa Gomez


Script execution time: 12.370727 seconds
PS H:\>
```



```
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
------------- Location Status Report -------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
Bishop Oliver is Offline.
Mack Bell is Working Remotely.
Zariyah Knox is Working Onsite.
Ariel Jones is Working Remotely.
Walter Williams is Working Remotely.
Jim James is Working Remotely.
Luke Coby is Working Onsite.
Josh Justin is Working Onsite.
Tracy Usher is Working Onsite.
Walter Chris is Offline.
Melissa Gomez is Working Onsite.
Drew Brown is Offline.

PS H:\>
```



```
Location Status Report
-----------------------

--- Offline Users ---
Walter Chris
Bishop Oliver
Drew Brown

--- Remote Users ---
Mack Bell
Ariel Jones
Walter Williams
Jim James

--- Onsite Users ---
Zariyah Knox
Luke Coby
Melissa Gomez
Josh Justin
Tracy Usher

PS H:\>
```