

CSE 5306: DISTRIBUTED SYSTEMS

PROJECT 1 REPORT

I have neither given nor received unauthorized assistance on this work. I will not post the project description and the solution online.

Sign:

Manan Arora (1002143328)

Muskan Jain (1002033280)

Date:

18th Feb 2024

18th Feb 2024

INTRODUCTION:

In this programming project, we have implemented a simple file upload and download service and a computation service using **remote procedure call (RPC)** - based communication.

The file server in our project supports four basic operations: **UPLOAD**, **DOWNLOAD**, **DELETE**, and **RENAME** a file. The computation server provides a set of predefined procedures that can be called from a client. The server should support `add(i, j)`, and `sort(array A)`.

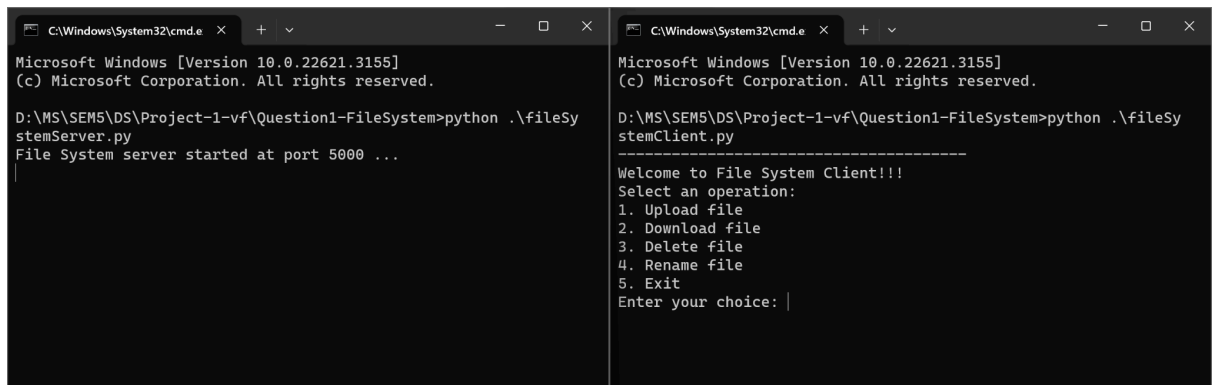
We have used Python as our programming language for this project.

Implementation:

Part-1:

We have implemented a multi-threaded file server that supports **UPLOAD**, **DOWNLOAD**, **DELETE**, and **RENAME** file operations. We have used different folders (downloads, uploads) to hold files downloaded to the client or uploaded to the server. We have used gRPC as our RPC framework.

1. Starting the server and client



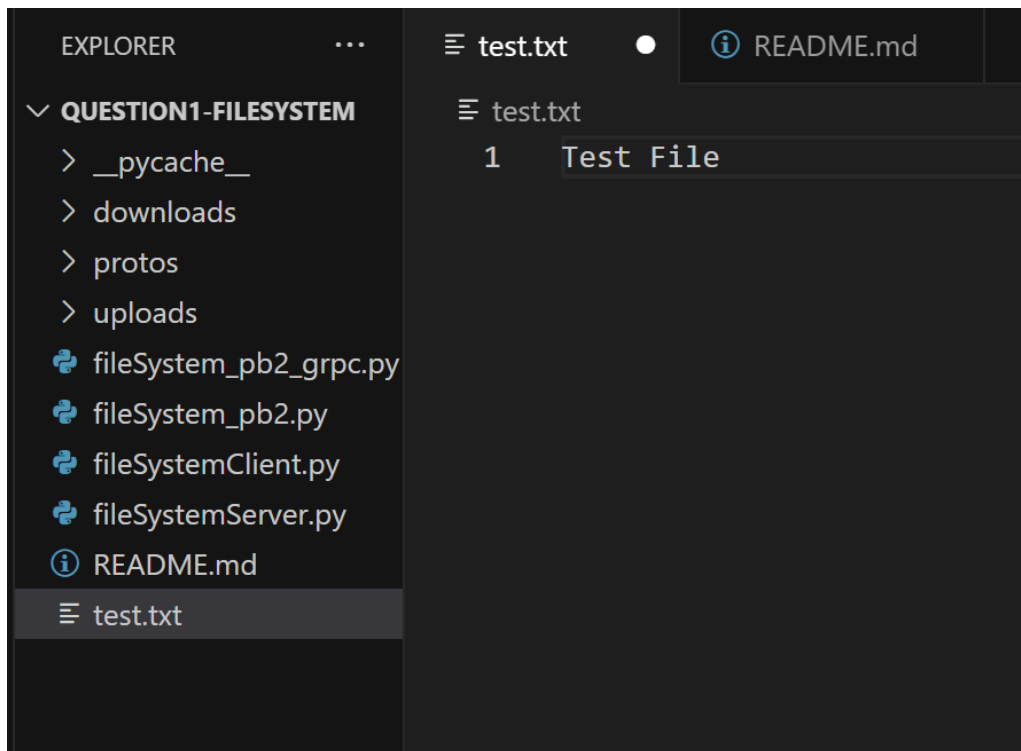
```
C:\Windows\System32\cmd.e x + v - □ X
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

D:\MS\SEM5\DS\Project-1-vf\Question1-FileSystem>python .\fileSystemServer.py
File System server started at port 5000 ...

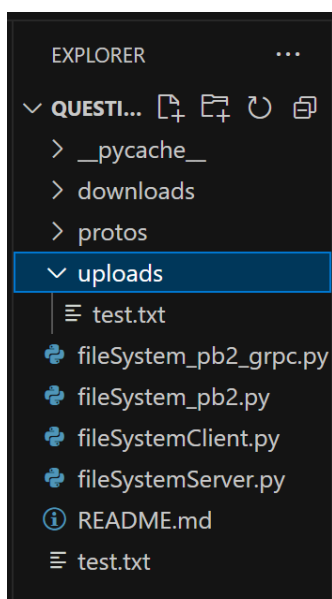
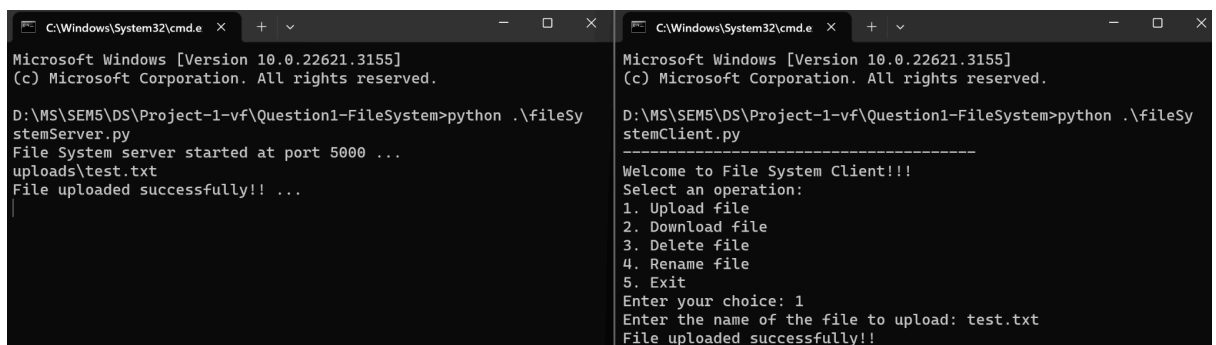
C:\Windows\System32\cmd.e x + v - □ X
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

D:\MS\SEM5\DS\Project-1-vf\Question1-FileSystem>python .\fileSystemClient.py
-----
Welcome to File System Client!!!
Select an operation:
1. Upload file
2. Download file
3. Delete file
4. Rename file
5. Exit
Enter your choice: |
```

2. Creating a test file in the directory (test.txt)



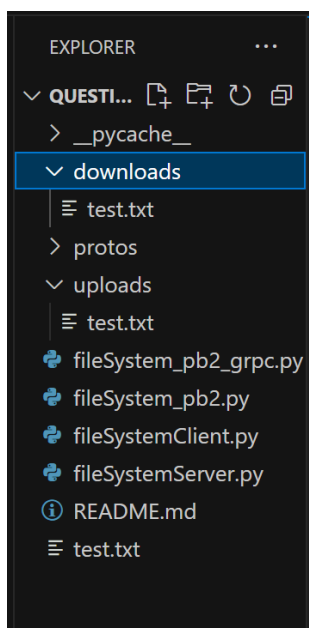
3. Upload Function:



4. Download Function:

```
D:\MS\SEM5\DS\Project-1-vf\Question1-FileSystem>python .\fileSystemServer.py
File System server started at port 5000 ...
uploads\test.txt
File uploaded successfully!! ...
test.txt
uploads\test.txt
File downloaded successfully!!!
```

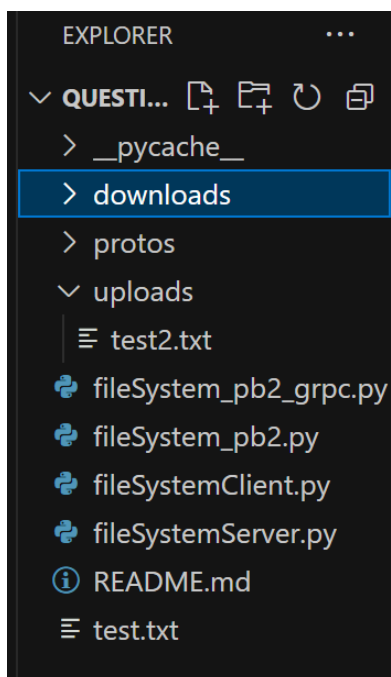
```
Welcome to File System Client!!!
Select an operation:
1. Upload file
2. Download file
3. Delete file
4. Rename file
5. Exit
Enter your choice: 2
Enter the name of the file to be downloaded: test.txt
File downloaded successfully!!!
```



5. Rename Function:

```
D:\MS\SEM5\DS\Project-1-vf\Question1-FileSystem>python .\fileSystemServer.py
File System server started at port 5000 ...
uploads\test.txt
File uploaded successfully!! ...
test.txt
uploads\test.txt
File downloaded successfully!!!
File renamed successfully!!!
|
```

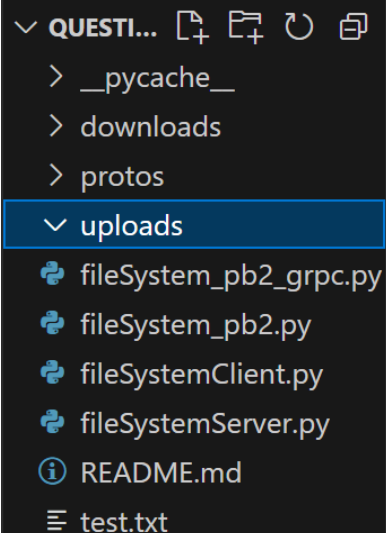
```
Welcome to File System Client!!!
Select an operation:
1. Upload file
2. Download file
3. Delete file
4. Rename file
5. Exit
Enter your choice: 4
Enter the name of the file to be renamed: test.txt
Enter the new name for this file: test2.txt
Rename_response: File renamed successfully!!
```



6. Delete Function:

```
D:\MS\SEM5\DS\Project-1-vf\Question1-FileSystem>python .\fileSystemServer.py
File System server started at port 5000 ...
uploads\test.txt
File uploaded successfully!! ...
test.txt
uploads\test.txt
File downloaded successfully!!!
File renamed successfully!!!
File deleted successfully!!!
|
```

```
Welcome to File System Client!!!
Select an operation:
1. Upload file
2. Download file
3. Delete file
4. Rename file
5. Exit
Enter your choice: 3
Enter the name of the file to be deleted: test2.txt
Delete_response: File deleted successfully!!
```



```
▼ QUESTI... [Icons]
  > __pycache__
  > downloads
  > protos
  ▼ uploads
    📄 fileSystem_pb2_grpc.py
    📄 fileSystem_pb2.py
    📄 fileSystemClient.py
    📄 fileSystemServer.py
    ⓘ README.md
    ≡ test.txt
```

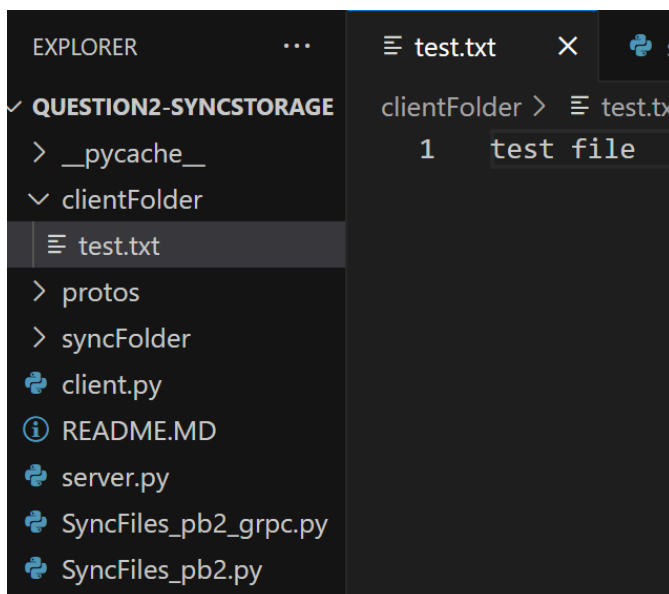
Part-2:

For this part, we have created a synchronized storage service that features a Dropbox-like functionality that syncs the changes that are made on the client folder (clientFolder) to the server folder (syncFolder). To implement this functionality we have used a library called watchdog that enables us to keep track of the changes that are made to the client folder.

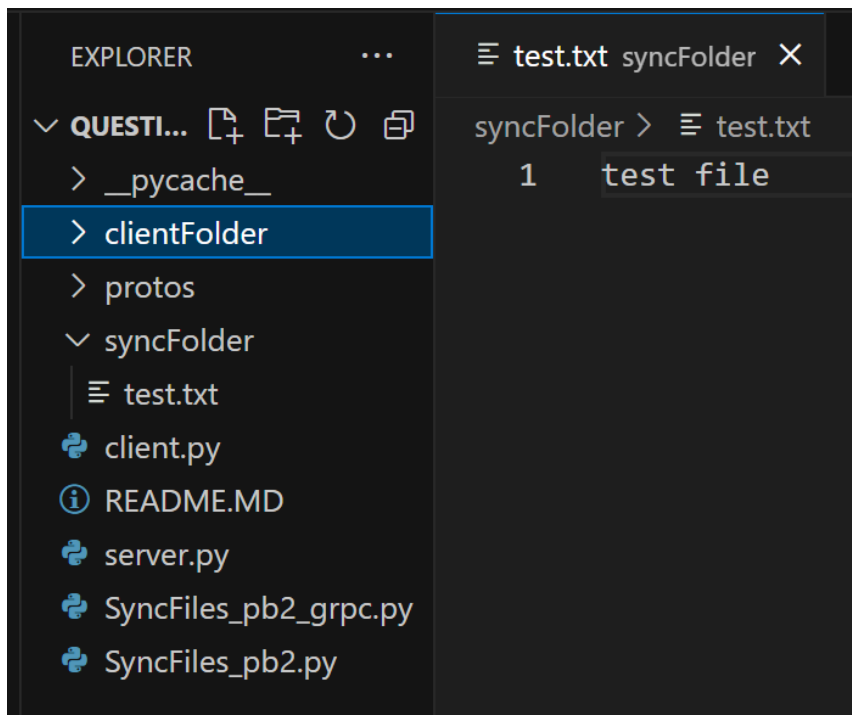
1. Starting the server and the client

<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question2-SyncStorage>python .\server.py Server Started at port 50050!!</pre>	<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question2-SyncStorage>python .\client.py Client running!!!</pre>
---	--

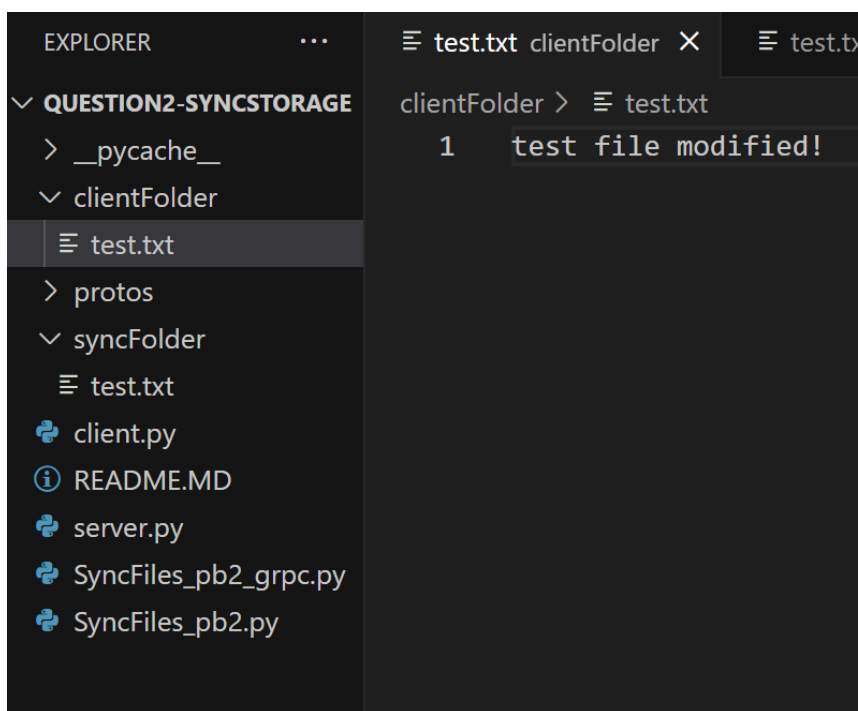
2. Creating a test file (test.txt) in the client folder (clientFolder)

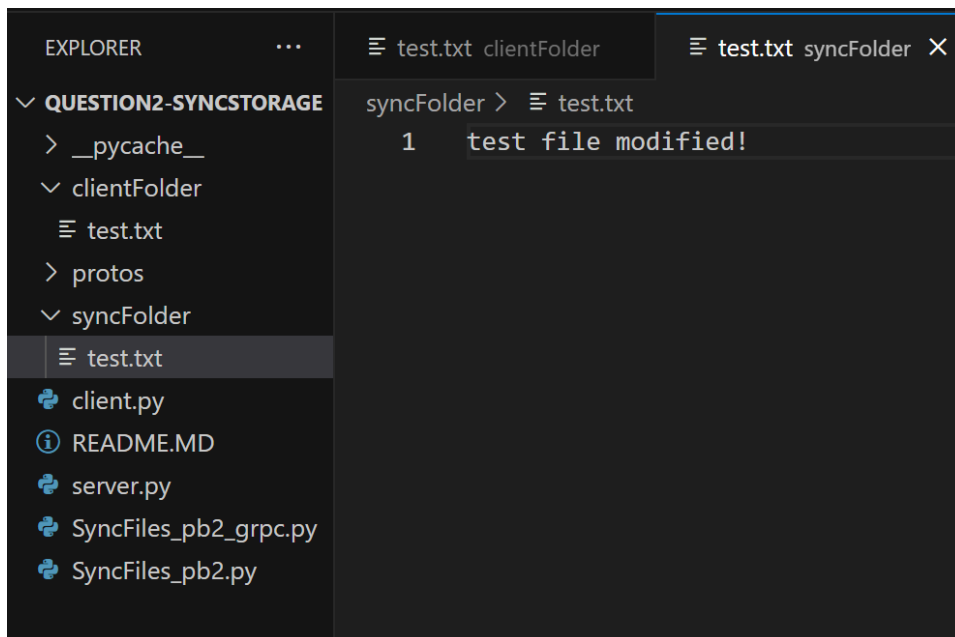


3. Test File (test.txt) synced to the server folder (syncFolder)

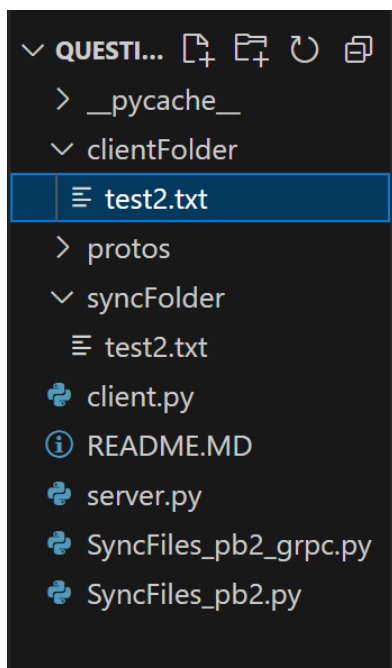


4. Modifying the content of the test file (test.txt)

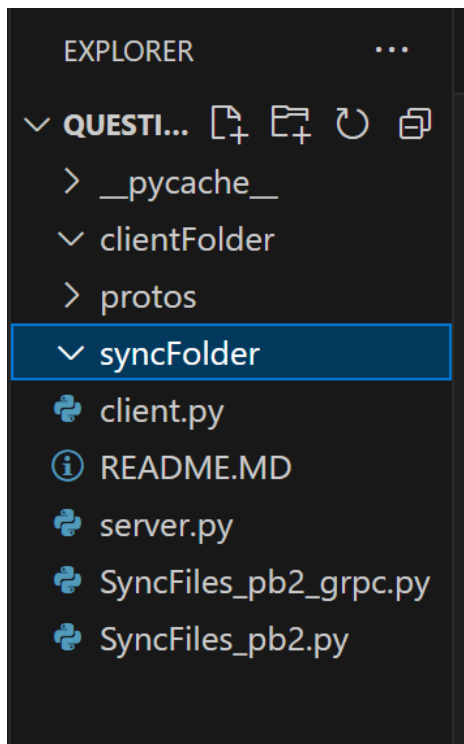




5. Renaming the test file (test.txt -> test2.txt)



6. Deleting the test file



Part-3:

In this part, we have implemented the computation server using both synchronous and asynchronous RPCs.

Synchronous RPC:

1. Starting the server (syncServer.py) and client (syncClient.py)

<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Synchronous>py thon .\syncServer.py Server started at port 50052</pre>	<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Synchronous>py thon .\syncClient.py ----- Synchronous Computation Client!!! 1. Add Operation 2. Sort Operation 3. Exit Enter the number of the operation: </pre>
--	---

2. Selecting the operation that we want to choose
3. For the Add operation,

<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Synchronous>py thon .\syncServer.py Server started at port 50052 Add function called</pre>	<pre>Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Synchronous>py thon .\syncClient.py ----- Synchronous Computation Client!!! 1. Add Operation 2. Sort Operation 3. Exit Enter the number of the operation: 1 Enter the 1st number: 3 Enter the 2nd number: 6 Result of add operation: 9 ----- Synchronous Computation Client!!!</pre>
--	--

4. For the Sort operation,

```
D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Synchronous>py
thon .\syncServer.py
Server started at port 50052
Add function called
Sort function called
|
```

```
1. Add Operation
2. Sort Operation
3. Exit
Enter the number of the operation: 2
Enter the number of elements in the list: 5
Enter element, 1: 9
Enter element, 2: 3
Enter element, 3: 1
Enter element, 4: 6
Enter element, 5: 12
Result of sort operation:  [1, 3, 6, 9, 12]
-----
Synchronous Computation Client!!!
```

Asynchronous RPC:

1. Starting the server (asyncServer.py) and the client (asyncClient.py)

```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Asynchronous>p
ython .\asyncServer.py
Server started at port 50055
```

```
D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Asynchronous>p
ython .\asyncClient.py
-----
Asynchronous Computation Client!!!
1. Add Operation
2. Sort Operation
3. Exit
```

2. Selecting the operation we want to choose
3. For Add operation,

<pre> Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Asynchronous>p ython .\asyncServer.py Server started at port 50055 Request received! Add 2 and 6 </pre>	<pre> Microsoft Windows [Version 10.0.22621.3155] (c) Microsoft Corporation. All rights reserved. D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Asynchronous>p ython .\asyncClient.py ----- Asynchronous Computation Client!!! 1. Add Operation 2. Sort Operation 3. Exit Enter the number of the operation: 1 Enter the 1st number: 2 Enter the 2nd number: 6 Request sent to add: 2 and 6 1. Add Operation 2. Sort Operation 3. Exit Enter the number of the operation: Result: 2 + 6 = 8 </pre>
--	---

4. For Sort operation,

```

D:\MS\SEM5\DS\Project-1-vf\Question3-Computation\Asynchronous>p
ython .\asyncServer.py
Server started at port 50055
Request received! Add 2 and 6
Request recieved! Sort : [6, 3, 2, 8]

```

```

1. Add Operation
2. Sort Operation
3. Exit
Enter the number of the operation:
Result: 2 + 6 = 8
2
Enter the number of elements in the list: 4
Enter element, 1: 6
Enter element, 2: 3
Enter element, 3: 2
Enter element, 4: 8
Request sent to sort: [6, 3, 2, 8]
1. Add Operation
2. Sort Operation
3. Exit
Enter the number of the operation:
Given List: [6, 3, 2, 8] and Sorted List: [2, 3, 6, 8]

```

OUR LEARNINGS:

1. We are now able to understand the function of the client-server system.
2. Can establish client-server communication using gRPC
3. Understood how the library watchdog works for synchronization of the file system
4. Understood in-depth about the synchronous and asynchronous RPC calls using gRPC and threads.

ISSUES ENCOUNTERED:

1. While implementing Part 2, we were unable to add the deletion synchronization, but after some research, we came across the watchdog library that helped us to implement this feature
2. We encountered a bug in Part 1, if the file that was uploaded to the server was empty, then while trying to download it to the client folder, it was not being downloaded. So we tested the code with a few changes and rectified this bug
3. We tried using asyncio which is a Python asynchronous framework but were unable to implement it correctly so instead we used threads for the asynchronous functionality

CONTRIBUTION:

Both of us contributed equally to the project either remotely or by collaborating after classes.