Lesson 3: Mastery Project Installation 2

# Submission

1. Based on what you proposed, describe what will you be submitting for Installation 2.

This installation consists of the implementation of the file transfer (upload and download files) using FTP between server and client. Whenever a file/list of files are uploaded to the server, it is stored at a location in the server and server acknowledges the client with the storage location path. The storage location is a folder path where the files are stored, and these files will be maintained in the server until the expiration time is reached (which will be covered in Installation 3). Whenever client requests for the file download using the storage location path, server responds with the list of filenames stored in the given location to the client. By looking at the list of file names, user makes a specific file download request to the server.

1. What code files are you submitting for Installation 2?

We are submitting client.py and server.py files as part of Installation 2. The code files contain the implementation of uploading and downloading the files to and forth between client and server. The code includes the implementation of functions download\_file(), download\_multiple\_files(), upload\_file(), upload\_multiple\_files(), list\_files(). We are also sharing the StoragePathService.py, UtilityService.py, Config.py.

1. Provide a URL to your video demo. Remember, the contents will be evaluated against what you proposed for the demonstration in Lesson 1.

<https://youtu.be/GKSsylV1EIg>

1. Paste the Installation 2 Evaluation Rubric that you developed in your proposal below. This will be used to help evaluate your Installation.

|  |  |  |
| --- | --- | --- |
|  | Fail (0) | Pass (1) |
| [Item 1] | This installation fails to demonstrate mastery if server cannot upload the files to the server via FTP. | This installation succeeds at demonstrating mastery by uploading the files to the server via FTP. |
| [Item 2] | This installation fails to demonstrate mastery if server cannot respond with the storage location path after the file upload done via command line. | This installation succeeds at demonstrating mastery by responding to the client with storage location path after the file upload done via command line. |
| [Item 3] | This installation fails to demonstrate mastery if server cannot respond with the list of files in the storage location provided by the client. | This installation succeeds at demonstrating mastery if the server responds with the list of files in the storage location, which is provided by the client via command line. |
| [Item 4] | This installation fails to demonstrate mastery if server cannot download the requested files via FTP. | This installation succeeds at demonstrating mastery by downloading the requested files from the server via FTP. |
|

1. What were some challenges that you encountered during the implementation of this installation and what are some specific ways you can work to overcome these challenges?

Here are the following challenges faced while implementation:

1. **Storage path issue :** For creating files in storage location, we need absolute path. But this absolute path cannot be static. For our storage location path creation in different systems, we have created a Storage Path Service(which helps in file creation or accessing functionalities) which uses Config.py and create or access files in the given location.
2. **Encoding and decoding method :** The data being transferred in the socket is usually in bytes. In few occurrences, when we missed encoding or decoding of data, we faced related issues. So, we have created a common service named Utility Service which helped using in resolving encoding and decoding issues.
3. **Folder import issue :** While working with common folders providers, contracts, we faced difficulty in importing the classes as these classes are not in the root structure of code files client.py and server.py. So, we have used relative path technique using sys package and resolved this issue.