GitFTPSync Application Functionality

=== GitFTPSync App implementation has been suspended due to multiple user conflicts ===

1.Who can use the application:

This application is developed for sitecore developers who develops specific components like front end of site core. This application helps them to push their local changes to Azure FTP location where the application is deployed.

2.Prerequisites to use the application.

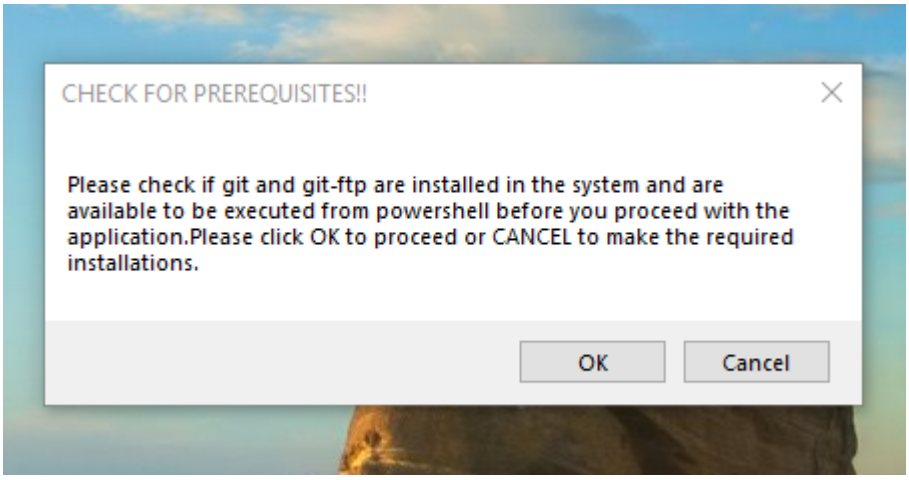
Git and Git-FTP has to be installed in the system. Git and Git FTP commands should be executable from Powershell.

Powershell version should be upgraded to >= 5.1

3.Application work Flow:

3.1. Pre-check for Git and Git-ftp :

As soon as you run the application you see the below given notification. This is to make sure that you have git and git-ftp installed in the system. Click on 'OK' if you have git and git-ftp installed in the system and their paths are added in environmental variables so that they are executable from Powershell. If they are not installed, then do the required installations before using the app.



To check if git and git-ftp are installed, you can run 'git' and 'git ftp' commands. You should see below, if :

1. git is installed.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM>
```

References to install git:
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

2. git-ftp is installed

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git ftp
git-ftp <action> [<options>] [<url>]
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM>
```

References for git-ftp installation :
https://github.com/git-ftp/git-ftp
https://blog.jongallant.com/2017/01/install-git-ftp-windows/
http://www.codingsips.com/install-configure-git-ftp-windows/

c. Powershell with version >= 5.1

```
PS C:\WorkStation\GitLocalRepo\GitFtpSync\master\GitFTPSync>  $PSVersionTable.PSVersion
>>

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      16299  98


PS C:\WorkStation\GitLocalRepo\GitFtpSync\master\GitFTPSync>
```

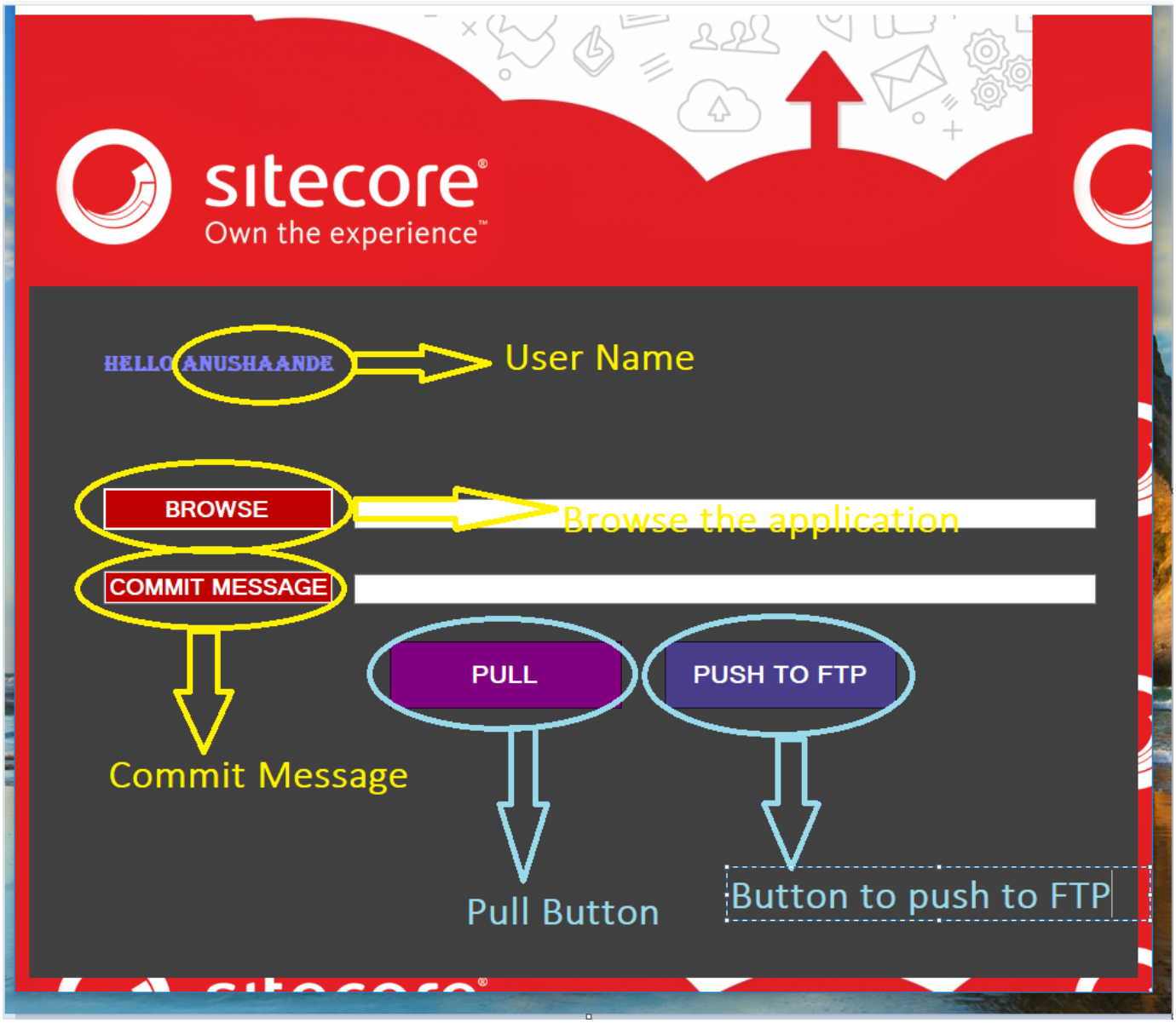
3.2. Click on 'OK':

Once you click on 'OK', you will be able to use the application. This is what you should see if application is installation is successful. You can see your name displayed

3.3 Fields of application page :

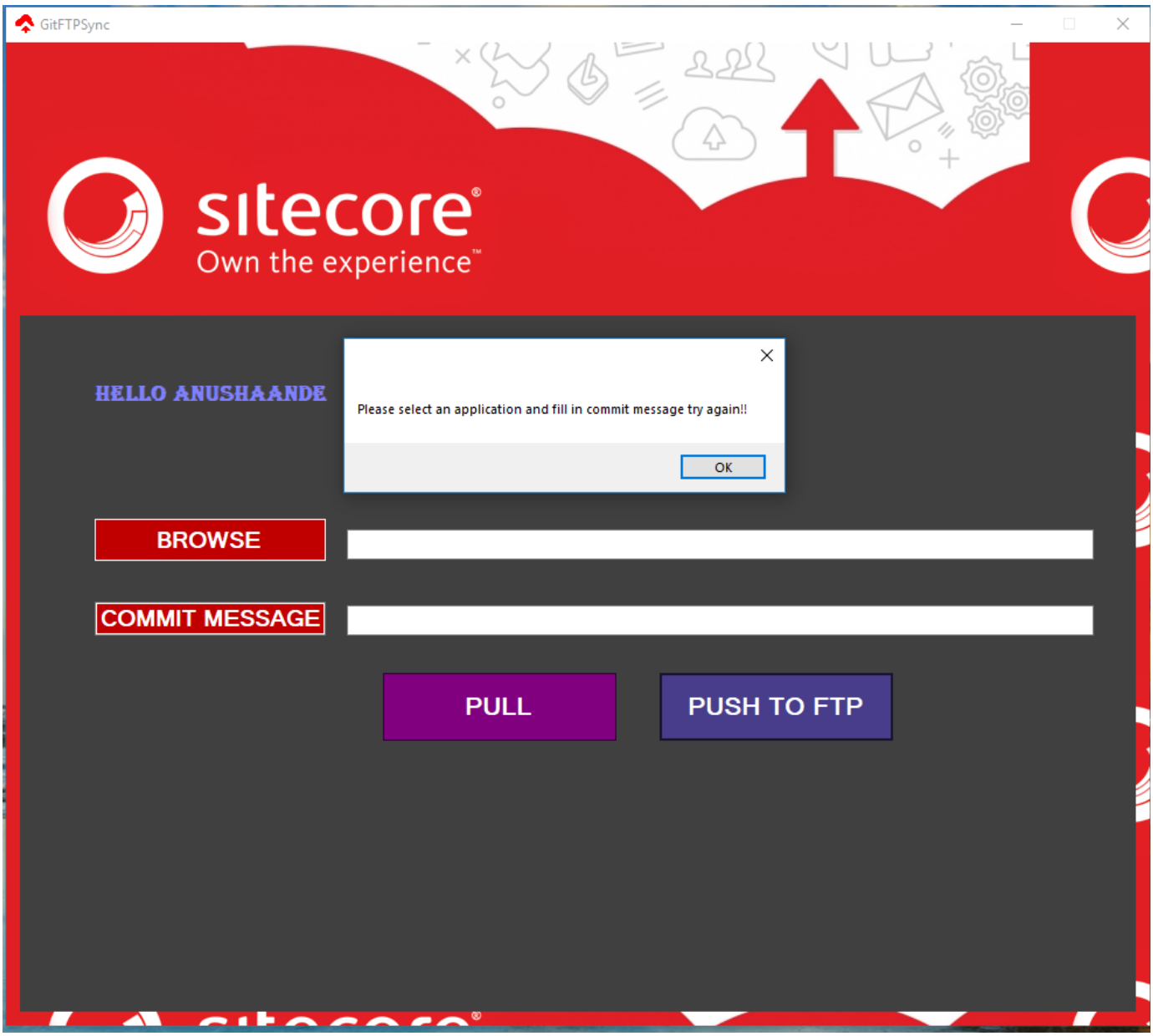
In this page, we find total of 7 fields.

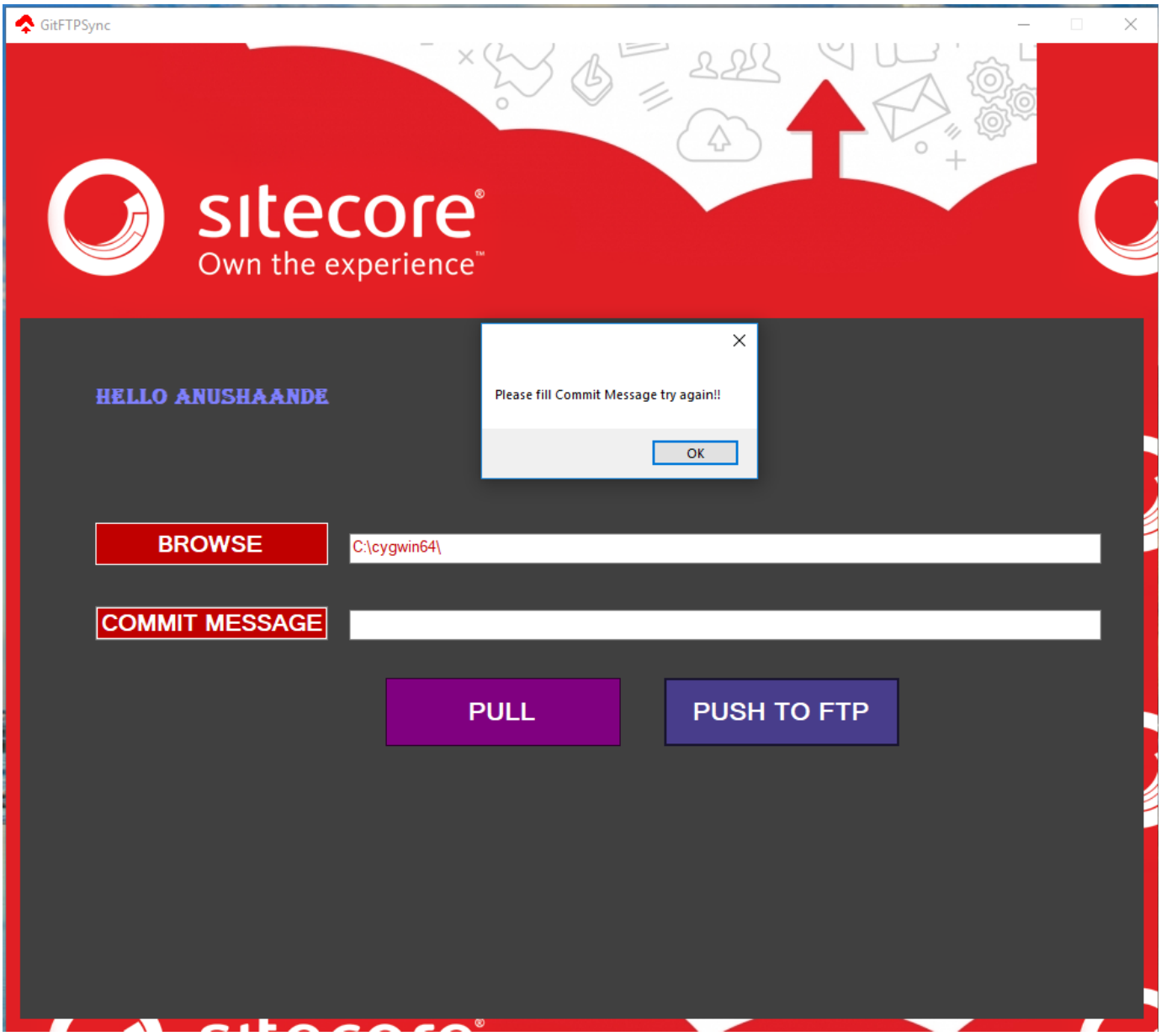
1. A label, that displays User Name
2. Browse button to browse the application folder.
3. Text box beside browse button to display folder that is browsed.
4. Label that says Commit message goes in the text box next to that.
5. Text box beside 'Commit Message' label to enter commit message.
6. PULL button to Pull the application code.
7. 'PUSH TO FTP' button to push application code to Azure FTP location.



3.4 : Mandatory Fields:

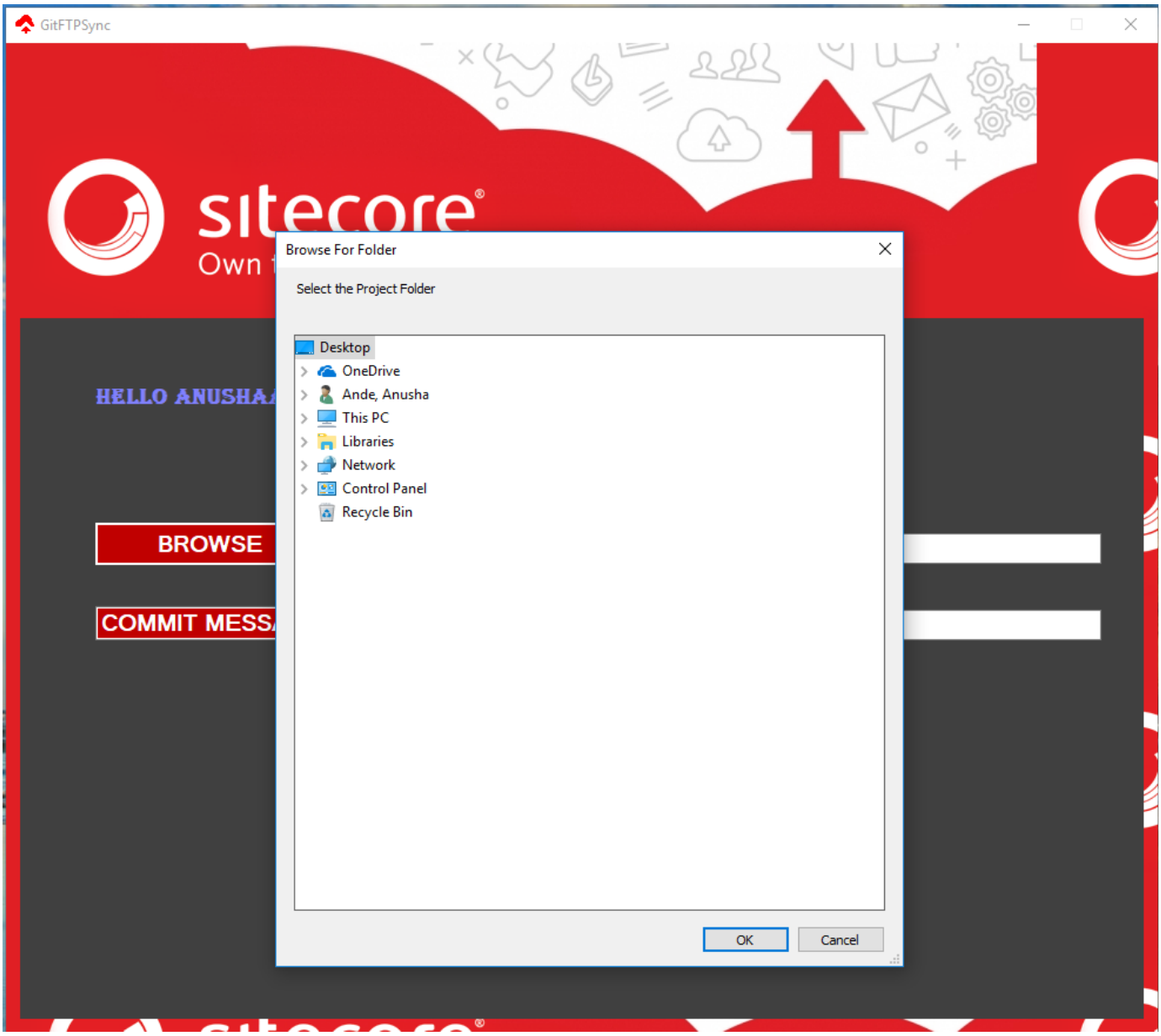
Before clicking on 'PUSH TO FTP' button, check if both the text boxes are not empty. Even if one filed is empty then, you will get an exception to check if fields are empty.


GitFTPSync
sitecore
Own the experience
HELLO ANUSHAANDE
Please select an application and fill in commit message try again!!
OK
BROWSE
COMMIT MESSAGE
PULL
PUSH TO FTP

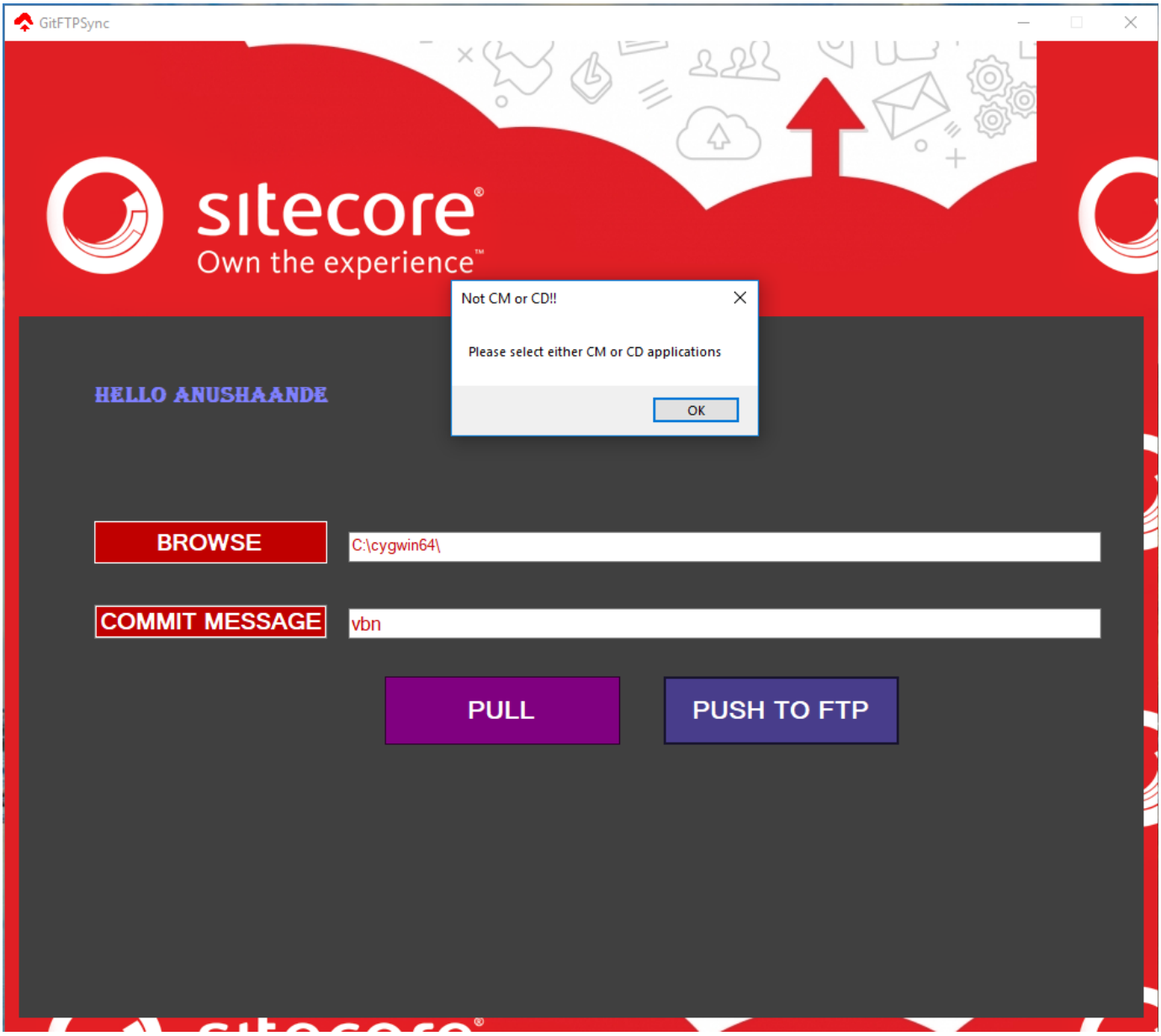


3.5. Application accepts only CM or CD:

On clicking "BROWSE" button, you will get a window, to select your application.

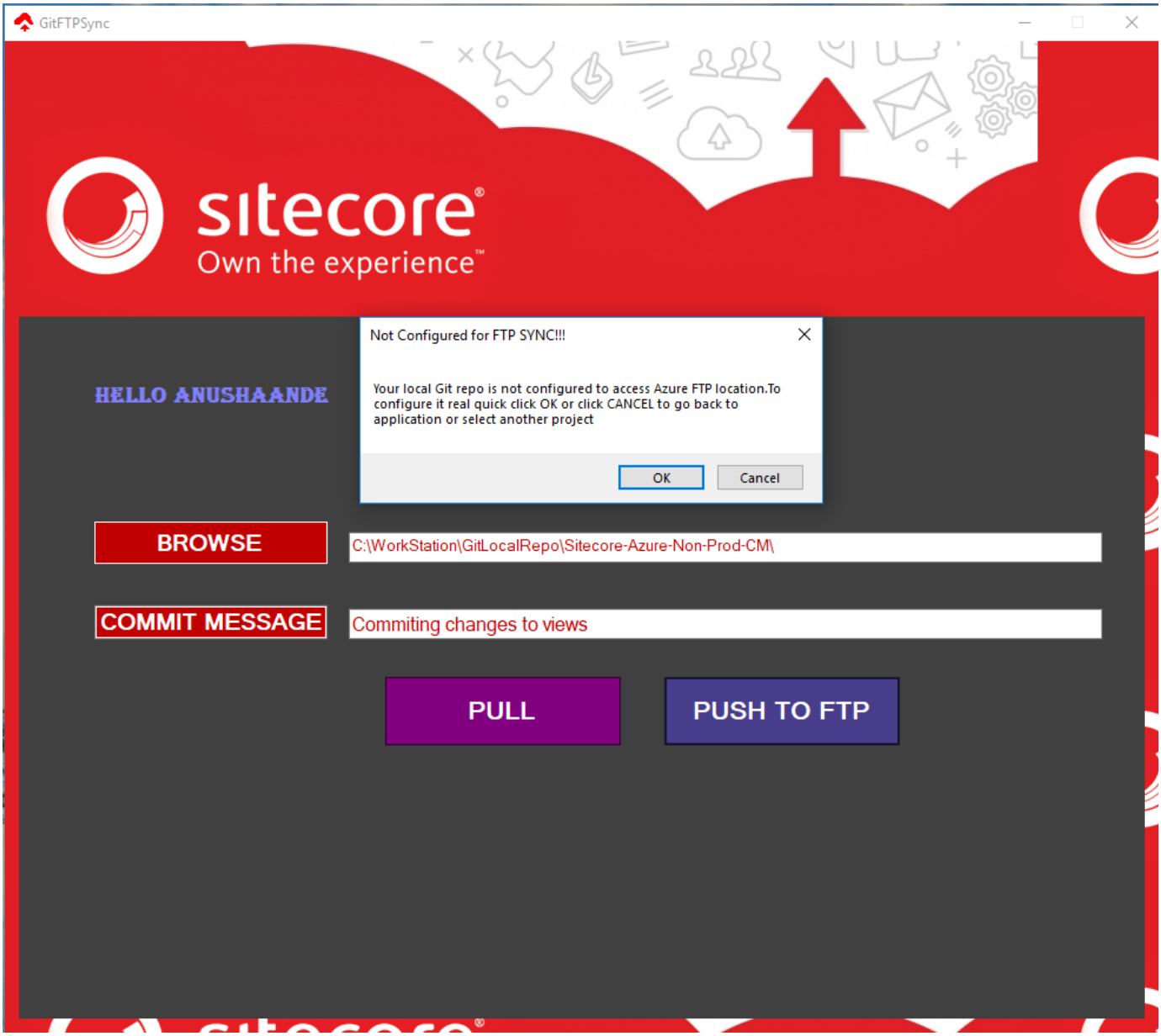


Once you browse for your sitecore folder and click on 'OK' , the application checks if it is CM or CD. If it is not either CM or CD, then you will get an alert asking you to select either CM or CD folder.

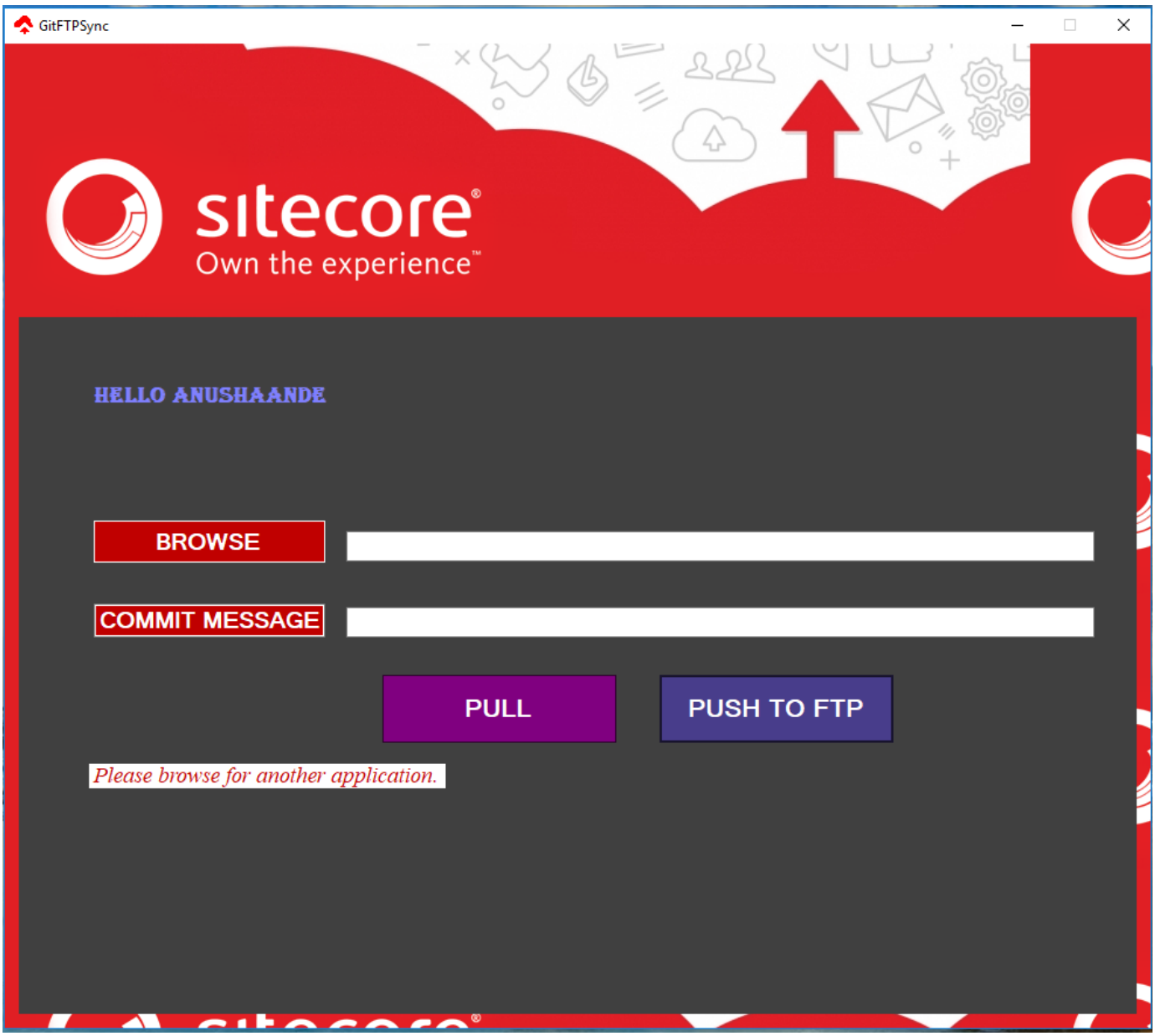


3.6 git ftp configuration.

Once you select the appropriate folder, application will check if that git repository is configured to push to its appropriate FTP location. If it is not configured, then you will get an alert letting you know that your local machine is not configured. Click on 'OK' to proceed further, or click on 'Cancel'.

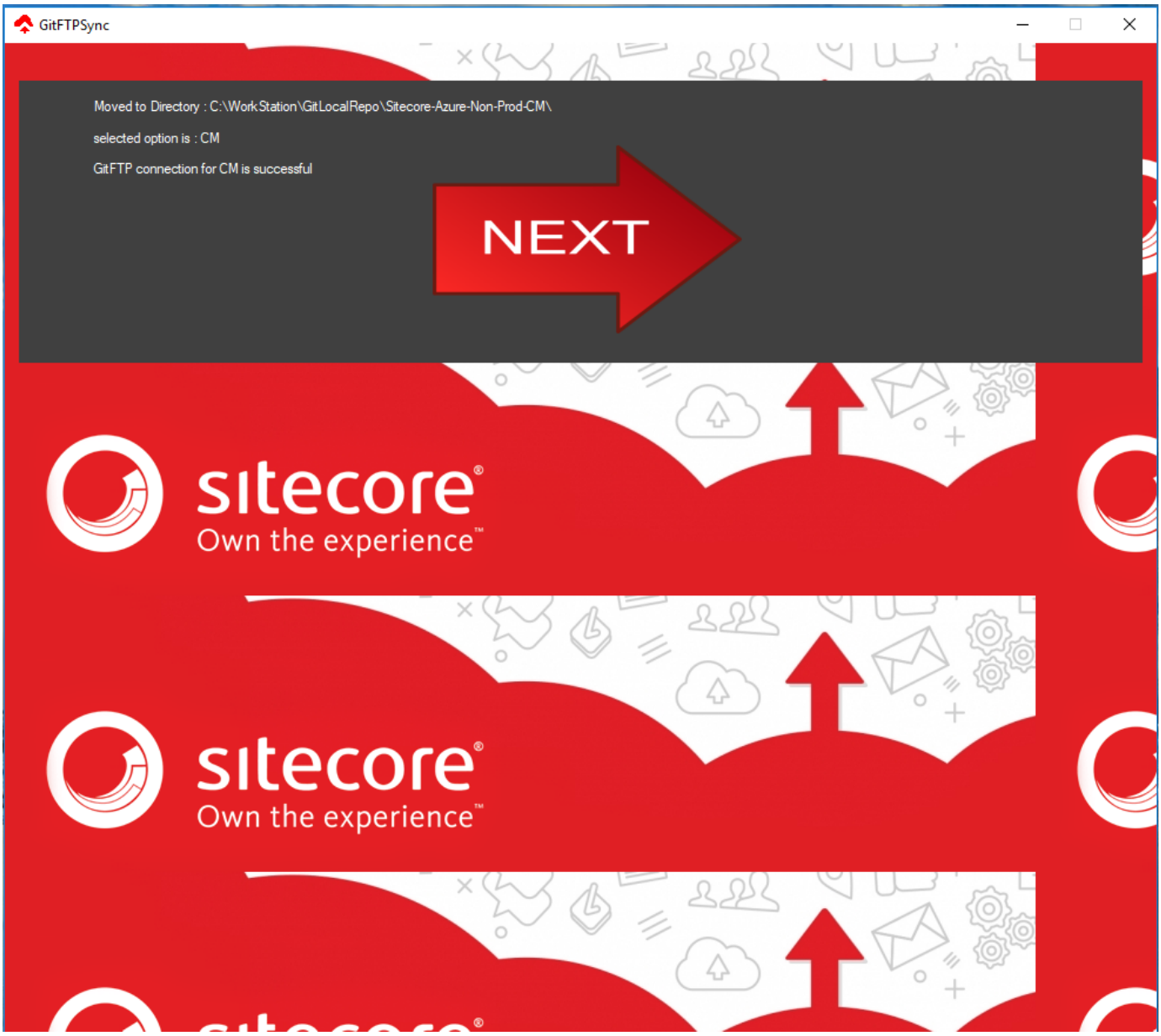


If you click on 'Cancel', you will stay in the same frame, and all the fields will be cleared for you, to select another folder.



If you click on 'OK', you will see this frame where you just need to click on 'CLICK TO CONTINUE' button, to proceed for configuration. Else you can click on 'CANCEL' to go back to main frame.

If you click on 'CLICK TO CONTINUE' button, your repository will be configured to access Azure FTP location and will have access to push code.



Once you click on 'NEXT' button, you will get to main frame. Here, you can continue to Push your changes to FTP, change the commit message again or simply exit.

3.7 Pushing Changes to FTP:

Once you click on 'PUSH TO FTP', there will be a bar showing you that process is running in background. Your changes in local repository, will start to be pushed to git ftp in background.

3.8 Folder restrictions:

As this application is developed to push changes from specific folders (Views , layouts, references for now), if there are changes in any other folder, then 'git ftp push' fai

1. Modifying a file from Views folder

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> vim .\Views\abc.txt
>>
```

```
fhjkhgkljnkldmngvmjhvbjkvhbg
sfjjkhfjkhjkljkdjfbdsvhfgdnsgbnfhnll
```

2. Modifying a file from current directory which is restricted .

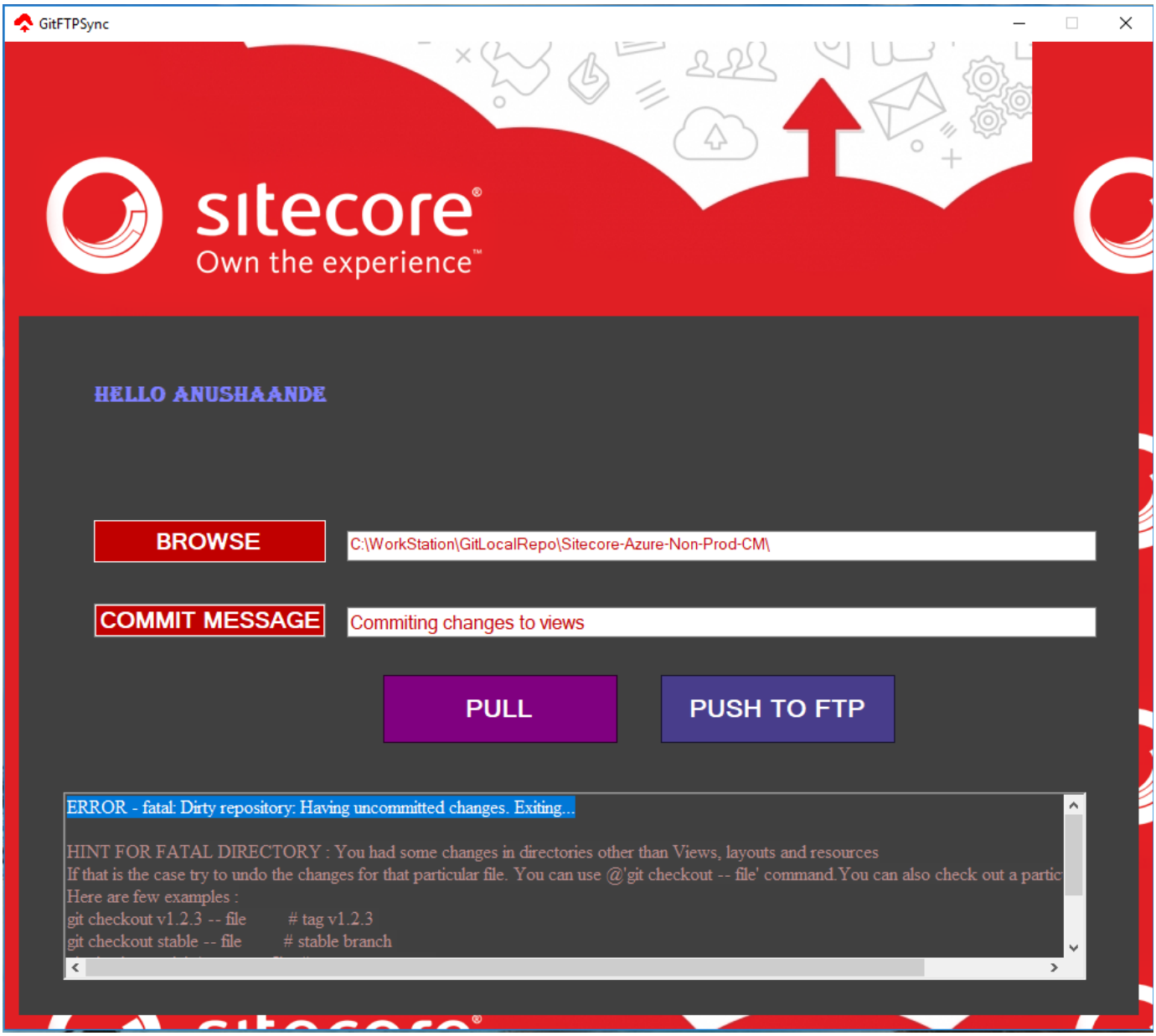
```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> vim .\abcMain.txt
```

```
This file is in root of the directorkmnnbdsjkfbjsfjnbgjgbky
```

3. We can see the files modified and need to be committed to local repository by running 'git status' command.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Views/.abc.txt.un~
        modified:   Views/abc.txt
        modified:   Views/abc.txt~
        modified:   abcMain.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .abcMain.txt.un~
        abcMain.txt~

no changes added to commit (use "git add" and/or "git commit -a")
```

4. As i have changes in main repository, 'git ftp push' failed. This is how error is displayed. In the description you can find commands that help you to revoke the changes made to those files



5. To check which files are restricted , run 'git status' command once again.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   abcMain.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .abcMain.txt.un~

no changes added to commit (use "git add" and/or "git commit -a")
```

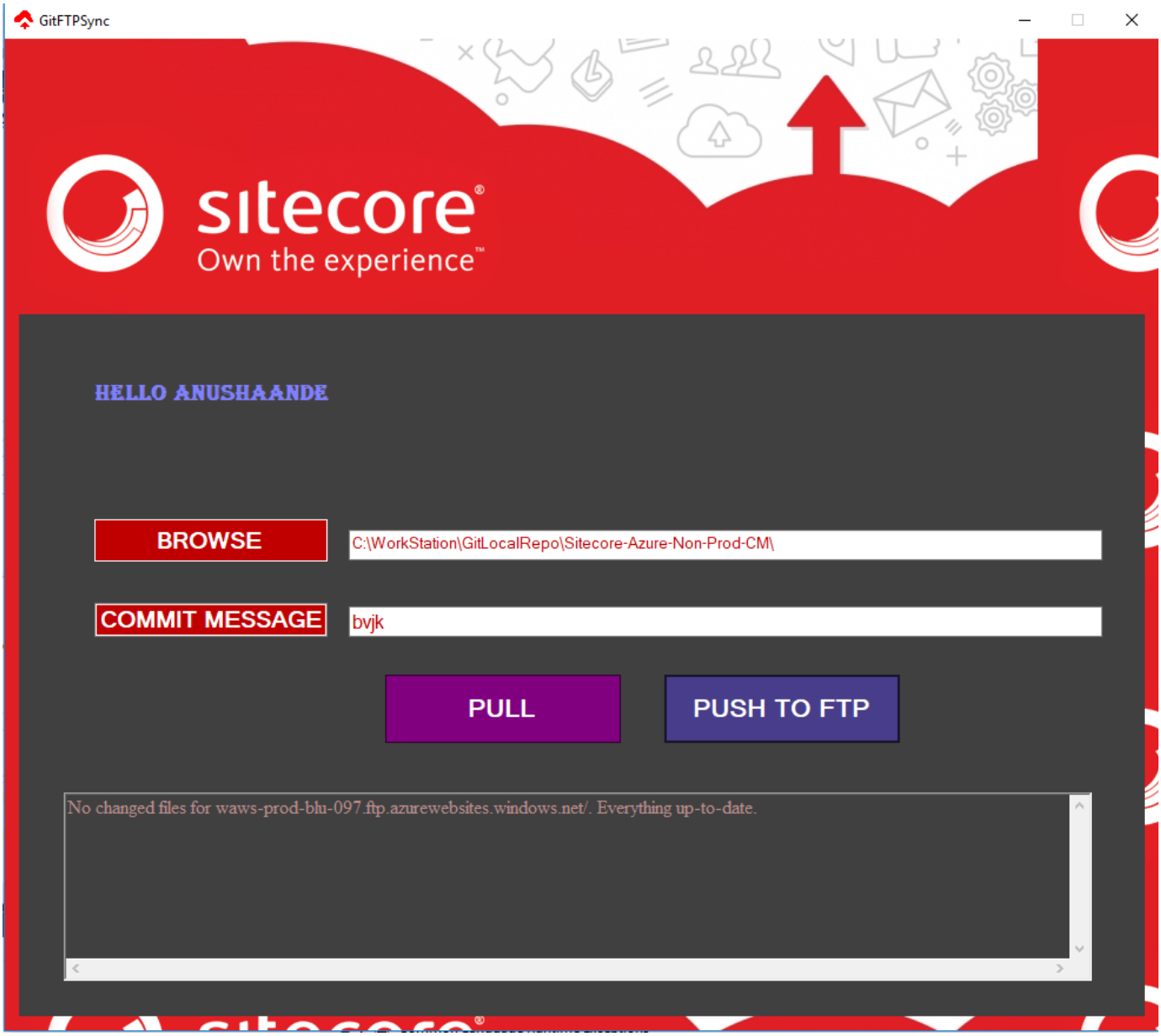
6. Sample example of how to revoke changes.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git checkout HEAD -- abcMain.txt
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .abcMain.txt.un~

nothing added to commit but untracked files present (use "git add" to track)
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> rm .abcMain.txt.un~
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git status
On branch master
nothing to commit, working tree clean
```

7. Once you make the required changes and click on 'PUSH TO FTP' button again, then again application starts pushing the changes, and you can see the progress bar.

8. Once if everything is fine, you can see the status in text box.

3.9. If there are no changes to Push:



3.10. Exit the application

To exit the application, click on X mark top right side of the frame. You will see an alert asking if you want to exit the application. Click on 'YES' twice to exit. Else click on 'NO'.

Commands List

1. git add * - adding changes from work station.
2. git add .\Views* - adding files from selected folders
3. git add .\layouts* - adding files from selected folders
4. git add .\resources* - adding files from selected folders
5. git commit -m "Commit commit" - Committing folders to local repository.
6. git ftp push - pushing changes to ftp location
7. git ftp catchup - catchup commit id in remote repository.

Trouble Shooting the application:

Trouble Shoot 1:

Few times application gets stuck after clicking on "Push To FTP" Button. The reason for this is because of the multiple commits to FTP location from different users. If that is the reason, we will see this on cli.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git status
On branch master
nothing to commit, working tree clean
```

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git ftp push
Unknown SHA1 object, make sure you are deploying the right branch and it is up-to-date.
Do you want to ignore and upload all files again? [y/N]: n
Aborting...
```

SHA1 is the commit id.

To get out of the situation, it is always a good idea to update your local repository with remote repository commit id. There will not be any changes to the files doing this.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git ftp -h
USAGE
        git-ftp <action> [<options>] [<url>]


DESCRIPTION
        git-ftp does FTP the Git way.

        It uses Git to determine which local files have changed since the last
        deployment to the remote server and saves you time and bandwidth by
        uploading only those files.

        It keeps track of the deployed state by uploading the SHA1 of the last
        deployed commit in a log file.

ACTIONS
        . init
                Does an initial upload of the latest version of all non-ignored
                git-tracked files to the remote server and creates .git-ftp.log
                file containing the SHA1 of the latest commit.

        . catchup
                Updates the commit id stored on the server.

        . push
                Uploads git-tracked files which have changed since last upload.

        . download (EXPERIMENTAL)
                Downloads changes from the remote host into your working tree.
                WARNING: It can delete local untracked files that are not
                listed in your .git-ftp-ignore file.

        . pull (EXPERIMENTAL)
                Downloads changes from the remote server into a separate commit
                and merges them into your current branch.

        . snapshot (EXPERIMENTAL)
                Downloads files into a new Git repository. Takes an additional
                optional argument as local destination directory. Example:
                `git-ftp snapshot ftp://example.com/public_html projects/example`

        . show
                Downloads last uploaded SHA1 from log and hooks `git show`.

        . log
                Downloads last uploaded SHA1 from log and hooks `git log`.

        . add-scope
                Add a scope (e.g. dev, production, testing).
```

Please visit Git FTP page online to understand more.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git ftp catchup
Last deployment changed from  to 0c9fee29df51d79a43c4f724e9a9db7365de7e02.
```

Once we execute " git ftp catchup " command, commit id of local repository is updated as of remote repository.

Hence this time Git FTP Push will be successful.

```
PS C:\WorkStation\GitLocalRepo\Sitecore-Azure-Non-Prod-CM> git ftp push
No changed files for waws-prod-blu-097.ftp.azurewebsites.windows.net/. Everything up-to-date.
```

And application does not stuck at processing git ftp push.